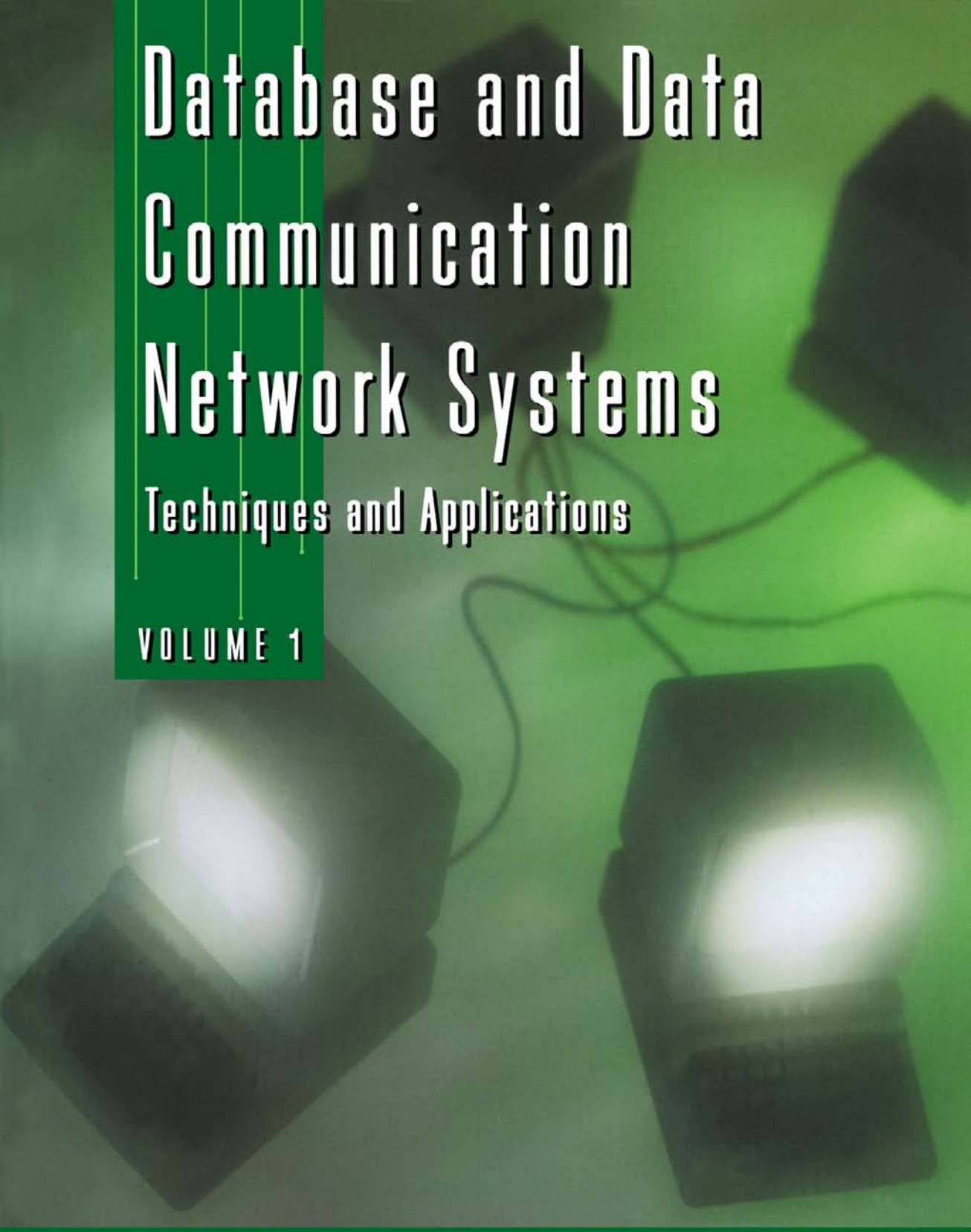
Database and Data Communication Network Systems

Techniques and Applications

VOLUME 1



Edited by

CORNELIUS T. LEONDES

DATABASE AND DATA COMMUNICATION NETWORK SYSTEMS

Techniques and Applications

VOLUME I

This Page Intentionally Left Blank

DATABASE AND DATA COMMUNICATION NETWORK SYSTEMS

Techniques and Applications

VOLUME I

Edited by

Cornelius T. Leondes

Professor Emeritus
University of California
Los Angeles, California

ACADEMIC PRESS
An imprint of Elsevier Science

Amsterdam Boston London New York Oxford Paris
San Diego San Francisco Singapore Sydney Tokyo

Front cover: Digital Image © 2002 PhotoDisc.

This book is printed on acid-free paper. ♾

Academic Press
An imprint of Elsevier Science
525 B Street, Suite 1900, San Diego, California 92101-4495, USA
http://www.academicpress.com

Academic Press
84 Theobalds Road, London WC1X 8RR, UK
http://www.academicpress.com

Library of Congress Catalog Card Number: 20016576

International Standard Book Number: 0-12-443895-4 (set)
International Standard Book Number: 0-12-443896-2 (volume 1)
International Standard Book Number: 0-12-443897-0 (volume 2)
International Standard Book Number: 0-12-443898-9 (volume 3)

PRINTED IN THE UNITED STATES OF AMERICA
02 03 04 05 06 07 MM 9 8 7 6 5 4 3 2 1

CONTENTS

CONTENTS OF VOLUME I

CONTENTS OF VOLUME 2

CONTENTS OF VOLUME 3

CONTRIBUTORS

Numbers in parentheses indicate the pages on which the authors' contributions begin.

C. Apostolas (859) Network Communications Laboratory, Department of Informatics, University of Athens, Panepistimioupolis, Athens 15784, Greece

Bernardino Arcay (825) Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Gill Barequet (277) The Technion—Israel Institute of Technology, Haifa 32000, Israel

József Békési (233) Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

Elisa Bertino (175) Dipartimento di Scienze dell'Informazione, Università di Milano, 20135 Milano, Italy

Hsin-Horng Chen (203) Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

Shyi-Ming Chen (203) Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan, Republic of China

Jian Chen (663) Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Carlos Dafonte (825) Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Timon C. Du (1) Department of Industrial Engineering, Chung Yuan Christian University, Chung Li, Taiwan 32023; and Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Shatin, NT Hong Kong

Elena Ferrari (175) Dipartimento di Chimica, Fisica e Matematica, Università dell'Insubria - Como, Italy

Gábor Galambos (233) Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

Jacob Gan (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

Mario Gerla (693) Computer Science Department, University of California—Los Angeles, Los Angeles, California 90095

Stathes Hadjiefthymiades (635) Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

M. Ahsan Akhtar Hasin[1] (417) Industrial Systems Engineering, Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

Li Hong-Yi (901) Advanced Wireless Networks, Nortel Research, Nepean, Ontario, Canada K2G 6J8

Hiroshi Ishikawa (77) Department of Electronics and Information Engineering, Tokyo Metropolitan University, Tokyo 192-0397, Japan

Hiroo Iwata (549) Institute of Engineering Mechanics and Systems, University of Tsukuba, Tsukuba 305-8573, Japan

Rong-Hong Jan (719) Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

Chua Chee Kai (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

Myoung Ho Kim (41) Department of Computer Science, Korea Advanced Institute of Science and Technology, Taejon 305-701, Korea

Wataru Kishimoto (783) Department of Information and Image Sciences, Chiba University, Chiba 263-8522, Japan

Shmuel Tomi Klein (573) Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel

Marco Lazzari[2] (751) ISMES, Via Pastrengo 9, 24068 Seriate BG, Italy

Doheon Lee (41) Department of BioSystems, Korea Advanced Institute of Science and Technology, Daejon, Republic of Korea

[1] Current address: Industrial and Production Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh

[2] Current address: Dipartimento de Scienze della Formazione, Università di Bergamo, Bergamo 24029, Italy

Chiang Lee (123) Institute of Information Engineering, National Cheng-Kung University, Tainan, Taiwan, Republic of China

Mong Li Lee (497) School of Computing, National University of Singapore, Singapore 117543

Kunwoo Lee (517) School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea

Ngoh Lek-Heng (901) SingAREN, Kent Ridge Digital Labs, Singapore 119613

Jason Lowder (663) School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

Drakoulis Martakos (635) Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

Saverio Mascolo (693) Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 70125 Bari, Italy

Tong Mei (367) Gintic Institute of Manufacturing Technology, Singapore 638075

Troy Milner (663) School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

P. C. Pandey (417) Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

Sangkun Park (517) Institute of Advanced Machinery and Design, Seoul National University, Seoul 151-742, Korea

Samuel Pierre (289) Mobile Computing and Networking Research Laboratory (LARIM); and Department of Computer Engineering, École Polytechnique de Montréal, Montréal, Quebec, Canada H3C 3A7

Sameer Pradhan (663) Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Paolo Salvaneschi (751) ISMES, Via Pastrengo 9, 24068 Seriate BG, Italy

G. Sfikas[3] (859) Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

Timothy K. Shih (327) Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 25137, Republic of China

José A. Taboada (825) Department of Electronics and Computer Science, Universidade de Santiago de Compostela, 15782, Santiago de Compostela (A Coruña), Spain

R Tafazolli (859) Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

I-Fei Tsai (719) Wistron Corporation, Taipei 221, Taiwan

[3]Current address: Lucent Technologies, Optimus, Windmill Hill Business Park, Swindon, Wiltshire SN5 6PP, England

Irene M. Y. Woon (497) School of Computing, National University of Singapore, Singapore 117543

Xindong Wu (663) Department of Computer Science, University of Vermont, Burlington, Vermont 05405

Du Zhaohui (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

FOREWORD

Database and Data Communication Network Systems: Techniques and Applications is a significant, extremely timely contribution to the emerging field of networked databases. Edited by Cornelius T. Leondes, a leading author in the areas of system analysis and design, this trilogy addresses three key topics: (1) database query, organization, and maintenance; (2) advanced database applications, and; (3) data communications and network architectures. This landmark work features 25 authoritative and up-to-date expositions from world-renowned experts in industry, government, and academia.

The two most valuable features of this work are the breadth of material covered and the alignment of the many diverse topics toward a common theme—the integration of database and data communications technologies. This work provides an extremely valuable reference for researchers and practitioners interested in the analysis, design, and implementation of distributed, networked databases. Collectively, the 25 chapters will assist the reader in building the necessary background and in acquiring the most advanced tools to effectively engage in the evaluation of existing network and database systems and in the design/development of new ones. Volume I covers data processing techniques and includes 9 chapters that describe the architectural characteristics of a modern database system (including object-oriented structure, multilevel organization, data compression, and security aspects) as well as the most efficient methods to query a database (e.g., data mining, query optimization, fuzzy query processing, and geometric hashing). Volume II covers database application techniques and includes 7 chapters that describe challenging

applications that rely on advanced database concepts and technologies. In these chapters, database practitioners describe their experience in adapting and often extending state-of-the-art technology to support several demanding applications. The applications range from scientific (data visualization, virtual environments) to manufacturing and engineering support (training, product demonstration, prototyping, manufacturing, resource planning), and corporate (finance, management information systems). Volume III covers data communications networks and closes the loop, introducing the third key component (next to database technology and applications) in modern database design, namely, the network. Of the 9 chapters in this volume, some cover the key design issues in a data communications network (ATM congestion control; ATM connectionless service optimization; fault tolerant network design; heuristic tools for network topology optimization; wireless ATM). Other chapters address the efficient use of the web (web access during heavy traffic periods; web information exploration). The balance of the chapters brings networks, databases, and applications together in vertically integrated system designs (integration of database, network, and AI to support monitoring and safety applications; networks and databases for hospital intensive care units; high-speed, multimedia application support over ATM). Collectively, this material represents a tremendous resource of methodologies, examples, and references to practitioners engaged in the development of networked database systems for advanced applications.

The publication of this work could have not been better timed. Let us take a look back in history to study the evolution and interplay of data communications, computer processors, and databases. A little over 30 years ago the ARPANet was born. The packet switching technology introduced by the ARPANet made it possible to integrate computers and communications in a common fabric, where the cycles and software of remote machines could be shared and parallelized to achieve results unthinkable on a single machine. Since then, the concept of opportunistic sharing of computers across the Internet has come a long way, as witnessed by the all powerful "computational grid," where intelligent middleware steers real time cycle hungry applications to idle computer resources. Ten years ago, the web revolution propelled a new key player to the stage—the data files and the data information bases stored across the network. It immediately became clear that databases and network will grow hand in hand to form a distributed information fabric. In this fabric, data will be stored in a hierarchical fashion, from the tiniest sensors monitoring a patient's heartbeat or the traffic on a freeway to the largest servers storing the economic parameters of the universe and the models to forecast the future of the world economy. This massively distributed database environment will require radically new technologies for efficient data organization, data query, security, and load balancing.

The trilogy by Cornelius T. Leondes addresses precisely the issue of synergy between databases and data networks. It provides the ideal foundation and set of tools for the professional that wants to venture into the field of networked databases. Across the chapters, the reader will find a wealth of information and advice on how to select the database organization, the query strategy, and

the network architecture and how to best match the architecture choices to the target application.

Mario Gerla
Computer Science Department
University of California, Los Angeles

This Page Intentionally Left Blank

PREFACE

The use of databases and data communication networks (the Internet, LAN/Local Area Networks, and WAN/Wide Area Networks) is expanding almost literally at an exponential rate and in all areas of human endeavor. This illustrates their increasing importance and, therefore, the strong need for a rather comprehensive treatment of this broad area with a unique and well-integrated set of volumes by leading contributors from around the world, including 12 countries in addition to the United States.

It is worth noting that this subject is entirely too broad to be treated adequately in a single volume. Hence, the focus of **Database and Data Communication Network Systems: Techniques and Applications** is broken into three areas: database processing techniques are covered in Volume 1, database application techniques are covered in Volume 2, and data communication networks are covered in Volume 3.

The result is that each respective contribution is a remarkably comprehensive and self-contained treatment of major areas of significance that collectively provides a well-rounded treatment of the topics in these volumes. The authors are all to be highly commended for their splendid contributions to this three-volume set, which will provide an important and uniquely comprehensive reference source for students, research workers, practitioners, computer scientists, and others for years to come.

C. T. Leondes

This Page Intentionally Left Blank

1 EMERGING DATABASE SYSTEM ARCHITECTURES

TIMON C. DU

Department of Industrial Engineering, Chung Yuan Christian University, Chung Li, Taiwan 32023; and
Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Shatin, NT Hong Kong

A database is a repository that collects related data, and is different from the traditional file approach, which is only responsible for data maintenance. Instead, a database should have the characteristics of maintaining persistent data, preserving a self-describing nature, controlling insulation between programs

Database and Data Communication Network Systems, Vol. 1

and data, supporting multiple views of data, and sharing data among multitransactions. From the 1970s to 1990s, the relational database model has replaced the hierarchical database model and network database model in most application areas. Section II briefly describes the database system in historical perspective, and Section III presents the relational data model. Then other database models such as the deductive data model, object-oriented data model, distributed data model, active database, and other databases are discussed in Section IV. Moreover, Section V shows the hybrid system, e.g., deductive and object-oriented database (DOOD) and the joint of active databases and object-oriented databases, which integrates the advantages of individual database models. For example, researchers in DOOD have attempted to combine the merits of the deductive database and the object-oriented database. The future study of database technology will concentrate on design perspectives, knowledge exploration, and system interfaces. These topics are covered in Section VI. Section VII introduces several important database applications, such as data warehousing, on-line analytic processing, decision-support system, and engineering and production applications. Finally, Section VIII presents a summary.

I. INTRODUCTION

The database system is a system to support compact, speedy, ease-of-use, and concurrent databases. Therefore, a database system includes hardware, software, data, and users. A database management system (DBMS) is software that interacts with the operating system and is responsible for creating, operating, and maintaining the data. A DBMS has several responsibilities [19]:

(a) *Redundancy control.* Duplicated data require data to be stored many times, and thus there are problems of duplication of effort, waste storage, and more importantly, data inconsistency.

(b) *Restriction of unauthorized access.* Some data may only be permitted to be retrieved, or updated by specific users. A DBMS should provide a database security and authorization subsystem to maintain data in either (1) a discretionary security mechanism, granting privileges to users, or (2) a mandatory security mechanism, enforcing multilevel security toward various security classes.

(c) *Database inference.* New information may be needed to be deduced from the data. The deductive database, data mining, and other technologies, explained later, can produce new information from existing data.

(d) *Representation of complex relationships among data.* The relationships between data reveal a lot of information. To well represent relationships in a database model for efficient usage is an important duty of a DBMS. An entity-relationship (ER) model is normally adopted for data modeling.

(e) *Retrieveval and update of related data easily and efficiently.* Since the data are maintained for future usage, the file organization, such as indexing structure, is critical for retrieval and updating easily and efficiently.

(f) *Enforcement of integrity constraints.* The DBMS provides the capabilities to define and maintain the data integrity. The DBMS evaluates the data

states and triggers actions if the constraint is violated. In general, the data are maintained in the ways of *must be* and *must not be*. Defining the data integrity means that the data *must be* in certain states. For example, domain constraints specify the value of attributes *must be* the domain values while the key constraints assert each table in the relational database *must have* the primary key and entity integrity constraints affirm the key value cannot be null. Similarly, the referential integrity constraint maintains the data consistency between tables. On the other hand, the DBMS can make sure the data *must not be* in specified states. For example, the basic salary *must not be* lower than a certain amount and the employees' salary *must not be* increased higher than their immediate supervisor. Most of these types of constraints are considered as business rules and are preserved by the semantic constraints.

(g) *Concurrency control.* Concurrency control is a mechanism to allow data to be accessed by many users concurrently so that the lost update problem, the dirty read problem, and the incorrect summary problem will not happen [19]. Most DBMS use a locking mechanism, timestamp protocol, multiversion control, optimistic protocol, or any combination of them to achieve concurrent data access. The higher degree of concurrency, the more heavier the operations are on the DBMS.

(h) *Backup and recovery.* Data transaction may fail because of system crash, transmission errors, local errors, concurrency control enforcement, or catastrophes. The recovery process is a mechanism for recovering the committed data without losing any data. The recovery process can be done by DBMS automatically or through restoring from backup copies by users.

In order to meet these responsibilities, a DBMS has several components to define (specify the data types, structures, and constraints), construct (store the data itself on some storage media), and manipulate (query, update, generate report) the data. The components include data definition language (DDL), data manipulation language (DML), data security and integrity subsystem, data recovery and concurrency mechanism, query optimization algorithm and performance monitoring functions.

It is an important job of a DBMS to respond to user requests. A query language is a normal tool for processing a request. The ad hoc query languages are a structured query language (SQL)-like language. SQL is a kind of declarative language, which specifies what the user wants instead of how the job is done. The language is easy to use, and is composed of several key words, i.e., select, from where, group by, having, order by, insert, update, delete, that most relational DBMS have adopted the query standard. Corresponding to the SQL language, relational algebra and relational calculus are also well-known query languages. Any query can be written in either relational algebra, relational calculus, or SQL, and can be transformed from one to the other. The combination of these three methods provides a strong tool set for theoretical analysis or computing implementations. Furthermore, both relational algebra and relational calculus support the query optimization and SQL is easy to comprehend.

As an example, find which options are unavailable on a 2002 Ford Explorer using the database represented by the ER model. If the relational schema

(tables) is

option (*CODE, DESCRIPTION, PRICE*)
avail_opt (*MAKE, MODEL, YEAR, OPTCODE*, STD_OPT),

The relational algebra is written as

$$option_{(CODE,\ DESCRIPTION,\ PRICE)} - \pi_{(CODE,\ DESCRIPTION,\ PRICE)} (\sigma_{MAKE='Ford'\ MODEL='Explorer'\ YEAR='2002'} (avail\_opt_{OPTCODE=CODE} option)),$$

where two relational tables, *option* and *avail_opt*, are joined (∞) based on attribute $OPTCODE = CODE$, then select (σ) using the conditions of $Make = Ford$, $Model = Explorer$, and $Year = 2002$, then projected (π) to leave only three attributes *CODE*, *DESCRIPTION*, and *PRICE*. Finally, values for the resulting tuples are returned from the *option* table.

Similarly, the SQL is written as

```
select  CODE, DESCRIPTION, PRICE
from    option
where   not exists
    (   select  *
        from    avail_opt
        where   CODE = OPTCODE and
                MAKE = 'FORD'
                MODEL = 'Explorer'
                YEAR = '2002');
```

The tuple relational calculus is

$$O \mid option(O) \text{ and not } (exist\ A)(avail\_opt(A) \text{ and } A.\text{OPTCODE} = O.CODE \text{ and } A.MAKE = \text{'Ford' and } A.MODEL = \text{'Explorer' and } A.year = \text{'2002'}$$

There are several kinds of users using a DBMS: database administrators, database designers, and end users. A database administrator oversees and manages the database and secondary storage resources (hard disk, tapes, etc.), authorizes access to the database, coordinates and monitors database usage, and acquires software and hardware resources. He/she is the key person responsible for the success of implementing a DBMS. The database designer identifies the data to be stored and chooses appropriate structures to represent data. He/she can be the system analyst who communicates with end users and interacts with application programmers to develop a database system. A database system is built to support the needs of different kinds of end users, such as causal end users, naïve end users, and sophisticated end users. Causal end users normally are middle-or high-level managers who access the database for querying, updating, and generating reports occasionally. Therefore, an appropriate interface with higher flexibility is required for the causal users. Naïve end users, e.g., bank tellers and reservation clerks, use standard types of queries and updates.

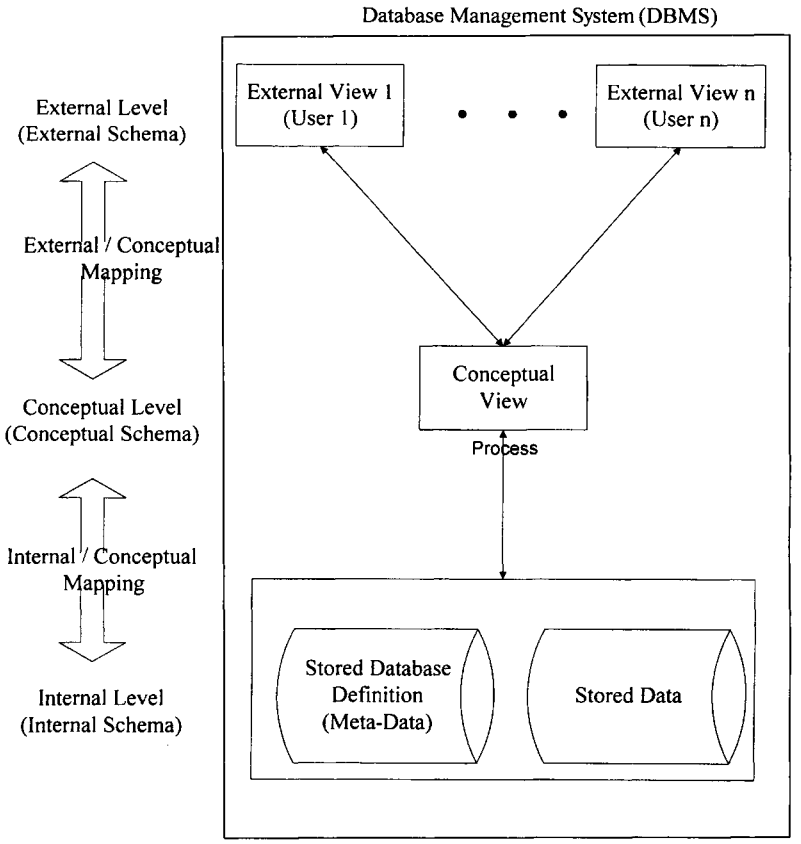


FIGURE 1 The three-schema architecture and two-mapping independence.

A designated interface with canned query transactions is needed. Sophisticated end users, e.g., engineers, scientists, and business analysts, know the database system very well. They use tools to meet their complex requirements.

As has been stated, a DBMS has several functions. However, the most important concept of building a database is the three-schema architecture and two mapping independents. As shown in Fig. 1, the three schemas are external schema, internal schema, and conceptual schema. The internal schema describes the structure of the physical data storage and the access path while the external schema portrays the database from the user's point of view and hides the detailed information from users. The conceptual schema describes the structure of the whole database just like the entity types, relationship types, and constraints in the ER model. Two mapping independents, logical data independence and physical data independence, make the database design and implementation possible. Logical data independence means any change of the conceptual schema will not affect the external views of users, and physical data independence means any change of the internal schema will not change the conceptual schema. That is, the reorganization of the physical structure and modification of the conceptual

design will not affect the database user's perspective. This feature allows the database to be improved.

II. HISTORY

The first computer was invented in 1942 by Dr. John V. Atanasoff, a professor of University of Iowa, and his graduate student Clifford E. Berry for solving physics problems [24]. The computer was named ABC, which stands for "Atanasoff Berry Computer." The computer used an electronic medium and vacuum tubes to operate binary computation. However, the era of the first generation of computers is considered from 1951 to 1959, characterized by the computers using vacuum tubes. The IBM 701 was the first commercialized computer. The second generation of computers was triggered by the invention of transistors in 1959. Computers with transistors were compact, more reliable, and cheaper than the vacuum tube computers. Integrated circuits, the most important invention in the history of computers, created the era of the third generation of computers in 1964. In 1971 more circuits were able to be confined in a unit of space, called very large scale of integrated circuits, VLSI, and the age of the fourth generation of computers was declared. However, the commonness of information technologies is due to the announcement of IBM personal computers, PCs, in 1981. From then on, the use of computers grew and the requests for data storage prevailed.

Corresponding to the movement of computer technology, the trend of information system can also be divided into five generations [19,28] (see also Fig. 2). In the 1950s, the information system was responsible for simple transaction

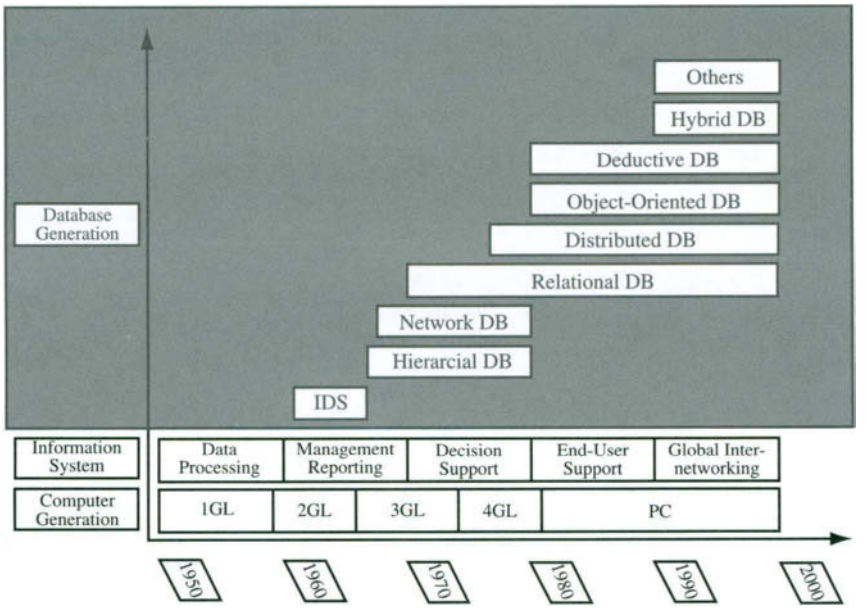


FIGURE 2 The generation of computers, information systems, and database models.

processing and record-keeping. At that time, database technologies were too far ahead to think about. In the 1960s, the concept of a management information system (MIS) was initiated. During that time, the information system provided the data and information for management purposes. The first generation database system, called Integrated Data Store (IDS), was designed by Bachman of GE. Two famous database models, the network data model and the hierarchical data model, were also proposed during this period of time. The network data model was developed by the Conference on Data Systems Languages Database Task Group (CODASYL DBTG) and the basis of the hierarchical data model was formed by joint development groups of IBM and North American Aviation. Note that the innovation of the hierarchical data model is earlier than both the network data model and the relational data model [25]. The implementation of information systems grew rapidly in various areas in the 1970s. Therefore, information technologies were proposed for specific implementation topics. For example, the decision support system (DSS) provides decision-making style information for managers when he/she confronts problems in the real world. Related to the database technologies in this period of time, the relational data model proposed by Ted Codd and the ER model introduced by Chen are the most significant achievements. These technologies have prevailed for decades. The use of SQL, a declared data definition language and data manipulation language, is another push for the database population growth. Several famous relational database management systems were built from academic research projects, such as INGRES from University of California, Berkeley, Austin's System 2000 from University of Texas, Austin, and computer companies, such as System R from IBM. In 1980, the popularity of personal computers awakened the need for end-user computing. For example, executive information systems (EIS), expert systems, strategic information systems, and the personal DBMS i.e. DBASE, PARADOX, etc., were developed. The main stream of database technologies moved from hieararchical and network database to the relational database. Several prominent commercial DBMS are DB2, ORACLE, SYBASE, and INFORMIX. On the other hand, the computer network technology and the artificial intelligent technology pushed the database system toward a new perspective of implementation. That is, the computer network technology enhanced the relational database into the distributed database, and the artificial intelligent technology integrated the database system into the deductive database. Some famous prototypes of the deductive databases are NAIL! (Not Another Implementation of Logic) by Standard University in 1985, the CORAL system by University of Wisconsin at Madison, the LDL (Logic Data Language) project by Microelectronics and Computer Technology Corporation (MCC) in 1984, and the Datalog language, a subset of Prolog language. The distributed database distributes data over multiple sites and communicates with one another via computer networks. Thus, there are several advantages to the distributed database, such as local autonomy, improved performance, improved reliability and availability, economics, expandability, and shareability [31]. There are numbers of experimental distributed DBMS, such as distributed INGRES by Epstein *et al.* in 1978, DDTS by Devor and Weeldreyer in 1980, SDD-1 by Rothnie *et al.* in 1980, System R* by Lindsay *et al.* in 1984, SIRIUS-DELTA by Ferrier and Stangret in 1982, MULTIBASE by Smith *et al.* in

1981, and OMNIBASE system by Rusinkiewicz *et al.* in 1988 [19]. However, since the developing time of the distributed database technology is matched to the maturity of the relational database, most commercial function-complete relational DBMS offer some degree of database distribution capabilities. The example can be found in ORACLE, SYBASE, and SQL Server. It is worth noting that the client/server architecture can also be considered as a kind of distributed database with limited distributed functions.

The object-oriented database was another important landmark in the 1980s since the need for data-intensive applications, such as the Office Information System (OIS), Computer-Aided Design and Manufacturing (CAD/CAM), Computer Aided Software Engineering (CASE), and Computer-Integrated Manufacturing (CIM). The object-oriented database integrated the technologies of object-oriented concepts and database concepts. There were several prototypes of the object-oriented database: O2, GemStone, ONTOS, ORION, Iris, POSTGRES, Versant, and ObjectStore [18].

In the 1990s, the main trend of database technologies is building hybrid DBMS since the individual database tools have matured. For example, the importance of the object-oriented database increased in the 1990s and were even admired as the next generation database system. Since that time some researcher tried to integrate the deductive database and the object-oriented database for obtaining the advantages of both systems. Therefore, a new system, called the deductive and object-oriented database (DOOD), has been proposed. Examples can also be found on integrating the distributed system and object-oriented database into the distributed object-oriented database, and integrating the distributed system and the knowledge base system into the distributed knowledge bases [31]. Another trend of information technologies is that the database is expected to play more important roles and store more complicated data types. For example, the active database tries to monitor environmental conditions than just be a passive data storage. The multimedia database stores complicated datatypes, such as documented, musical, pictorial, and knowledge-related data. Others database systems, including scientific, statistical, spatial, temporal, etc., will be discussed in the later sections.

III. RELATIONAL DATA MODEL

As discussed under History, Ted Codd, an IBM research fellow, proposed the relational data model in 1970. Not surprisingly, the relational data model soon became the most successful database architecture because of its simple and uniform data structure. A relational database represents data in tables, called relations. A table means it is an entity class, which is composed of a set of homogeneous instances, such as students class, courses class, instructors class, and department class. In each table the columns stand for attributes, defined in domains, of these entities and the rows, also called tuples, are the instance set.

In the relational data model, the data are organized in the fashion of tables, and the data consistency is secured by integrity constraints. Constraints

can be categorized into *class constraints* and *operational constraints*. The class constraints include ISA class constraints, disjoint class constraints, property induction constraints, required property constraints, single-valued property constraints, and unique property constraints. In the list, both the key constraints and referential integrity are the kernel components for the relational database. The key constraints assure that all instances of a table are distinct and the referential integrity constraints maintain consistency among instances of two tables. For example, if a student in a student class intends to take a course, we must assure that this course is actually offered in a course class. A DBMS should maintain the data consistency when data insertion, deletion, and modification are requested.

To successfully design a relational database, a designer could start from two approaches: relational normalization or semantic data modeling. The relational normalization process allocates facts based on the dependencies among attributes. In a well-designed database, two kinds of dependencies exist among attributes: functional dependency and multivalued dependency. The attribute A is functionally dependent on attribute B if the value of attribute A is dependent on the value of attribute B. Otherwise, the attribute A is multivalued dependent on attribute B.

Both the ER model and the extended-ER (EER) model are semantic data modeling tools. This approach is called the synthesis process. The semantic data modeling technique starts from small tables. In the graphical representation, the designer identifies the primary key and foreign key of each table, and uses integrity constraints, e.g., entity integrity constraints and referential integrity constraints, to maintain the consistency and relationships. The entity integrity constraints require that no null is allowed in the primary key, while the referential integrity constraints state the relationships in tables and tuples. The relationship between tables is represented by the relationship entity and cardinality ratio. This ratio is a constraint that specifies the number of participating entities. It is recommended that the designer first uses the relational normalization process to generate the elementary tables and then revises them using a semantic data model. At the beginning, the designer may be confronted with a large source table that includes many entities with several kinds of attributes that relate to different candidate keys. Structured guidelines were developed to decompose (normalize) a source table into smaller tables so that update anomalies are avoided. Loomis [25] listed the following five steps of normalization (each step results in a level of normal form):

1. A table can only have one single value in the same attribute and the same tuple.
2. Every nonkey attribute has to be fully functionally dependent on the key attribute or attributes.
3. A table cannot have any nonkey attributes that are functionally dependent on other nonkey attributes.
4. A table cannot have more than one multivalued dependency.
5. A table cannot be decomposed into smaller tables then rejoined without losing facts and meaning.

IV. NEXT GENERATION DATA MODEL

From the 1970s to 1990s, the relational database model replaced the hierarchical database model and network database model in most application areas. However, the constraints on simple and uniform data types and normalization requirements of the relational database model limited the implementation scopes, such as the operations on version maintenance, long transaction time, quick and complex data retrieval, evolutionary nature, and knowledge generation. Therefore, other database models have emerged to fulfill these needs. The examples below include the deductive data model, object-oriented data model, distributed data model, active database, and other databases, which will be discussed in the later sections.

A. Deductive Data Model

A deductive database uses facts and rules, which are concepts from the artificially intelligent arena. Unlike other database systems, the deductive database defines what a database process wants to achieve rather than specifying the details of how to achieve it.

The facts are a collection of ground predicates in the first-order logic manner. For example, *Mother (John, Mary)* can be used to describe that John's mother is Mary. Rules are the core of a deductive database. They describe the relationships among facts. For example,

if x is y's mother, then x is y's parent.

New facts are generated by interpreting the rules through the inference engine. This interpretation can be by way of either the bottom-up or top-down inference mechanism.

The bottom-up inference mechanism (forward chaining) starts by checking the facts and then applies the rules to generate the new facts to achieve the query goal. The top-down mechanism (backward chaining) starts with the query goal by matching one condition after another until valid facts are obtained. *Prolog* is the most famous artificial programming language that utilizes backward chaining [19]. The drawback of this inference mechanism is that the order of facts and rules can significantly affect the query results. An example will illustrate the point.

Consider the query: *James is whose ancestor?*, written as: *?Ancestor (James, Y)*, the database in which contains these facts

Parent(Franklin, John),
Parent(Franklin, Tom),
Parent(Franklin, Joyce),
Parent(Jennifer, Alicia),
Parent(Jennifer, Mary),
Parent(James, Franklin), and
Parent(James, Jennifer),

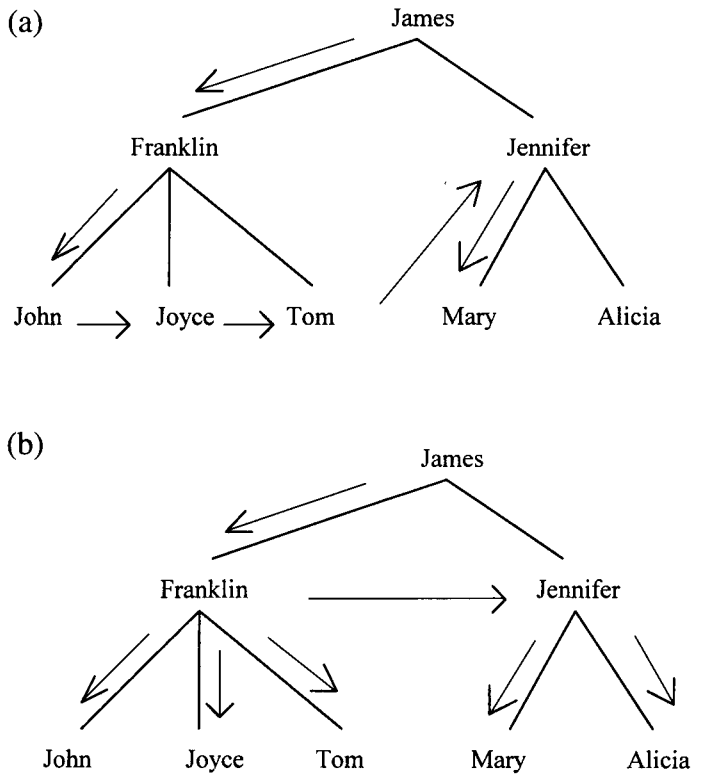


FIGURE 3 Backward and forward inference mechanism search paths [17]. (a) The search path of backward chaining inference and (b) the search path of forward chaining inference.

and these rules

rule 1: Ancestor(X, Y):- Parent(X, Y), and
rule 2: Ancestor(X, Y):- Parent(X, Z), Ancestor(Z, Y).

There are no facts of Ancestor(James, Y). However, using the rules, both the backward and forward inference mechanism generates new facts, such as Ancestor(James, Franklin). This permits the query to be answered. They are: Y = Franklin and Y = Jennifer. The backward chaining inference will stop after finding one answer, e.g., Franklin, and will find other answers only when requested. Therefore, the sequence of rules and facts is important. Figure 3 shows the backward and forward chaining paths.

When the inference mechanism generates new facts by applying the individual rules, higher level of facts may be generated several times. Therefore, it is important to have a mechanism for duplication elimination. Consequently, this is also an important issue in the deductive and object-oriented database architecture.

A deductive database would use a forward chaining inference mechanism, such as *Datalog*, rather than a backward chaining inference mechanism such

as *Prolog*. Datalog is a bottom-up evaluation mechanism in which the order is not important. Datalog requires that rules be written as a Horn Clause. Such a clause can have at most one positive atom formula (literal). For example, a rule can be written as, if P_1 and $P_2 \ldots$ and P_n are true, then Q is true (it can be written formally as Q:- P_1, $P_2, \ldots, P_n$) where Q is the only positive literal. If there is no positive literal, the rule is used for integrity control, referential integrity, and entity integrity, such as P_1, $P_2, \ldots, P_n$.

Another important feature of a deductive database is that only facts and rules are physically stored in the database. However, the new facts for query process will not be stored in the database permanently. In other words, after finishing the execution of a query, the new facts will be eliminated from the database. This feature can minimize memory requirements, especially when recursive rules are used and the same recursive predicate occurs in the antecedent and consequence. An example can be written as

```
unavail_Ford_Explorer (CODE, DESCRIPTION,PRICE)
  :- option (CODE,DESCRIPTION,PRICE),
  not (avail_opt('Ford', 'Explorer', '2002', CODE,--)).
```

Deductive database research projects started in the 1970s. Since then, several prototypes have been developed, such as the LDL prototype from MCC in Austin, NAIL! from Stanford, the Aditi project at University of Melbourne, the CORAL project at University of Wisconsin, the LOLA project at Technical University Muenchen, and the Starburst project at IBM [33]. However, it should be noted that few of these prototypes have been implemented in the industry.

B. Object-Oriented Database

Recently, as a result of hardware improvement, more application designs and implementations of data-intensive applications have occurred. Many of these implementations require complex data structures, which bring out the following issues: (1) object version maintenance, (2) long transaction time operations, (3) quick and complex object retrieval, (4) communication protocols, (5) evolutionary nature, (6) primitive and component manipulation, (7) protection mechanisms, and (8) rules and integrity constraints [8].

Since the object-oriented database model has been praised as the next generation database model, this section will present it thoroughly [18]. An object-oriented database management system integrates concepts of object-oriented and database technologies. The core concepts of an object-oriented technology include identity of objects, complex objects, encapsulation, classes, and inheritance. On the other hand, integrity constraints, data organization (update, read, deletion, etc.), schema design, rules, queries, and associations are important in designing databases. However, some of these concepts are not congruent. Therefore, building an object-oriented database management system requires making compromises. This section examines the important issues in designing object-oriented database prototypes from three perspectives: object-oriented technologies, database technologies, and application environments. The discussion from these three perspectives will explain object-oriented database fundamentals. In the object technologies, the focus will be on issues

related to object-oriented databases, which include (1) class methods and attributes, (2) multiple inheritance, (3) exceptional instances, (4) class extent, (5) versions, and (6) object identifier. The data technology topics are (1) composite objects, (2) association, (3) persistence, (4) deletion of instance, (5) integrity constraints, and (6) object migration. Many of these issues are also relevant to other database management architectures, such as relational, deductive, and active databases. Application environment topics discussed include (1) application program interface, (2) transactions, (3) object sharing/concurrency, (4) backup, recovery, and logging, (5) adhoc queries, and (6) constraints and triggers.

1. The Object-Oriented Technologies

Object-oriented programming is an accepted approach for large scale system development. The important concepts include class, object, encapsulation, inheritance, and polymorphism. An object can be any kind of conceptual entity. Object-oriented technologies build on the concept of classes.

A class represents a group of objects with conceptual similarities. For building a solid class structure, a class maintains private memory areas (called *attributes*), and the memory is accessed only through some designated channels (the channel is called *method*). This mechanism is called encapsulation. The outsider uses *messages* to communicate through methods to access the private memory. A class can inherit properties from other classes (called *class inheritance*), thereby forming a class hierarchical structure. A subclass can extend the information from a superclass (called *class exception*). By means of the class inheritance, unnecessary specifications and storages can be avoided. However, a class can only inherit properties from its immediate superclasses. Fortunately, more than one superclass is allowed (called *multiple inheritance*).

The same name can be assigned in the class hierarchy. If the contents of the class are different, the specification in the subclass can override the one in the superclass (called *overloading* or *overriding*). If two attributes with the same names but different contents appear in the superclasses in multiple inheritance, the name conflict can be resolved by assigning a higher priority to the default superclass or the first superclass.

In object-oriented technologies, both classes and their instances are treated as objects. Although the object-oriented technology appears promising, Banerjee *et al.* [5] asserts that lack of consensus on the object-oriented model is the most important problem to be resolved in object-oriented technologies, a fact born out by the different interpretations appearing in different object-oriented prototypes.

In designing an object-oriented database, there are several important issues that should be addressed:

1. *Class methods and attributes.* A class consists of methods and attributes. The method is the mechanism to manipulate stored data in an object. It includes two parts, *signature* and *body*. The signature is encapsulated in a class and describes the name of the method, the name of entities to be manipulated, and the return entities. The body consists of a sequence of instructions for implementing the method.

In an object-oriented design, the concept of encapsulation forbids the attributes in a class to be accessed from the outside without using the assigned methods. However, the database management system does allow values and definitions of attributes to be read and written in order to increase the implementation efficiency. A trade-off between these two conflicting goals must be made in the design phase.

Several recommendations have been proposed for resolving this conflict:

(1) Provide a "system-defined" method for reading and writing the attributes.
(2) Allow users to decide which attributes and methods can be modified from outside.
(3) Allow all attributes and methods to be accessed from the outside, but use the authorization mechanisms to limit the access level.

2. *Multiple inheritance.* The inheritance property supports object reusability. This is done by inheriting attributes, methods, and messages from superclasses. Furthermore, a subclass can have its own attributes, methods, and messages. The multiple inheritance focuses on whether a subclass can inherit properties from more than one superclass. If the system allows multiple inheritance, several conflicts may happen:

(1) The subclass has the same names for attributes and methods within superclasses that are within different domains. In this case, the solution is either to warn the user or simply not to inherit the functions. The overloading property takes advantage of this conflict by assigning the same function names to different domains.
(2) More than two superclasses have the same function name but with different domains. In this case, the system needs to maintain rules for resolving the conflict.
(3) An inheritance structure might have cyclic inheritance relationships. Therefore, the system needs to detect the cycle automatically. If the cycle exists when the new inheritance is assigned, the inheritance assignment should be forbidden.

3. *Exceptional instances.* Objects are generated through instantiation from classes and prototype objects. When instantiation from classes occurs, the class is used as a template to generate objects with the same structure and behaviors. When instantiation from other prototype objects occurs, new objects are generated from the old objects through modification of their existing attributes and methods. Modifying the attributes and methods of an object during the generation process is called *exceptional instances*. The features of late-binding and overloading are accomplished through exceptional instantiation.

4. *Class extent.* The object-oriented systems interpret classes and types in several different ways. Generally speaking, the purpose for differentiating types and classes is to avoid assigning object identifiers (OIDs) to some kinds of objects, i.e., types. A type represents a collection of the same characteristics (integers, characters, etc.) of a set of objects, while a class is its implementation. From this point of view, a class encapsulates attributes and methods, and its instances are objects while the instances of types are values.

5. *Versions.* Multiple-version is a necessary property in some application areas, such as the design version in CAD, but not all prototypes allow more

than one version to exist. When multiple-versioned objects are allowed, systems must resolve referential integrity problems.

6. *Object identifier.* The OID is used to identify objects. There should not be any two objects with the same OID. A system assigns OIDs according to how the system manages objects and secondary storage. Two approaches are used in assigning OIDs:

(1) The system assigns the logical OIDs (surrogate OIDs) and uses a table to map the OIDs to the physical locations, or

(2) The system assigns persistent OIDs; the OIDs will point directly to their physical locations.

2. Database Technologies

A database management system is a tool for organizing real world data into a miniworld database. To create a good database management system, several important issues need to be considered:

1. *Composite objects.* A composite object in an object-oriented database is equivalent to an aggregation relationship in a relational database. Several component objects can be grouped into a logical entity, a composite object. As discussed by Bertino and Martino [8], locking, authorization, and physical clustering are achieved efficiently through this process. The relationship between a composite object and its component objects is described by the composite reference. The component objects are called dependent objects. The structure of dependent objects is used to maintain the integrity constraints. Thus, the existence of a dependent object depends on the existence of its composite object.

Composite references can be implemented in two ways: shared and exclusive. A shared composite reference means that a component object can belong to more than one composite object, while an exclusive composite reference can only belong to one. The composite reference can also be distinguished as dependent and independent. This aspect is used to analyze the existence between the composite object and the component object. It is called a dependent composite reference if the component objects cannot exist when the corresponding composite object is deleted. However, in the independent composite reference, the component objects can exist no matter whether the composite object exists.

2. *Associations.* An association is a link, i.e., a relationship, between related objects. For example, a manager is associated with employees in his research department through the project relationship. Therefore, the project number can be an attribute of manager entities that link with employee entities. The number of entities involved in an association is indicated by a degree value. The minimum and maximum number of associations in which an entity can participate is defined in the cardiality ratio.

The function of association is not explicitly implemented by most object-oriented prototypes. One approach is to use a separate attribute to indicate each participating entity. For example, if an *order* entity is associated with three other entities, *customer*, *supplier*, and *product*, then three attributes, say *to_customer*, *to_supplier*, and *to_product*, can be added to indicate the associations.

Furthermore, for more efficient reverse traversal, adding reverse reference attributes is necessary. The reverse reference attribute is an additional attribute in the associated entity and points back to the original entity. The consistency of such an attribute can be controlled by users or the system.

3. *Persistence.* The persistent design issue deals with whether the objects should permanently and automatically exist in the database. There are three kinds of approaches to deal with the existence of instances:

(1) Persistence is automatically implied when an instance is created.
(2) All the instances created during the execution of a program will be deleted at the end of execution. If the user wants to retain the instances in the database, he or she needs to insert the instance into the collection of objects.
(3) Classes are categorized into temporary objects and permanent objects. The instances created through the permanent classes are permanent instances, while those created through temporary classes are temporary objects.

4. *Deletion of instance.* The issue of deletion of an instance is related to persistence. There are two approaches to deleting instances:

(1) Provide a system-defined deletion function. When an instance is deleted, the reference integrity constraint is violated. The system can totally ignore the violation or maintain a reference count. The reference count records the number of references connected to other objects. An object can be deleted when the count is zero.
(2) An object will be removed automatically when no references are associated with it.

5. *Integrity constraints.* Integrity constraints are used to maintain the consistency and correctness of data by domain and referential integrity constraints. The domain constraints specify the legalization and range of data, while the referential integrity constraints control the existence of referenced entities.

Integrity constraints can be implemented as static or dynamic constraints. Whether the state values of objects are legal is controlled by the static constraints, and the state transactions of objects are controlled by the dynamic constraints. Integrity constraints in an object-oriented database prototype is defined imperatively in methods. This approach is different than a relational database. However, the declarative approach in the relational database seems to be superior to the object-oriented database approach.

6. *Object migration.* The migration of instances through classes is an important function in object-oriented databases. For example, when a student goes from undergraduate to graduate school, the student instance should be able to migrate from an undergraduate class to a graduate class without deleting the instance from one class and recreating it in another.

3. Application Environments

To successfully implement an object-oriented database on different platforms, consideration needs to be given to several issues:

1. *Transactions.* If a prototype is implemented in a client/server architecture, the maintenance of data transaction becomes important. Using a network

to transfer large data files, e.g., bipmap files, can result in a long transaction period, which can create unexpected problems such as system or network crashes during the transaction.

There are two kinds of transactions: short and long. The short transaction is designed for an atomic transaction, one that can either commit or rollback completely. The long transaction has similar behavior but is used for checking-out objects from a group database to a personal database with a persistent write or read lock on source database to decrease the network traffic. Therefore, a long transaction can continue for an extended period.

2. *Object sharing/concurrency.* Concurrency control is a critical issue in the multiuser environment. Some prototypes choose an optimistic approach, which means that no lock is requested when updating the data. However, whether the new data can overwrite the old data depends on successful commitment. The other, pessimistic approach requires all transactions to grant locks before doing any processing. This approach is more conservative but reliable.

The locking strategy is dependent upon the transaction type. As has been discussed before, there are two kinds of transactions. The short transaction has four kinds of lock: write, read, updating, and null. The write lock needs to be granted before updating the objects. If an object is locked by the write lock, it cannot be read or written by other transactions. The read lock is useful when the user only wants to read the data of objects. For preventing the data to be changed at the moment of reading, the write lock and updating lock are not allowed if an object has a read lock. The updating lock stands between the write lock and the read lock and is useful when updating is expected after the reading. Therefore, any other write and updating locks should be rejected when an object has an updating lock. The long transaction is designed for checking out objects from the group database to the personal database. Therefore, the updating lock is not appropriate in the long transaction. The write lock in the long transaction is used to lock the object in the source database to prevent other transactions from checking out the object. Since the data will be changed only when an object is checked back into the group database, the read lock allows the object to be checked out by other transactions. The null lock for both short and long transactions is only a snapshot of the object, not a real lock. When the null lock is used for checking out an object in the long transaction, the object is actually being copied from the group transaction to the short transaction with an assigned new OID. If this object is checked-in back to the source database, it will be treated as a new object.

3. *Backup, recovery, and logging.* These are important issues when transferring data in a network. Some prototypes provide transaction logs, but some do not.

Logging can be used either for backing up values or for recovering from a crash. For example, logging provides a place for temporarily storing values when the savepoint is chosen in a short transaction. Logging is even important for long transactions in the following situations:

(1) To maintain the status (e.g., locks, creation/deletion, or updating) of source copies and duplicated copies when objects are checked-out or checked-in between a personal database and a group database.

(2) To reconstruct a long transaction if the personal database crashes after objects have been checked-out.
(3) To reconstruct the locks and links of the source objects between the duplicated objects in the personal database if the group database recovers from the crash.

4. *Ad hoc queries.* The ad hoc query function provides a declarative query, e.g., SQL, for retrieving data. It is difficult to declare a query in an object-oriented database management system. Stonebreaker *et al.* [36] insisted that a third-generation database must provide an ad hoc queries capability. To query an object-oriented database, the user needs to navigate from one object to another through links. Also, to not violate the rule of encapsulation, the only way to access data is through designated methods in objects. This is one of the major criticisms of an object-oriented database.

5. *Constraints and triggers.* Constraints and triggers are not mandatory concepts in an object-oriented database. Gehani and Jagadish [20] identified the difference between triggers and constraints. Generally speaking, constraints are used to ensure the consistency of the states and can be implemented in two ways: (1) to serve as constraints to protect the integrity, and (2) to implement query functions. Triggers are the active alert systems for detecting integrity violations whenever the specified conditions are satisfied. Normally, constraints and triggers are declaratively stated in the relational database and deductive database.

C. Distributed Database

When the computer network technologies matured in the late 1980s, distributed database concepts prevailed. Research focused on placing the existing centralized databases into networks so that these databases could utilize advantages provided by the networks, such as local autonomy, improved performance, improved reliability and availability, economics, expandability, and sharability [31]. Through the networks the processing logic, functions, data, and controls could be distributed to local processing nodes. By interconnecting the networks, the local processing elements could achieve an assigned task cooperatively.

The design of a distributed database system can start from either a top-down or a bottom-up approach. The top-down approach fragments the global schema into local schema, while the bottom-up approach emphasizes the integration of local conceptual schema. The latter approach creates a multidatabase architecture that compromises the existing heterogeneous database architectures. In this approach, the different local conceptual schema are transformed by a translator modular for schema integration. Since the local schema are different, the data of a local processing node are unable to be backed-up from one processing node to another in the multidatabase architecture. Fortunately, most approaches focus on the top-down approach while using the concepts of a relational database for the design phase. This approach has been successfully commercialized.

The system designer must carefully select the process used to fragment a source table into small tables and replicate them in local processing nodes. The process used should be carefully combined with relational normalization. There

are two kinds of fragmentation schema: vertical and horizontal fragmentation. The vertical fragmentation process cuts the source table into several fragments (tables) with different attributes. For joining purposes, the key attributes should be included in all fragments. The horizontal fragmentation slices the tuples of a source table into smaller tables with the same schema. The tuples in different fragments should not be overlapped.

Bochmann [9] proposed the following set of criteria for classifying distributed databases even though the degree of distribution may vary greatly:

1. Degree of coupling of the processing nodes, e.g., strong or weak.
2. Interconnection structure of the nodes, e.g., point-to-point.
3. Independence of data components.
4. Synchronization between data components.

For increasing availability and reliability, some distributed databases maintain multiple copies in different sites. However, updating the data is a difficult job. For example, when a processor wants to update the data in one site, it needs to update all copies to maintain data consistency. The intercommunication among sites for granting the data locks and the permissions for data access increases the network loading. Related issues are the handling of updates when one site crashes at the moment of updating and finding the most current copy when recovering. Therefore, timestamping, locking, updating, recovering, and deadlock, weave a complex web for a distributed database. Also, the reliability of networks creates an obstacle in using a distributed database. Many questions, such as how many copies should be kept and where they should be kept, affect cost and performance. Too many replicates may decrease the system performance since the network communication among nodes for maintaining the data consistency will increase.

D. Active Database

The earliest active database research was associated with System R. Eswaran and Chamberlin described this work at the first International Very Large Database Conference in 1975 [38]. The concepts of triggers and assertions were added to System R [26]. Briefly speaking, an active system is a database system with intelligence. It allows users to specify the actions that will be executed automatically when the expected conditions are satisfied. These actions are defined by specifying *assertions (or constraints)*, *triggers*, or *rules*. These three kinds of actions are written in Horn Clause form but have different behaviors. For example, a constraint for the relationship of advisor and advisee can be written as

$$advisor(x,y) \rightarrow advisee(y,x).$$

In some research, the terminology of constraints, triggers, and rules is confusing. A distinction between constraints and triggers was presented by Gehani and Jagadish [20]:

1. Constraints are used to ensure the consistency of the states. Triggers do not concern the consistency of states. They are simply triggered whenever conditions are true.

2. The actions of triggers can be separated from system transaction while the actions of constraints are parts of system transactions.
3. Triggers are evaluated after the events have been completed while constraints are evaluated at the moment of events processing.
4. Constraints are applied to all objects, but different triggers may be activated by different objects.

Both constraints and triggers are mainly used for active maintenance of database consistency. Gehani and Jagadish also suggested three kind of triggers:

1. *Once-only triggers.* The once-only triggers are deactivated after the triggers are fired and the actions are completed. If they are going to be used later, they need to be reactivated. For example, if the inventory is lower than the reorder level, a trigger is fired to place a new order. Then this trigger needs to be deactivated. Otherwise, the trigger will place orders several times. After the order products are received, the trigger should be reactivated.
2. *Perpetual triggers.* Unlike the once-only triggers, the perpetual triggers will be activated whenever the conditions are satisfied.
3. *Timed triggers.* The timed triggers will be fired by system clock.

Constraints can also be divided into two kinds of categories: hard constraints and soft constraints. The hard constraints will be checked right after any part of the predicate is completed, while the soft constraints will defer the constraint checking until all the predicates are completed.

Rules are different from constraints and triggers. Rules are used for knowledge deduction; therefore, the rules are able to propagate more rules and knowledge (facts).

There is some disagreement as to whether the rules, constraints, and triggers should be embedded within the application codes or declared outside the application. From a coding point of view, it is easy to write the codes if they are embedded within the application. However, both deductive reasoning and query optimization cannot be achieved if the rules, constraints, and triggers are embedded, because the deductive mechanism cannot infer from program codes. Another advantage of using the declarative approach is the ease of explaining the behaviors [38].

Today, the most important project in active database management system research is HiPac, which was supported by the 1987 Defense Advances Research Project [26]. HiPac proposed the event–condition–action (ECA) model. This model has been used in several consecutive research projects. The basic concept is simple: after the event occurs the conditions will be evaluated; if the conditions are satisfied, the corresponding actions will be applied. However, the most significant contributions of this work were the definitions of the attributes of ECA.

There are five attributes: *event, condition, action, E–C coupling,* and *C–A coupling.* E–C and C–A coupling can be executed in immediate, separate, or deferred modes. Therefore, a transaction can be divided into several subtransactions. Dayton and Hsu [16] used this idea for long-running activities and multiple transactions. Beeri and Milo [6] proceeded further to allow a programmer to specify an execution interval between condition evaluations and

their corresponding actions. The interval can be written as

execution-interval: [start-event, end-event].

Therefore, the triggered action can be executed between the start time and end time. This design alleviates system loading and increases parallelism. Moreover, a structure for storing the information about triggered actions until they are executed is provided. The events can come from database operations, temporal events, or external notifications. The database operations include all operations on the database such as data definition, data manipulation, and transaction control. The temporal events are based on the system clock and can be absolute, relative, or periodic. Any sources other than internal hardware or software events can be considered as external notification.

E. Other Database Models

There are other databases also appraised as the next generation databases. The following briefly explains the concepts.

1. *Scientific database.* The science-used data have the characteristics of rarely discarded data, low update frequencies, and large volume. Most of the time, the scientific data need to be analyzed by statistic analysis, time-series analysis, or pattern search. That is, the conventional database cannot be used for science data because the sequence of data and their characteristics are unable to be represented.
2. *Spatial database.* The spatial database models spatial data such as engineering design, cartography, and geological data. Lacking the appropriate data model is a problem for using the spatial database. The spatial database also needs to be able to support the query on the spatial data. For example, the queries can be "the line that intersects a given line segment?" or "the nearest point to a given point?"
3. *Temporal database.* The temporal database stores the temporal data and supports the queries for time points, time intervals, or the relationships between time points, i.e., before, after, during, simultaneously.
4. *Multimedia database.* There are several areas expect the support of the multimedia database, such as documents management, education, advertising, traveling, and system control. The multimedia database stores data in the format of numeric data, text, bitmap, video, or audio. Some issues of designing the multimedia database needs to be addressed: (a) The conventional database focuses on the schema design rather than on the data contents since the data have a rigid structure and the meaning of data can refer to the schema. However, the importance of data contents is greater in the multimedia database than in the conventional database. The meaning of data content cannot rely on the schema but on the application system. Therefore, a linkage between applications and data is needed. (b) The conventional keyword search cannot be used in bitmap, video, or audio types of data. From this perspective, the design process, data model, index structure, query processing, and performance should have different approaches. These remain for further study.

V. HYBRID DATABASE TECHNOLOGIES

New database models solve the deficiency of the conventional database model. However, further studies have proposed that the hybrid system needs to integrate the advantages of individual database models. The following presents two types of approaches: DOOD and the joining of active and object-oriented databases.

A. Deductive and Object-Oriented Database (DOOD)

Combining the merits of both deductive and object-oriented database architecture is called a deductive and object-oriented database (DOOD). There are two approaches being proposed for the database architecture: (1) adding the inference capability and rules into an object-oriented database architecture to and (2) adding the object-oriented concepts into a deductive database architecture. However, Ullman [37] asserted that both object-oriented and deductive database philosophies cannot be combined because of object identity and declarative programming problems. These two issues are explained below.

1. *Object identity.* Zaniolo [40] tried to incorporate object identity into a deductive database. He presented a set of steps to transform a rule into a new form. To the original rule

grandpa(Child, Gpa) <-- mother(Child, Mother), father(Mother, Gpa),

he added the OID so that the rule is transformed into

grandpa(Child, Gpa) <-- int (I), -: mother (Child, Mother), -: father(Mother, Gpa).

By following the new rule, when an atom is generated it will be assigned a new ID. Unfortunately, there are three problems in this process:

1. One atom should have only one ID. The new model will generate more than one ID for the same atom in the recursive rule evaluation.
2. The atom cannot permanently exist in the deductive database. (It will lose the objective of deduction if the atom exists permanently.)
3. Since the atom (object) is identified by an ID, if one atom is generated with a different ID, it is treated as a different atom. In this case, the function of duplication delimitation will not work.

Hence, Ullman's first assertion is valid.

However, such rigid constraints of OID also created some difficulties for the object-oriented database architecture. For example, in order to process data efficiently, an object-oriented database distinguishes the objects into persistent objects and transient objects. The persistent objects are the objects that will be stored permanently while the transient objects will be deleted after processing. From this perspective, the transit objects will face the same kinds of problems as the deductive database did. Another example can be found in the version issue of concurrent engineering. Therefore, it is necessary to relax the severe object identity constraint.

One way is to distinguish between identification and equality. That is, two objects are said to be identical if all the values and OID are the same. If the values are the same but with different OIDs, the two objects are said to be equal. As discussed in the deductive database section, a major advantage of a deductive database is the recursive predicate. If the equality concept can be accepted (not necessarily identical), then the previous problems would no longer be an issue. Actually, the recursive function can help an object-oriented database generate structured complex objects.

A query in both deductive and relational models uses declaration with high level abstraction. The optimization process and semantics in both models are clear and lead to relatively simple strategies for producing optimal results as the relational algebra does. The object-oriented model, however, does not use declaration. Manipulation and query of data are defined in the methods of a class. Even though the query in the object-oriented database can be written in SQL mode, it is still necessary to specify which method is going to be used. Furthermore, complex objects in an object-oriented database include structured and unstructured complex objects. A structured complex object is defined by specifying the data types, and the type of inheritance operates as usual. An unstructured complex object is for binary large objects (e.g., a bitmap file for graphical representation). However, since the data and its implementation software are stored in another object, an unstructured complex object cannot be queried in a declarative way. Thus, Ullman's second assertion is valid.

2. *Declarative programming.* Associated with the deductive database structure is the declarative programming approach, in which data are grouped by properties. Controversially, the object-oriented database architecture uses imperative programming, and data are grouped by properties. Controversially, the object-oriented database architecture uses imperative programming, and data are grouped by objects. Kifer and co-workers [10] presented a criteria for classifying future database prototypes. In this classification, the prototypes Pascal-R and GLUE represent imperative and relational approaches. The deductive object-oriented database architecture, such as F-logic, O-logic, and C-logic, stands between object-oriented and deductive database architectures. It utilizes declarative programming, grouping data by objects.

Based on Kifer's point of view, "pure object-oriented relational languages are not flexible enough; some knowledge is naturally imperative whereas some is naturally declarative." Therefore, even though some problems exist, the DOOD will become important if those problems can be solved, because DOOD can combine the advantages of a deductive databases structure, such as explicit specification, high-level abstraction, extendibility, modifiability, and easy comprehension, with the merits of an object-oriented structure, such as the richness of object structure and the potential of integration. However, Bancilhon in [10] insisted that it is too early to combine object-oriented and deductive database concepts since there are still some difficulties in implementing object-oriented concepts into an object-oriented database itself. He proclaimed that the object-oriented database and deductive database systems need to be developed separately rather than abruptly combining them. Moreover, in the long term, research efforts should focus on a deductive database approach, since it transfers more work to machines. Ullman agrees with this view point. Furthermore,

Ullman predicted that if it is necessary to combine these two architectures, some object-oriented features should be added to the deductive database but not vice versa.

In conclusion, as Ullman addressed earlier, the full integration between object-oriented database and deductive database is impossible. However, if some concepts in these two approaches can be removed or relaxed, it is possible to combine the merits of the two approaches. Still to be solved are the questions: should we add deductive functions into object-oriented databases or add object-oriented functions into deductive databases? and which approach will provide the greatest promise for the next database generation?

B. The Joining of Active Databases and Object-Oriented Databases

An object-oriented database organizes data and the static relationship of data into class hierarchies and provides superior complex data structure manipulation, data reusability, and the potentiality of integration and distribution. Recently, some research projects attempted to combine artificial intelligence and database management technologies with object-oriented databases so that the object-oriented database can have intelligence and operate dynamically.

Based on Dayal's opinion [15], an active database system can automatically respond to satisfied conditions and recover from constraint violation. To achieve this objective, the active database system should include rules, constraints, and triggers [26]. By definition, rules are used for deducing new knowledge from existing facts, as in a deductive database [27]. The distinction between constraints and triggers is that constraints are used for internal integrity maintenance while the triggers are used for external operations, and both can be evaluated at the moment of event processing or after finishing processing, and can apply to all objects or selected objects in the database.

However, an active object-oriented database should (1) organize and maintain data with class hierarchies, i.e., object-oriented approaches, and (2) process internal or external operations automatically. This should be done by adding constraints and triggers but not rules.

1. *Rules.* In a deductive database, rules and facts are used to represent static knowledge. The deductive database uses an inference engine to generate new facts from existing facts and rules. Unlike other database management systems, the deductive database system does not specify how to achieve but what to achieve. The deduced facts are not physically stored in the database. In fact, the new facts will be eliminated from the database after finishing the query operation. Therefore, a deductive database can provide a concise database model and high-level abstract operations.

Researchers in DOOD have attempted to combine the merits of the deductive database and the object-oriented database. Generally speaking, there are two approaches: (1) adding inference capacities into the object-oriented database [1,39], and (2) adding the object-oriented technologies into the deductive database [12,41]. However, the object-oriented technology which provides rich class structure for organizing similarities of instances conflicts with the concise data model target in the deductive database. Furthermore, declarative

programming in the deductive database versus imperative programming in the object-oriented database is the essential difference between these two database systems. Other than that, the strategy of assigning object identification in the object-oriented database also resists the approach of deducing new facts in the deductive database. From this point of view, there are rudimentary differences which create difficulties of adopting DOOD [10,37]. Therefore, the opinions of Ullman [37] and Brodie *et al.* [10] are adopted in this research. Consequently, rules are not included in the active object-oriented database.

2. *Constraints.* A constraint is used for internal consistency maintenance. Unlike the static knowledge of rules, a constraint is the representation of either *static* (state variable x) or *dynamic* (state changing variable Δx) knowledge. For example, the statement that an employee's salary must not be below the minimum salary is a static constraint while the statement that an employee's salary cannot decrease is a dynamic constraint [8]. Constraints can also be categorized into *class constraints* and *operational constraints*. The class constraints include ISA class constraints, disjoint class constraints, property induction constraints, required property constraints, single-valued property constraints, unique property constraints, inverse property constraints, keys constraints [38], and referential integrity constraints [23]. The operational constraints mainly focus on maintaining relative circumstances of objects; for example, the inventory level of component A should never be lower than 300 if the throughput is higher than 400. From this point of view, joining an active database and an object-oriented database requires both class and operational constraints.

3. *Triggers.* A trigger provides the means for an active database to respond with satisfied conditions automatically. Since a trigger can work in consonance with an object-oriented database (triggers can retrieve data from object-oriented databases without violating encapsulation), this capability should be added into the object-oriented database. Roughly speaking, a trigger is used for external control and will be activated only when state variables change (Δx). In other words, a trigger is the representation of dynamic knowledge. Unlike constraints, the conditions of triggers are unsatisfied all the time. If the conditions of events (transactions) are satisfied, the actions can be applied right away or wait for further instructions, depending on how the ECA rules are written. Also, a trigger is independent of transactions. Therefore, if the trigger is aborted, the transaction should not be aborted [20].

As has been stated, there are three kinds of triggers: once-only triggers, perpetual triggers, and timed triggers. A once-only trigger will be disabled after the action is completed and requires reactivation for the next usage. For example, if a trigger is used for ordering new parts while the inventory is lower than a certain level, the trigger should be disabled after one order has been triggered. Otherwise, the conditions will keep being satisfied in the ordering lead time. In contrast, the perpetual triggers will always be activated whenever the conditions are satisfied. Triggering alarms or alerts are the examples of applications. Timed triggers are activated by a system clock. For example, a trigger can be set for examining the inventory level every three weeks.

Adding constraints and triggers into an object-oriented database creates an active object-oriented database that has the advantages of object-oriented

database and dynamic behavior. However, when the knowledge is too complex for analysis by conventional quantitative techniques or when the available data sources are qualitative; inexact, or uncertain, the fuzzy logic controller can provide better performance than conventional approaches [22]. This is done by replacing fuzzy if–then rules with triggers or constraints and is especially useful in dynamic knowledge representation, i.e., operational constraints and triggers. Unfortunately, there is no systematic approach for finding membership functions for triggers and constraints. Fortunately, a neural network utilizes a simple computing algorithm for optimization with adapting, learning, and fault-tolerance properties. By combining fuzzy logic controllers and neural networks, the triggers and constraints are able to process many kinds of data for an active object-oriented database.

VI. FUTURE STUDY RELATED TO DATABASE TECHNOLOGIES

Section 5 presented the new database model of DOOD, and the joining of different database models. The advanced database models integrate the data models of the deductive database, the distributed database, the object-oriented database, the active database, and the multimedia database. Other than the integration of different database models, future study of the database technology will concentrate on design perspectives, knowledge explores, and system interfaces. Section A discusses the issues in software engineering. Artificial intelligence and data mining topics are presented in Sections B and C. The user-interface issues will be covered in Section D.

A. Software Engineering

The classical system development life cycle (SDLC) includes five stages: system planning, system analysis, system design, system implementation, and system maintenance. System planning analyzes company strategically level issues for developing the information system while system analysis concentrates on specific system requirements. System design describes the detailed components to meet the requirements specified in the previous stage, and then the actual coding process is completed in the system implementation stage. The operations of customer service and system testing belong to system maintenance. In SDLC, the procedure of requirements gathering and refinement, quick design, building prototyping, customer evaluation of prototype, prototype refinement, and engineer product development are critical. Therefore, how to develop ways to make the software development process easier, more effective, and more efficient are important. On the other hand, the database is designed corresponding to program development. A large centralized relational database system, which is hierarchically decomposed into smaller components, is normally adopted. However, when each component evolved and design specifications changed, a new design alternative is expected. In this situation, the existing relational database has difficulty meeting the current needs. Therefore, other database models, such as object-oriented models and deductive models, can provide a new paradigm.

In system engineering, CASE tools support application generation from high level specification and can reduce the effort and cost in application development. A integrated case environment provides (1) common user interface, (2) utilities, (3) trigger mechanism, (4) metadata, i.e., object definitions, relationships, and dependencies among objects of arbitrary granularity, design rules, etc., and (5) shared project data repository [32]. The software utilities (1) can reduce the work of certain tasks for people using requirements specification and analysis tools, conceptual design tools, tools for view integration, tools for schema mapping among models, tools for physical design and optimization, and partitioning/allocation tools; (2) improve the performance of certain machines using performance monitoring, tuning, reorganizing, and reconstructing tools along with requirement tracing, configuration management, audits trails, dependency tracking, and data versioning. The shared project data repository is a database that must be able to control a wide variety of object types including text, graphics, bit maps, complex documents, report definitions, object files, test data, and results.

B. Artificial Intelligence

How to implement the knowledge in the database is always important. The knowledge comes in various forms: (a) structural knowledge: knowledge about dependencies and constraints among the data; (b) general procedural knowledge: knowledge described only by a procedure or a method; (c) application-specific knowledge: knowledge determined by the rules and regulations applicable in a specific domain; and (d) enterprise-directing knowledge: adding artificial intelligent functions into the database and generating a knowledge base for storing intentional knowledge are still substantial topics. For example, creating intelligent agents to retrieve correct data at the right time requires further study.

Thus, neural network technology is a keen tool for developing the intelligent agents. A neuron has two phase computations: input phase and output phase. The input phase computation, called *basic function*, is used to process the input signals such as weights w_{ij} and input data x_i. The weights are assigned arbitrarily at the beginning stage of the training process and are later tuned up through algorithms. The input data of the neurons can either come from environment or neurons. Two kinds of basic functions are commonly used: linear and radial. The output of basic function computation is the net value u_j. This value will become the input data for the output phase computation called *activation function*. Therefore, a neuron output value is obtained from $a_j(\mathbf{W}, \mathbf{X}) = f(u_j(\mathbf{W}, \mathbf{X})) = a_j(u_j(\mathbf{W}, \mathbf{X}))$. A neuron network, also called *connectionist*, is a collection of neurons synapticly connected. The types of connection structures are also an important factor in the neural network designing because they decide the data transmission direction. Three major connections are the following:

1. *Feedforward connections.* The data are passed from input nodes to output nodes. In other words, the data are propagated from the lower level to the higher level.

2. *Feedback connections.* The data are passed in the opposite direction of feedforward connection networks. However, they are the complementary connections of the feedforward networks.

3. *Lateral connection.* The lateral connections allow the data to be passed laterally. The winner-take-all network is a typical application.

Another important factor in designing neural networks is the size of the network. The size factor is measured through the number of layers or the number of hidden nodes. The system with the hidden layer can be used to simulate the nonlinear model. However, the number of hidden layers and nodes can only be decided through experiments. Therefore, more than one network with different hidden neurons should be trained simultaneously at the early design stages. Then the best performance can be picked.

1. *Fixed-weight networks.* The weights of fixed-weight networks are assigned early and will not be changed in the training process. A set of threshold values will be assigned to neurons and plays an important role for pattern retrieval. The connection of this kind of network can be feedforward, feedback, auto-associate, or hetero-associate. The most successful models are the Associative Memory Model, Hamming Net, and Hopfield Net.

2. *Unsupervised networks.* The training data in unsupervised networks are only composed of input data. There is no expected output data for teaching. The learning capacity comes from the experience of previous training patterns. The weights for both unsupervised and supervised networks are updated through $w_{ij}^{(\kappa+1)} = w_{ij}^{(\kappa)} + \Delta w_{ij}^{(\kappa)}$. The training period of unsupervised networks is longer than supervised networks, and less accurate results may be obtained. The most commonly used unsupervised networks are Neocognitron, Feature Map, Competitive Learning, ART, and Principal Component.

3. *Supervised networks.* To build a supervised network, two steps are required: training and retrieving. In the training step, the weights are learned (adjusted) to classify training patterns till no more adjustment is needed. Then the networks are trained appropriately and ready for retrieving patterns. The output of the retrieving step is a function, called *discriminate function*, of weights and input values. The training data set is composed of input and expected output. These types of networks have been the mainstream of the neural network arena. The examples of famous supervised networks are Percetron, Decision-Based NN, ADALINE (LMS), Multilayer Perceptron, Temporal Dynamic Models, Hidden Markov Model, Back-propagation, and Radial-Basis Function (RFB) Models. Basically, there are two kinds of teaching patterns in supervised networks: decision-based formulation and approximate-based formulation. In the decision-based formulation, telling the system correctness of training patterns and finding a set of weights to yield the correct output are the training objectives. The Percetron is one such example and will be discussed later. The purpose of the approximate-based formulation is to find the optimal set of weights that can minimize the error between the teaching patterns and the actual output. The most famous example in this category is the back-propagation algorithm.

C. Data Mining

Since more and more data are stored in the database, e.g., data generated from science research, business management, engineering application, and other sources, finding meaning from data becomes difficult. Therefore, assistance to help find meaning from data is expected urgently. Data mining technology can discover information from data, and has drawn more attention recently. Data mining, also called knowledge discovery, knowledge extraction, data archaeology, data dredging, or data analysis in a database, is a terminology of nontrivial extraction of implicit, previously unknown, and potentially useful information from data in a database. The discovered knowledge can be applied to areas such as information management, query processing, decision making, and process control.

The data mining techniques can be classified based on [13]:

1. What kinds of database to work on. Normally, the relational data model is the focal point. Not surprisingly, data mining technologies on other database models, such as object-oriented, distributed, and deductive databases, require further exploration and should have astonishing results.
2. What kind of knowledge to be mined. There are several kinds of data mining problems: classification, association, and sequence. The classification problems group the data into clusters while association problems discover the relationships between data. The sequence problems focus on the appeared sequence of data [3]. For solving these problems, there are several well-known data mining techniques, such as association rules, characteristic rules, classification rules, discriminate rules, clustering analysis, evolution, deviation analysis, sequences search, and mining path traversal [6]. Other than that, machine learning approaches are also being adopted, such as neural network, genetic algorithm, and simulated annealing [7]. The following introduce several techniques:

(a) *Association rules.* An association rule discovers the important associations among items such as that the presence of some items in a transaction will imply the presence of other items in the same transaction. Several famous algorithms of the association rule approach are Apriori [2], DHP (Dynamic Hash Pruning) [6], AIS (Agrawal, Imielinski, and Swami) [2], Parallel Algorithms [3], DMA (Distributed Mining of Association Rules) [14], SETM (Set-Oriented Mining) [21], and PARTITION [34].

(b) *Characteristic rules.* Characteristic rules are also called data generalization. It is a process of abstracting a large set of relevant data in a database from a low concept level to relatively high ones since summarizing a large set of data and presenting it at a high concept level is often desirable. The examples of characteristic rules are data cube approach (OLAP) and attribute-oriented induction approach. OLAP will be discussed later, and the attribute-oriented induction takes a data mining query expressed in an SQL-like data mining query language and collects the set of relevant data in a database.

(c) *Classification rules.* A classification rules technique is a kind of supervised training that finds common properties among a set of objects in a database and classifies them into different classes according to a classification model. Since it is a supervised training algorithm, training sets of data are

needed in advanced. First, analyze the training data using tools like statistics, machine learning, neural networks, or expert systems to develop an accurate description. The classification variables can be categorical, ranking, interval, or true measure variables. Categorical variables tell to which of several unordered categories a thing belongs. Ranking variables put things in order, but don't tell how much larger one thing is than another. Interval variables measure the distance between two observations, and true variables, e.g., age, weight, length, volume, measure from a meaningful zero point.

(d) *Clustering analysis.* Unlike classification rules, the clustering analysis technique is an unsupervised training. It helps to construct meaningful partitioning by decomposing a large scale system into smaller components to simplify design and implementation. Clusters are identified according to some distance measurement in a large data set. Normally, the process takes a longer time than the classification rules.

(e) *Pattern-based similarity search.* The pattern-based similarity searches for similar patterns in a database. Different similarity measures such as Euclidean distance, distance between two vectors in the same dimension, and the correlation, linear correlation between data, are normally used.

(f) *Mining path traversal pattern.* The mining path traversal pattern captures user access patterns and path. The information can be used to analyze the user's behavior, such as the tendency of depth-first search, breadth-first search, top-down approach, and bottom-up approach.

3. What kind of techniques to be utilized. Data mining can be driven based on an autonomous knowledge miner, data-driven miner, query-driven miner, or interactive data miner. Different driven forces trigger different mining techniques, and the database is also used in different ways. The techniques can also be identified according to its underlying data mining approach, such as generalization-based mining, pattern-based mining, statistics- or mathematical theories-based mining, and integrated approaches.

D. User Interfaces

The user interfaces of a database system is also another topic that can be improved. A database user needs (1) customized languages for easily fitting their problem domain and (2) alternative paradigms for accessing the database. In the future, data can be presented in different formats such as three-dimensional, visualization, animation, and virtual reality. Therefore, a new concept of the user interface is expected. A natural language interface is one of the choices. The natural language interface demands multilingual support to interact between people and computers, and would allow complex systems to be accessible to everyone. People would be able to retrieve data and information using a natural language without the need of learning complicated query languages. The new technology would be more flexible and intelligent than is possible with current computer technology. The applications can be divided into two major classes [4]:

1. *Text-base applications.* The text-based interface uses documented expression to find appropriate articles on certain topics from a database of texts,

extract information from messages on certain topics, translate papers from one language to another, summarize texts for certain purpose, etc.

2. *Dialogue-based applications.* The dialogue-based application would resolve the problems of communication between human and machine firstly. The system is not just process speech recognition. Rather, it provides smooth-flowing dialogue; therefore, the system needs to participate actively in order to maintain a natural dialogue. A simple system can use a question-and-answer system to query a database. A complete system should be able to automate operations, such as customer service over the telephone, language control of a machine, and general cooperative problem-solving systems.

VII. FUTURE DATABASE APPLICATIONS

There are several applications that will increase the importance of database systems, such as data warehousing, on-line analytic processing (OLAP), decision-support system, and engineering and production applications.

A. Data Warehousing

The business automation of diverse computer systems and the increasing of service quality for a flexible marketplace have requested large data amounts. On the other hand, the computer and network technology have improved to support requests on the large data amounts and versatile data types. Also the user-friendly interface eases the difficultly of using computers and therefore increases the degree of user dependence on computers. The data warehousing technology is the process of bringing together disparate data from throughout an organization for decision-support purposes [7]. The large amount of data comes from several sources such as (a) automated data input/output device, i.e., magnetic-ink character recognition, scanner, optical mark recognition, optical character recognition, bar code, and voice input [11], and (b) data interchange, i.e. electrical bank, point-of-sale, POS, electronic data interchange, EDI, ATM machine, adjustable rate mortgages, just-in-time inventory, credit cards, and overnight deliveries.

The data warehouse stores only data rather than information. Therefore, for getting the right information at the right time, the data warehousing technology is normally implemented with data analysis techniques and data mining tools. In order to implement data, the different levels of abstraction show that data exists on several interdependent levels. For example, (1) the operational data of whom, what, where, and when, (2) the summarized data of whom, what, where, and when, (3) the database schema of the data, tables, field, indexes and types, (4) the logical model and mappings to the physical layout and sources, (5) the business rules of what's been learned from the data, and (6) the relationship between data and how they are applied can help to support different levels of needs [7]. When applying analysis and mining tools, the actionable patterns in data is expected. That is, the consistent data are required. In this case, a database management system is the heart of the data warehousing. However,

the current research on data mining technology is only limited to RDBMS. Further study on other data models are needed.

There are some common types of the data warehousing system: (1) operational data warehouse which stores raw data, (2) departmental data warehouse which stores summarized representations of interest to a single application area, (3) interdepartmental data warehouse which uses downloads and uploads to exchange the data between departments and does not have a logical data model.

The data warehouse can locate three different types of architecture: client/server networks, 3-tier networks, and distributed networks (Fig. 4).

1. *Client/server network.* The client/server network allows disparate source systems to talk to each other via middleware tools. The middleware does not include the software that provides the actual service. There are three types of middleware.

(a) *General middleware.* The middleware is responsible for processing the communication stacks, distributed directories, authentication services, network time, remote procedure calls, and queuing services. Products that fall into this category are OSF's DCE, Netware, Name Pipes, LAN Server, LAN Manger, Vines, TCP/IP, APPC, and NetBIOS.

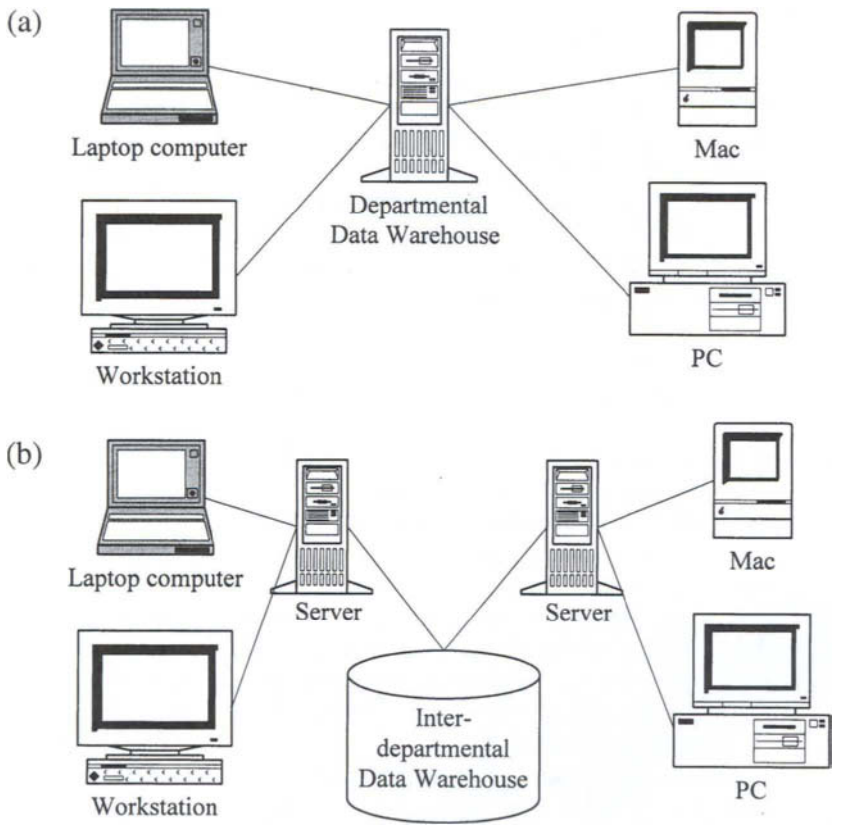


FIGURE 4 The data warehouse can be located in different types of architecture such as client/server networks, 3-tier networks, and distributed networks. (a) Departmental data warehouse, (b) interdepartmental data warehouse, (c) middleware approach, and (d) multitiered approach.

(b) *Service-specific middleware*. The middleware accomplishes a particular client/server type of service. Examples for database-specific middleware are ODBC, IDAPI, DRDA, EDA/SQL, SAG/CLI, and Oracle Glue; for OLTP-specific middleware are Tuzedo's ATMI, Encina's Transactional RPC, and X/Open's TxRPC and XATMI; for groupware-specific middleware are MAPI, VIM, VIC, and Lotus Notes calls; for object-specific middleware are OMG's ORB and Object Services and ODMG-93; and for system management-specific middleware are SNMP, CMIP, and ORBs [30].

There are several.different types of client/server architectures:

(a) *File servers*. The server is only responsible for managing data as files. The files are shared across a network.

(b) *Database server*. The server uses its won processing power to find the requested data instead of singularly passing all the records to a client and letting it find its own data as was the case of the file server.

(c) *Transaction servers*. The client invokes remote procedures who reside on the server with an SQL database engine to accomplish jobs.

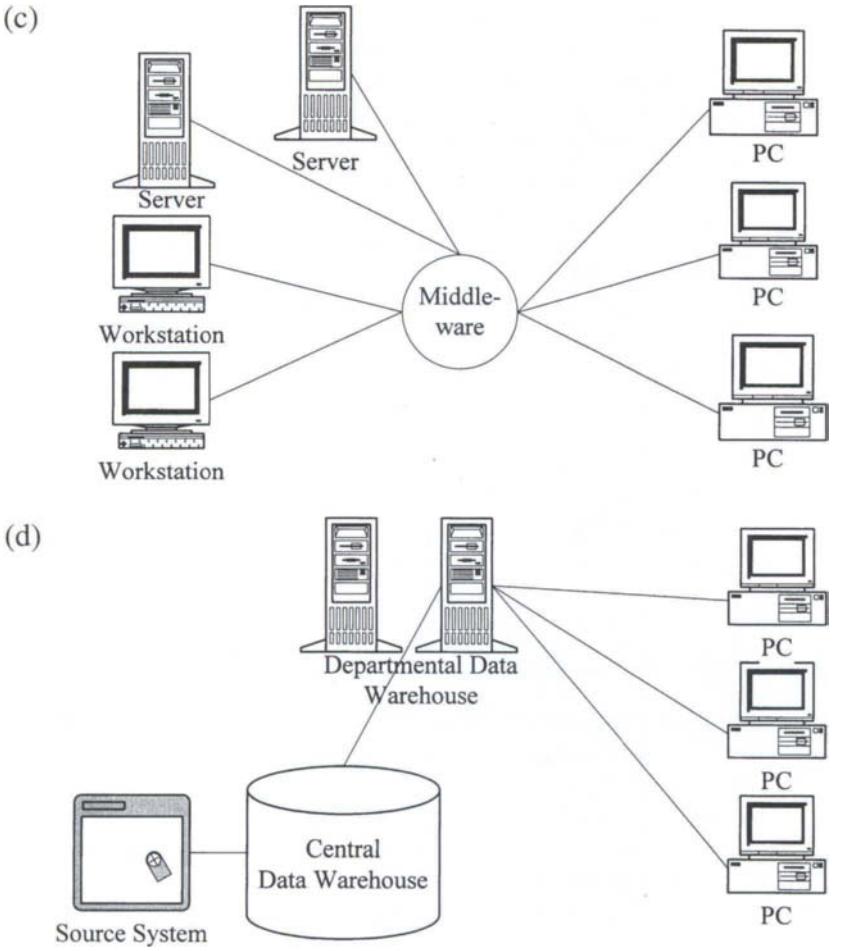


FIGURE 4 (*continued*)

(d) *Groupware.* Groupware, also called collaborative software, is a kind of software used for documents, imaging, and workflow management. The software lets a group of people share information or track information together [11]. Therefore, to work together effectively on a project, the data can be stored in a central place or in a distributed manner. Again, most of the data depository of the groupware is limited to the relational database, or simply a data storage. The integration between other data models with the groupware remains for further study.

(e) *Object servers.* In an object server, the client objects communicate with server objects using an object request broker (ORB). The ORB is located at the middleware and interacts with the instance of object server class, invokes the requested method, and returns the results back to the client object. Examples of object servers are DSOM from IBM, DOMS HyperDesk, DOMF from HP, and DOE from Sun.

2. *3-tier approach.* The 3-tier approach separate the architecture into three levels: source systems, data transport, and central repository. The first tier, source systems, belongs to the client such as the visual aspects of the business object. The second tier, data transport, belongs to traditional servers, such as common object requested broker architecture (CORBA) objects, who act as the middle-tier application servers. CORBA encapsulates the business logic, interacts with the client, implements the logic of the business object, and extracts their persistent state from multiple data sources. The object management model (OMG) has published standards for 16 object services: life cycle, persistence, naming, event, concurrency control, transaction, relationship, externalization, query, licensing, properties, time, security, collection, and startup. The third tier, central repository, provides an integrated model of disparate data source and back-end applications, i.e., TP Monitors, MOM, DBMS, ODBMS, Lotus Notes [29]. It is also worth noting that the central repository can be any kind of database model and will be changed to meet the modeling requirements thereafter.

3. *Distributed system.* A distributed system shares the load of processes data, providing services, and storing data from client/server and 3-tier architectures toward architecture. The responsibilities of data warehousing are shared from a central repository to many computers. It is clear that the advantages of economics and reliability can also be found in the distributed data warehouse. The distributed system has gradually drawn more and more attention.

B. On-line Analytic Processing (OLAP)

On-line analytic processing is a presentation tool for providing different views of large sets of data to end users no matter whether the data resides in a single centralized data warehouse, in virtual distributed warehouses, or on operational systems. The OLAP is a fast and powerful way of reporting data and can enable manual knowledge discovery. Therefore, the human intelligence for knowledge discovery is important. The data can be presented in a powerful and efficient representation, called a cube, through an advanced graphical interface. The cube is ideally suited for queries that allow users to look for specific targets for decision-support purposes (as shown in Fig. 5). The cube itself is stored either

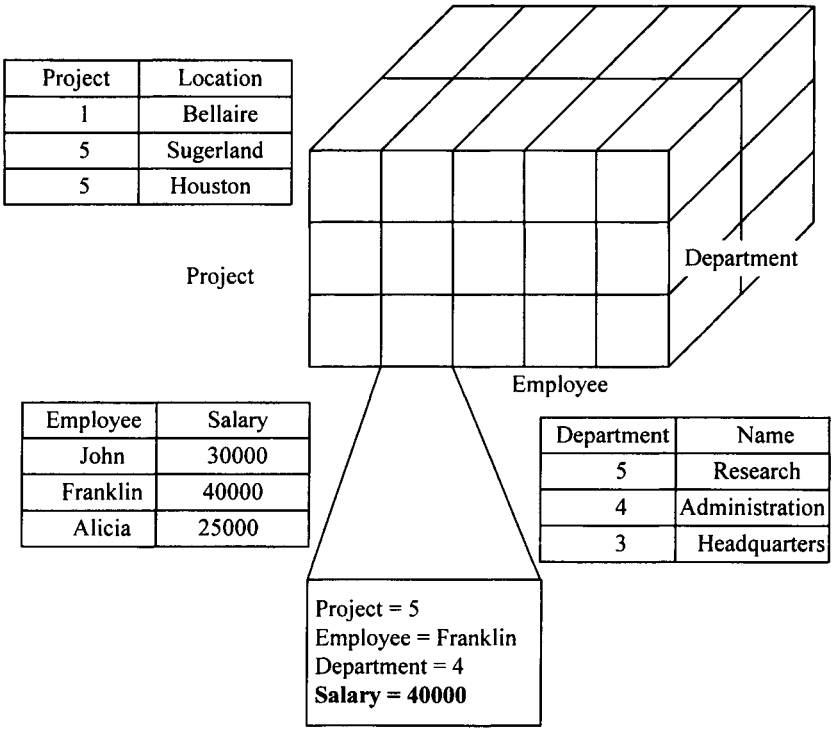


FIGURE 5 The cube used for OLAP is divided into subcubes. Each subcube can contain the calculations, counts, and aggregations of all records landed on it.

in a relational database or in any other kind of database that optimizes OLAP operations.

The most famous of OLAP representation is the star schema by Ralph Kimball [7]. The star schema starts with central fact tables that correspond to facts about a business issue. Each row in the central fact table contains some combination of keys that make it unique (again, a concept of the relational database) and these keys are called dimensions. The central facts table also has other columns to contain its own information. Associated with each key are auxiliary tables called dimension tables. Any query on the central fact table would require multiple joins of the dimension tables. Since many joining operations are required, indexing is important with the key attributes. Other than star schema, the multidimensional database (MDD) is also an important representation of OLAP. The MDD is a way of representing data that comes in relational tabular form. Each record is exactly represented in a subcube. Inside a subcube are calculations, counts, or aggregations of all the records that belong to it. Therefore, the useful information can be revealed to users quickly.

C. Decision-Support System

A decision support system (DSS) improves the performance of users through implementing information technology to increase knowledge level. The system is comprised of three components: database management software (DBMS),

model base management software (MBMS), and dialogue generation and management software (DGMS) [35]. The MBMS integrates data access and decision models to provide (1) creation generation functions, (2) maintenance update functions, and (3) manipulation use functions. The decision model is embedded in the MBMS of an information system, and uses the database as the integration and communication mechanism between strategic models, tactical models, and operational models. The output of MBMS is generated toward decision makers or intellectual processes. The model should have the functions of:

(a) the ability to create new models quickly and easily,
(b) the ability to catalog and maintain a wide range of models, supporting all levels of management,
(c) the ability to interrelate these models with appropriate linkage through the database,
(d) the ability to access and integrate model "building blocks", and
(e) the ability to manage the model base with management functions analogous to database management.

The DGMS has three parts: (1) active language, (2) display or presentation language, and (3) knowledge base (what the user must know). The DGMS has the knowledge base to interact with the active language to instruct what the user can do and with the presentation language to show what the user can see. In this situation, a good man/machine interface for interactive applications is required. Therefore, the interface should have:

(a) the ability to handle a variety of dialogue styles,
(b) the ability to accommodate user actions in a variety of media,
(c) the ability to present data in a variety of formats and media, and
(d) the ability to provide flexible support for the user's knowledge base.

The DBMS can be any data model as previous discussed. The distributed database model can generate distributed decision support systems, and an active system can increase the alive operations. Moreover, the deductive database can produce an intelligent decision support environment.

D. Engineering and Production Applications

Engineering and production applications are a structured approach to integrate production functions. This approach can result in significant savings in both time and cost. The application topics include financial functions, order processing and accounting functions, capacity requirements planning, inventory management, manufacturing performance planning, production data management, production monitoring and control, purchasing, sale forecasting, order processing, product planning, production control, inventory control, and material resource planning (MRP). An engineering application is a tedious and complicated procedure in which designers need to manage a large amount of data into applications, and the data needs to be maintained for specific applications. The engineering design has the characteristics of (1) long duration, (2) complex design object, (3) hierarchical object architecture, (4) multiple versions, and (5) cooperation. A database management system has been adopted

for sharing information and data management among applications. In order to integrate different applications, several protocols for data exchange have been proposed. An example can be the Standard for Exchange of Product model, abbreviated as STEP. However, conventional engineering applications employ a relational database system to manage data. There are obstructions in a relational database system [19]:

1. Engineering design requires complex data types. A relational database system is suitable for processing a high degree of similarity among data. The complex data type cannot be supported effectively by the relational database.
2. A relational database system is more appropriate for using in short-string or fixed-length types of data. Engineering applications, e.g., engineering design, require large data type.
3. Temporal and spatial data are important in planning, assembly, or allocating. The examples can be found in the topics such as facility layout production scheduling and solid modeling.
4. A database schema must be evolved frequently in the engineering applications for improving design quality.
5. A designer may need to edit designs for a long period of time, which requires a long transaction in the database. This approach places a locking mechanism on data and prevents other designers from working simultaneously.
6. The old and new versions of the same data need to be retained for reference. The key constraints of a relational database prohibit it from storing the same data twice.

In contrast to the obstructions in relational database systems, an object-oriented database has characteristics which are important in engineering usage [19]:

1. *Common model.* A common model allows designers to map real-world objects into a data object directly.
2. *Uniform interface.* Objects generated from the same class will have the same interface in operation.
3. *Complex objects.* Complex data types and relationships, including hierarchies and lattices, are supported by an object-oriented database.
4. *Information hiding and abstraction.* The abstraction process focuses on the generalization and aggregation of objects, which hides internal detailed differences to simplify the architecture.
5. *Versions.* Multiple versions of the same object are permitted in an object-oriented database.
6. *Modularity, flexibility, extensibility, and tailorability.* Adding new objects or editing old objects are simple procedures through the schema evolution function.

In the future, the database must support areas such as computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided engineering (CAE), manufacturing requirement planning (MRPII), computer-integrated manufacturing (CIM), computer-aided logistical system (CALS), and

enterprise-resource planning (ERP). In order to meet such a diversified environment, the distributed and heterogeneous environment and active database should be adopted.

VIII. SUMMARY

In this chapter, we presented the techniques and applications of emerging database system architecture. The history of computers, information systems, and database systems are covered. The relational data model, deductive data model, object-oriented data model, distributed data model, active database model, deductive and object-oriented database (DOOD), and the joining of active databases and object-oriented databases are discussed. It is worth noting that different data models can satisfy different needs, and future system requirements are diversified. In the future, database performance can be increased because of the improvement of system engineering, artificial intelligence, data mining, and user interface. Many techniques will also enhance the application of database systems, such as data warehousing, on-line analytic processing (OLAP), decision-support systems, and engineering and production applications.

REFERENCES

1. Abiteboul, S. Towards a deductive object-oriented database language. *Deductive and Object-Oriented Database*, 453–472, 1990.
2. Agrawal, R., Imielinshi, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proc. of the 1993 International Conference on Management of Data (SIGMOD-93)*, pp. 207–216, 1993.
3. Agrawl, R., and Shafer, J. Parallel mining of association rules. *IEEE Trans. Knowledge Data Engrg.* 8(6), 962–969, 1996.
4. Allen, J. *Natural Language Understanding*. Benjamin/Cummings, Redwood City, CA, 1995.
5. Banerjee, J. *et al.* Data model issues for object-oriented applications. In *Readings in Database Systems*, (M. Stonebreaker, Ed.), 2nd ed., pp. 802–813. Morgan Kaufmann, San Mateo, CA, 1994.
6. Beeri, C., and Milo, T. A model for active object oriented database. In *Proceedings of the 17th International Conference on Very Large Data Bases.* (Sep.), pp. 337–349, 1991.
7. Berry, M., and Linoff, G. *Data Mining Techniques: For Marketing. Sales, and Customer Support.* Wiley, New York, 1997.
8. Bertino, E., and Martino, L. *Object-Oriented Database Systems: Concepts and Architecture.* Addison-Wesley, Reading, MA, 1993.
9. Bochmann, G. *Concepts for Distributed Systems Design* Springer-Verlag, New York, 1983.
10. Brodie, M. L., Bancilhon, F., Harris, F., Kifer, M., Masunada, Y., Sacerdoti, E., and Tanska, K. Next generation database management systems technology. *Deductive and Object-Oriented Database* 335–346, 1990.
11. Capron, H. L. *Computers: Tools for an Information Age*, 5th ed. Addison-Wesley, Reading, MA, 1998.
12. Caseau, Y. Constraints in an object-oriented deductive database. *Deductive and Object-Oriented Database* 292–310, 1991.
13. Chen, M., Han, J., and Yu, P., Data mining: An overview from a database perspective. *IEEE Trans. Knowledge Data Engrg.* 8(6): 866–883, 1996.

14. Cheung, D. W., Vincent, T., Fu, W., and Fu, Y. Efficient mining of association rules in distributed databases. *IEEE Trans. Knowledge Data Engrg.* 8(6): 911–922, 1996.
15. Dayal, U. Active database management systems. In *Proceedings of the Third International Conference on Data and Knowledge Base,* Jerusalem, pp. 150–170, 1988.
16. Dayton, U., and Hsu, M. Organizing long-running activities with triggers and transactions. In *Readings in Database Systems,* (M. Stonebreaker, Ed.), 2nd. ed., pp. 324–334. Morgan Kaufmann, San Mateo, CA, 1994.
17. Du, T., and Wolfe, P. Overview of emerging database architectures. *Comput. Indust. Engrg.* 32(4): 811–821, 1997.
18. Du, T., and Wolfe, P. An implementation perspective of applying object-oriented database technologies. *IIE Trans.* 29: 733–742, 1997.
19. Elmasri, R., and Navathe, S. B. *Fundamentals of Database Systems,* 3rd ed. Addison-Wesley, Reading, MA, 2000.
20. Gehani, N., and Jagadish, H. V. Ode as an active database: Constraints and triggers. In *Proceedings of the 17th International Conference on Very Large Data Bases,* pp. 327–336, 1991.
21. Houtsma, M., and Swami, A. Set-oriented data mining in relational databases. *Data Knowledge Engrg.* 17: 245–262, 1995.
22. Lee, C. C. Fuzzy logic in control systems: Fuzzy logic controller—Parts I and II. *IEEE Trans. Systems Man Cybernet.* 20 (2): 404–418, 419–435, 1990.
23. Little, D., and Misra, S. Auditing for database integrity. *J. Systems Manage.* 6–10, 1994.
24. Long, L., and Long, N. *Brief Edition of Computers,* 5th ed. Prentice-Hall, Upper Saddle River, NJ, 1998.
25. Loomis, M. *The Database Book.* Macmillan, New York, 1987.
26. McCarthy, D., and Dayal, U. The architecture of an active data base management system. In *Reading in Database Systems* (M. Stonebreaker, Ed.), 2nd ed., pp. 373–382. Morgan Kaufmann, San Mateo, CA, 1994.
27. Minker, J. *Fundations of Deductive Databases and Logic Programming.* Morgan Kaufmann, Los Altos, CA, 1988.
28. O'Brien, J. A. *Introduction to Information Systems,* 8th ed. Irwin, Chicago, 1997.
29. Orfali, R., and Harkey, D. *Client/Server Programming with JAVA and CORBA.* Wiley, New York, 1997.
30. Orfali, R., Harkey, D., and Edwards, J. *Essential Client/Server Survival Guide.* Van Nostrand Reinhold, New York, 1994.
31. Ozsu, T. M. *Principles of Distributed Database Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1991.
32. Pressman, R. *Software Engineering: A Practitioner's Approach,* 3rd ed. McGraw-Hill, New York, 1992.
33. Ramakrishnan, R., and Ullman, D. A survey of deductive database systems. *J. Logic Programming* 125–148, 1995.
34. Savaere, A., Omiecinski, E., and Navathe, S. An efficient algorithm for association rules in large databases. In *Proc. Int'l Conf. Very Large Data Bases,* Zurich, pp. 432–444, 1995.
35. Sprague, R., and Watson, H. *Decision Support Systems: Putting Theory into Practice,* 3rd ed. Prentice-Hall, Englewood Cliffs, NJ, 1993.
36. Stonebreaker and the Committee for Advances DBMS Function. Third-generation database system manifesto. In *Reading in Database System,* pp. 932–945. Morgan Kaufmann, San Mateo, CA, 1994.
37. Ullman, J. A comparison between deductive and object-oriented database systems. In *Proceedings Deductive and Object-Oriented Database Conference,* pp. 7–24, 1990.
38. Urban, S., and Lim, B. An intelligent framework for active support of database semantics. *Int. J. Expert Systems.* 6(1): 1–37, 1993.
39. Wong, L. Inference rules in object oriented programming systems. *Deductive and Object-Oriented Database* 493–509, 1990.
40. Zaniolo, C. Object identity and inheritance in deductive databases—An evolutionary approach. *Deductive and Object-Oriented Database* 7–24, 1990.

This Page Intentionally Left Blank

2
DATA MINING

DOHEON LEE

Department of BioSystems, Korea Advanced Institute of Science and Technology, Daejon, Republic of Korea

MYOUNG HO KIM

Department of Computer Science, Korea Advanced Institute of Science and Technology, Taejon 305-701, Republic of Korea

This chapter introduces the concept and various techniques of data mining. Data mining is defined as the nontrivial extraction of implicit, previously unknown, and potentially useful knowledge from a large volume of actual data. Various techniques including data characterization, classification, and association rule discovery are discussed. Especially, the treatment of fuzzy information in data characterization is explained in detail. Finally, the relationship between data mining and data warehousing is briefly addressed.

I. INTRODUCTION

A. Definition of Data Mining

According to the Oxford dictionary, "mining" means the activity of digging for useful mineral resources such as coal and ores from the ground. In the context

Database and Data Communication Network Systems, Vol. 1

of data mining, "mineral resources" and "the ground" are mapped into "knowledge" and "data," respectively. That is, data mining is the activity of exploring data for useful knowledge. Although there are several precise definitions of data mining in the recent literature, we adopt the definition as "the nontrivial extraction of implicit, previously unknown, and potentially useful knowledge from large volume of actual data" since it represents essential attributes of data mining comprehensively [1]. There are several terms in the literature to represent the same or similar area with data mining such as knowledge discovery in databases, data archeology, and data visualization, to name a few. However, we adopt the term, "data mining," in this chapter since it is the most common term in recent days.

One typical example of data mining might be as follows. After exploring sales records in a point-of-sales system for a large retail shop, a marketing manager may discover a fact such that "customers who purchase baby diapers are likely to purchase some bottles of beer." This example is famous among data mining practitioners, since it is an unpredictable knowledge before the data mining is actually performed. Who can expect the relationship between diapers and beer? However, once such a pattern is discovered in actual data, the marketing manager could exploit it for redesigning the self-layout and deciding a target customer list for advertisement of new products. He/she is very likely to achieve competitive advantages over others without such knowledge.

Let us examine each phrase of the definition in detail with counterexamples to get more concrete insight for data mining. Firstly, the phrase "implicit" means that information stored explicitly in the database or the system catalog is not the subject. The results for ordinary database queries and the schema information such as column names and data types are examples of explicit knowledge. Secondly, the phrase "previously unknown" means that we are not looking for well-known knowledge. Suppose we come to know that "most adult men are taller than five feet" by exploring the national health information records. It is rarely worth the effort of exploration since the information is common sense. Thirdly, "potentially useful" implies that the data mining process is driven by application requirement. The cost of data mining should be rewarded by the business benefit. The last phrase, "from a large volume of actual data," distinguishes data mining from experimental machine learning. Since data mining is performed on a large volume of data, it is hard to adopt such algorithms whose execution times increase fast as the data size grows. In addition, missing or corrupted values in actual data require a more sophisticated treatment of data.

B. Motivation of Data Mining

As a variety of disciplines including business automation, engineering support, and even scientific experiments have begun to rely on database systems, the advance of database system technology has become faster, and in turn, the volume of data stored in databases has increased rapidly in recent days. It has been estimated that the amount of information in the world doubles every 20 months, and the total number of databases in the world was estimated at five million in 1989. Earth observation satellites, planned for the 1990s, are

expected to generate one terabyte of data every day and the federally funded Human Genomc project will store thousands of bytes for each of the several billion genetic bases. The census databases are also typical examples of a large amount of information [1]. As an example around our daily life, several millions of sales records in retail stores are produced every day.

What are we supposed to do with this flood of raw data? It is no longer helpful to make direct exposure of raw records to users. If it is be understood at all, it will have to be analyzed by computers. Although simple statistical techniques for data analyses were developed long ago, advanced techniques for intelligent data analyses are not yet mature. As a result there is a growing gap between data generation and data understanding [1]. At the same time, there is a growing realization and expectation that data, intelligently analyzed and presented, will be a valuable resource to be used for a competitive advantage [1].

The necessity of data mining capability is emphasized by this circumstance. A National Science Foundation workshop on the future of database research ranked data mining among the most promising research and engineering topics [2]. Decision support systems (DSS), executive information systems (EIS), and strategic information systems (SIS) are also typical examples that solicit knowledge through data mining techniques rather than scatter-brained, raw data chunks through simple SQL queries.

C. Comparison with Machine Learning

In the context of machine learning, there are four levels of learning situations as follows [3].

1. *Rote learning:* The environment provides the learning entity with how-to-do information precisely. Conventional procedural programs written in C language are typical examples. In such programs, the learning entity, i.e., the processor, is given step-by-step instructions to accomplish the goal. The learning entity does not have to infer anything. It just follows the instructions faithfully.
2. *Learning by being told:* The environment provides the learning entity with general information such as rules. For example, suppose that a rule is given as "if A is a parent of B, and B is a parent of C, then A is a grandparent of C." When there are facts such as "Tom is a parent of John" and "John is a parent of Mary," the learning environment is able to deduce a new fact that "Tom is a grandparent of Mary." Many expert systems and query processors for deductive databases have adopted this level of learning. This level of learning is also called deductive learning.
3. *Learning from examples:* The environment provides the learning entity with a set of detail facts. Then the learning entity has to infer general rules representing common properties in the set of given facts. For example, suppose that we have the facts such as "a pigeon can fly" and "a swallow can fly." From these facts, the learning entity can induce a rule that "a bird can fly" based on the knowledge that "a pigeon and a swallow are kinds of birds." Once the learning entity obtains the knowledge, it can utilize the knowledge to deduce new facts such as "a sparrow can fly." This level of learning is also called inductive learning.

4. *Learning from analogy:* The environment provides the learning entity with a set of facts and/or rules. When an unknown problem is given, the learning entity looks for known facts or rules similar to them. For example, the most effective treatment for a given patient is probably the treatment that resulted in the best outcomes for similar patients. This level of learning is also called memory-based reasoning (MBR).

The former two levels have been already well understood and exploited in actual applications. Most computer programs around us are primarily based on the first level, i.e., rote learning. Success stories about expert systems such as chemical analyses and medical diagnoses mostly belong to the second level of learning, i.e., deductive learning. However, the latter two levels have been at the stage of research activities and prototyping experiments.

Since data mining is the activity of learning patterns or general knowledge from a large volume of data, it can be regarded as a sort of the third level of machine learning, i.e., inductive learning. However, note that some data mining practitioners use the term, "data mining," in a broader meaning that includes most conventional data analysis techniques. For example, software packages for statistical hypothesis test and visual data analysis are often claimed as data mining tools. In this chapter, we limit the scope of data mining to the level of inductive learning in order to concentrate on the technical aspects of data mining.

Frawley *et al.* have distinguished data mining techniques from conventional inductive learning with respect to the input data to be explored [1]. It is summarized in Table 1.

While the learning data set for conventional inductive learning is typically predefined, the input data for data mining is apt to be dynamically changed. This difference raises both the possibility and challenge of incremental algorithms; i.e., algorithms take the changed portion of the data instead of the whole as their inputs. Actual databases are likely to include erroneous data, uncertain data, and missing data due to mistakes in database construction or genuine deficiency of information. Since data mining is performed on actual databases rather than test data sets, it must cope with such problems. It should be robust in noisy data while it should not miss important exceptions. Whereas test data sets are

TABLE 1 The Differences between Data Mining and Conventional Inductive Learning with Respect to the Characteristics of Input Data

Data mining	Conventional inductive learning
Dynamic data	Static data
Erroneous data	Error-free data
Uncertain data	Exact data
Missing data	No missing data
Coexistence with irrelevant data	Only relevant data
Immense size of data	Moderate size of data

collected carefully to contain only data relevant to the learning process, actual databases also include irrelevant data. Data mining techniques must identify relevant data before and/or in the middle of the learning process. Since data mining is performed on the immense size of data, it is unrealistic to directly apply inductive learning algorithms whose execution times increase fast as the data size grows.

In consequence, data mining can be regarded as the application of the inductive learning paradigm to an actual environment. Although many conventional inductive learning techniques such as classification and clustering can be adopted as starting points for data mining techniques, we must modify and supplement them for actual environment.

D. Steps of the Data Mining Process

Data mining techniques should be integrated into the business process in order to maximize its usefulness. The following six-step procedure is given for typical data mining processes. Although the procedure is not a panacea for all situations, it might be a helpful reference [4, 5].

1. *Identify the business problem:* As the first step, we must identify the business problem precisely, and define it in a formal manner. This is an essential gradient in data mining processes. Although it sounds intuitive and straightforward, it may not be in practice. Data mining for the sake of data mining itself is rarely successful.
2. *Prepare and transform the data to be explored:* There can be many internal and external sources of data for the data mining application. We must identify them and prepare access methods to them. In addition, we must transform the data to the proper format for the data mining application. It is seldom for them to be already in the proper format, since they are collected for their own operational purposes not for data mining.
3. *Apply specific data mining techniques:* We apply a specific data mining technique or a combination of several techniques on the prepared data.
4. *Act based on the obtained knowledge:* At this step, we exploit the obtained knowledge for our business problems. For example, customer purchase patterns can be used to plan an advertisement program for a new product.
5. *Measurement of the business benefit:* It is quite important to measure the business benefit that is gained from applying the knowledge.
6. *Assimilation of the knowledge:* Finally, we must incorporate the business insights gained so far into the organization's business and information systems.

It is quite interesting to note that the major effort for a data mining process is concentrated on the data preparation rather than the application of specific data mining techniques. It has been reported that the cost of data preparation is over 50% among the total process [5]. Actually, many data mining practices suffer from the difficulty of data preparation. However, we will concentrate on the third step, i.e., the application of specific data mining techniques, since other steps have many managerial aspects beyond the scope of this chapter.

II. OVERVIEW OF DATA MINING TECHNIQUES

A. Classification of Data Mining Techniques

Recently, there have been a large number of works on data mining in universities, research institutions, and commercial vendors. There are several criteria to classifying those works. The most common criterion is the type of knowledge to be obtained. According to the criterion, data mining techniques are classified into characterization, classification, clustering, association discovery, sequential pattern discovery, prediction, and so on.

Characterization is the activity of obtaining generalized descriptions to represent a large number of detail data records [6–9]. Classification is the activity of finding out rules to discriminate a group of objects from others [10–12], whereas clustering is grouping similar objects based on certain similarity measures [13, 14]. Association rules represent the co-occurrence tendency of multiple events [15, 16]. A rule such as "if an event A occurs, it is very likely for an event B to occur simultaneously" is an example of the association rule. The sequential pattern is a variation of the association rule, which considers relative orders of occurrence [17]. Prediction is the activity of interpolating or extrapolating an unknown value of interest based on known values. Linear regression is a typical example of prediction. To the best of our knowledge, there does not exist any notion of completeness for types of knowledge to be obtained. It means that it is quite possible to develop new types of knowledge according to the application requirements evolved continuously.

The other criteria for classification of data mining techniques can be types of input data. Even though many input data are stored in relational databases currently, useful data can reside in legacy file systems, object-oriented databases, object-relational databases, spatial databases, multimedia databases, Internet information-base, and so on. According to the type of input data, it may be required to differentiate data mining techniques. Especially, Internet information-base is regarded as the fruitful resource for data mining recently [18].

Meanwhile, since various techniques in several disciplines such as machine learning, statistics and database management are combined to provide data mining solutions, we can classify data mining techniques based on the types of adopted techniques. For example, techniques based on symbolic artificial intelligence are likely to produce human-understandable knowledge while neural network techniques may not.

B. Requirements for Effective Data Mining

When we design and utilize data mining techniques, we must consider the following requirements.

1. The technique should have sufficient efficiency and scalability: As we mentioned earlier, the input data of data mining is of immense size over hundreds of *giga*bytes in many cases. Thus, algorithms with exponential or high-order polynomial time complexity are seldom useful. Commonly, we try to achieve the time complexity to be linear or less than quadratic in the worst cases.

2. It is desirable to provide chances of user interaction in the middle of the process at various levels of abstraction: Since data mining is the activity of exploration, it is hard to expect the result before the exploration process. Thus, it is effective to allow the user interaction in the middle of the process in order to drill down or change the exploration focus.

3. It should handle various types of data in an integrated way: It is apt to have disparate sources of input data for the specific data mining. Thus, the technique should handle multiple types of data such as categorical data and continuous data in an integrated way.

4. It should be possible to measure the usefulness and the risk of the results: Since all the data mining results are not necessarily useful with respect to the application requirement, we should have proper measures of usefulness to distinguish better ones. In addition, some data mining results may include the degrees of genuine uncertainty. Careless adoption of uncertain results could come up with negative effects on the business operation. Thus, we should have proper measures of uncertainty to avoid it.

5. It should be possible to represent the results in understandable ways: The result of data mining activities is the subsequent input for the next step, i.e., business actions on the result. In many cases, business actions are performed by human entity. The data mining results need to have understandable forms.

6. It should consider the issue of data security: In many application domains, the result of data mining is highly sensitive to the security of the organization. For example, fraud detection techniques for credit card companies should be hidden for effective operations. In addition, likely to the cases of statistical databases, there are situations where individual information can be deduced from multiple aggregated information stuffs. It could violate the security policy of the organization.

III. DATA CHARACTERIZATION

Data characterization provides the user with comprehensive information for grasping the essence of a focused database portion in an understandable manner. It also establishes a starting point to make useful inferences from large collections of data, and facilitates easy communication of observations about the problem domain [8].

Informally, our definition of data characterization is a task that reduces a large number of actual database records into a relatively small number of generalized descriptions, i.e., generalized records. For example, a computer usage log table whose attribute scheme is (PROGRAM, USER, TIME), containing thousands of usage log records such as ⟨vi, John, 23:20⟩ and ⟨emacs, Tom, 23:31⟩, could be reduced into a few generalized records, say, ⟨editor, programmer, around midnight⟩. It delivers an assertion that programmers have executed editor programs around midnight.

In this section, we introduce two approaches for data characterization. One is top-down characterization [6, 7] and the other is bottom-up characterization [9]. The top-down approach introduced herein also accommodates fuzzy information. The bottom-up characterization is called attributed-oriented induction.

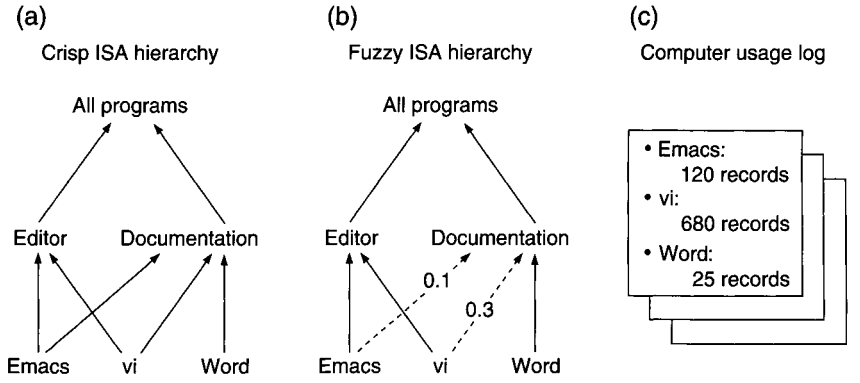


FIGURE 1 (a) and (b) show crisp and fuzzy ISA hierarchies: Dotted lines represent partial ISA relationships as strong as the augmented fraction numbers between two incident concept nodes, while solid lines denote complete ISA relationships. (c) depicts a situation where database summarization is to be done.

A. Top-Down Data Characterization[1]

Since data characterization can be regarded to summarize the content of a database, we use the terms "database summarization" and "data characterization" interchangeably herein. Among several requirements for effective data characterization techniques, we concentrate on the following ones: Firstly, it must be allowed to represent data characteristics in terms of fuzzy concepts, i.e., concepts with fuzzy boundaries, since crisp concepts are occasionally too restrictive to express complex situations [19, 20]. Secondly, it must be possible to utilize fuzzy domain knowledge, since actual domain knowledge is apt to include fuzziness inherently. Thirdly, users must be able to interact with a characterization process to reflect their own discovery purposes.

ISA hierarchies are commonly used to exploit "specialization" relationships among domain concepts. However, ISA hierarchies including only crisp ISA relationships are not sufficient to express actual domain knowledge. For example, suppose we know that programs "emacs" and "vi" are used to edit source codes, and a program word is used to write documents. In addition, suppose we also know that some users execute 'emacs' and 'vi' to write documents only on rare occasions. With only crisp relationships, there is no other way to represent the above-mentioned domain knowledge except as shown in Fig. 1a. If fuzzy relationships can be expressed, however, we would have an ISA hierarchy as shown in Fig. 1b.

Let us consider how different are the results these two ISA hierarchies yield, in data characterization. Suppose a computer usage log table during a certain period contains 120, 680, and 25 log records of "emacs," "vi," and "word" executions, respectively, as shown in Fig. 1c, and we want to determine whether "editor" or "documentation" programs are mainly executed in that period. With the crisp ISA hierarchy, we cannot identify the majority of usage since $120 + 680 = 800$ records are for "editor" programs, and $120 + 680 + 25 = 825$ records are for "documentation" programs. On the other hand, if we exploit

[1]Based on "Database Summarization Using Fuzzy Hierarchies" by Doheon Lee and Myoung Ho Kim, which appeared in IEEE Transactions on Systems, Man, and Cybernetics, vol. 27, no. 4, pp. 671–680 ©1997 IEEE.

the fuzzy ISA hierarchy, we can conclude that "editor" programs have been mostly executed, since 120 and 680 records of "emacs" and "vi," respectively, are known mainly for editing source codes not for writing documents.

1. Representation of the Database Summary

Herein, we define a generalized record as a representational form of a database summary, and elaborate how to evaluate the validity of a generalized record with respect to a given database. We assume that all attributes appear in a single table, i.e., the universal relation assumption, without loss of generality, to avoid unnecessary complexity of the presentation. However, this work can be applied to any other data models where a database record can be regarded as a series of attribute values.

a. Generalized Records

There are many domain concepts having fuzzy boundaries in practice. For example, it is difficult to draw a crisp boundary of documentation programs in a set of programs since some programs such as "vi" and "emacs" can be thought as source code editors for some programmers but as word processors for some manual writers. It is more natural and useful to express such domain concepts in terms of fuzzy sets rather than crisp sets [20]. Thus, we use a vector of fuzzy sets to effectively represent a database summary.

A fuzzy set f on a domain D is defined by its membership function $\mu_f(x)$, which assigns a fraction number between 0 and 1 as the membership degree to each domain value [19]. Since $\mu_f(x)$ represents the degree to which an element x belongs to a fuzzy set f, a conventional set is regarded as a special case of a fuzzy set whose membership degrees are either one or zero. Given two fuzzy sets, f_1 and f_2, on a domain D, f_1 is called a *subset* of f_2 and denoted as $f_1 \subseteq f_2$, iff $\forall x \in D, \mu_{f1}(x) \leq \mu_{f2}(x)$. In this paper, a special fuzzy set f such that $\forall x \in D, \mu_f(x) = 1$, is denoted as ω.

Definition 1 (Generalized Record). A *generalized record* is defined as an m-ary record $\langle f_1, \ldots, f_m \rangle$ on an attribute scheme $(A_1, \ldots, A_m)$, where f_j's are fuzzy sets and A_j's are attributes. Given two different generalized records $g_1 = \langle f_{11}, \ldots, f_{1m} \rangle$ and $g_2 = \langle f_{21}, \ldots, f_{2m} \rangle$, on the same attribute scheme, g_1 is called a *specialization* of g_2 iff $\forall_j$, $f_{1j} \subseteq f_{2j}$.

A generalized record $\langle f_1, \ldots, f_m \rangle$ on an attribute scheme $(A_1, \ldots, A_m)$ is interpreted as an assertion that "each record has $f_1, \ldots, f_m$ for attributes $A_1, \ldots, A_m$, respectively." Note that an ordinary database record is also regarded as a generalized record whose fuzzy sets are singleton sets. A singleton set is a set having a unique element. An example of a generalized record with respect to an attribute scheme (PROGRAM, USER) is ⟨editor programmer⟩. It implies an assertion that "the program is an editor and its user is a programmer"; in other words, "programmers have executed editor programs."

b. Validity of Generalized Records

From the viewpoint of inductive learning, a given database (or a part of it) and a set of possible generalized records are regarded as an instance space and a pattern space respectively [3]. Our summarization process searches a pattern space to choose valid generalized records with respect to a given instance space, i.e., a given database. Recall that the goal of database summarization is

to generate representative descriptions embracing as many database records as possible. Thus, the validity of a generalized record is determined by the number of database records that are compatible, i.e., of statistical significance. In the following this notion is formulated as the support degree.

Definition 2 (Support Degree). The *support degree* of a generalized record $g = \langle f_1, \ldots, f_m \rangle$ with respect to a given collection of database records C whose attribute scheme is $(A_1, \ldots, A_m)$ is defined as

$$SD(g|C) = \left\{ \sum_{ti \in C} \otimes[\mu_{f1}(t_1.A_1), \ \ldots \ , \mu_{fm}(t_1.A_m)] \right\} \Big/ |C|,$$

where $\otimes$ is T-norm operator [20]. $\mu_{fj}(t_i.A_j)$ denotes the membership degree of an attribute A_j of a record t_i with respect to a fuzzy set f_j, and $|C|$ denotes the cardinality of the collection C. We call generalized records with support degrees higher than a user-given threshold value *qualified generalized records.*

Note that we do not use the term, a set of database records or a database relation, because a collection of database records is allowed to have duplicates. We will denote $SD(g/C)$ as $SD(g)$ for simplicity as long as the data collection C is obvious in the context. T-norm operators are used to obtain conjunctive notions of membership degrees in the fuzzy set theory [20]. Examples include MIN and probabilistic product operators. Since the usage of T-norm operators in the fuzzy set theory is analogous with that of the product operator in the probability theory, the symbol $\otimes$, which is analogous with $\times$, is commonly used to denote a specific instance of a T-norm operator.

Since $\mu_{fj}(t_i.A_j)$ denotes the membership degree of an attribute A_j of a record t_i with respect to a fuzzy set f_j, a *T-norm* value over membership degrees of all attributes of a record t_i, i.e., $\otimes[\mu_{fi}(t_1.A_1), \ldots \ , \mu_{fm}(t_1.A_m)]$, represents how strongly the record t_i supports the assertion of a generalized record $\langle f_1, \ldots, f_m \rangle$. As a result, the support degree of a generalized record is the normalized sum of such support strength of individual database records. In other words, the support degree implies the fraction of supporting database records to the total data collection.

The defined support degree has the following properties:

- Boundary conditions:
  Given a generalized record g,
  — $0 \leq SD(g) \leq 1$.
  — If all fuzzy sets in g are ω, $SD(g) = 1$.
- Monotonicity:
  Given two generalized records, g_1 and g_2 on the same attribute scheme,
  $SD(g_1) \leq SD(g_2)$ if g_1 is a specialization of g_2.

While boundary conditions are self-evident by definition, the monotonicity property needs some explanation. The following theorem shows the monotonicity property.

Theorem 1 (Monotonicity of the support degree). Given two generalized records, $g_1 = \langle f_{11}, \ldots, f_{1m} \rangle$, and $g_2 = \langle f_{2l}, \ldots, f_{2m} \rangle$, on an attribute scheme

TABLE 2 The Support Strength of Example Data Records

Record	A_1	A_2	$\mu_{n1}(A_1)$	$\mu_{n2}(A_2)$	t_i supports $\langle f_1, f_2 \rangle$ as strong as
t_1	a	α	1.0	0.3	$\otimes(1.0, 0.3) = 0.3$
t_2	b	β	0.1	1.0	$\otimes(0.1, 1.0) = 0.1$
t_3	c	α	0.1	0.3	$\otimes(0.1, 0.3) = 0.1$

$(A_1, \ldots, A_m)$,

$$SD(g_1) \leq SD(g_2) \quad \text{if } \forall_j \in \{1, \ldots, m\},\ f_{1j} \subseteq f_{2j},$$

Proof. The premise of the theorem implies that $\forall j \in \{1, \ldots, m\}, [\forall x \in DOM(A_j), \mu_{f1j}(x) \leq \mu_{f2j}(x)]$, where $DOM(A_j)$ denotes the domain of an attribute A_j.

When a collection of database records is C,

$$SD(g_1) = \left\{ \sum_{t1 \in C} \otimes_{j \in \{1,\ldots,m\}} [\mu_{f1j}(t_i.A_j)] \right\} \Big/ |C|$$

$$\leq \left\{ \sum_{t1 \in C} \otimes_{j \in \{1,\ldots,m\}} [\mu_{f2j}(t_i.A_j)] \right\} \Big/ |C|, \text{ since}$$

$$\mu_{f1j}(t_i.A_j) \leq \mu_{f2j}(t_i.A_j) = SD(g_2).$$

Thus, $SD(g_1) \leq SD(g_2)$.

Let us look at an example of support degree computation. Suppose a generalized record g on an attribute scheme (A_1, A_2) is $\langle f_1, f_2 \rangle$, where fuzzy sets f_1 and f_2 are $\{1.0/a, 0.1/b\}$ and $\{0.3/\alpha, 1.0/\beta\}$, respectively. If a data collection C is given as Table 2, $SD(g)$ is computed as follows: The first record t_1 supports g as strong as 0.3, since its first and second attribute values, a and α, belong to fuzzy sets, f_1 and f_2, to the degrees, 1.0 and 0.3, respectively. Note that we use the MIN operator for the T-norm operation just for illustration throughout this paper. Similarly, both the second and third records support g as strong as 0.1. As a result, we can say that the generalized record g is supported by $0.3 + 0.1 + 0.1 = 0.5$ records out of a total of three records, i.e., 17% of the data collection.

2. Fuzzy Domain Knowledge

An ISA hierarchy is an acyclic digraph (N,A), where N and A are a set of concept nodes and a set of ISA arrows, respectively. If there is an ISA arrow from a concept node n_1 to another concept node n_2, we say that n_1 ISA n_2; in other words, n_1 is a specialized concept of n_2. While conventional ISA hierarchies have only crisp ISA arrows, fuzzy ISA hierarchies include fuzzy ISA arrows. The meaning of a fuzzy ISA arrow from n_1 to n_2 can be interpreted as n_1 is a partially specialized concept of n_2. Without loss of generality, we suppose that the root node of any fuzzy ISA hierarchy is the special fuzzy set ω, and each terminal node is a singleton set whose unique element is an atomic domain value, i.e., a value appearing in actual database records.

TABLE 3 Level-k Fuzzy Sets

Set label	Membership function	Level
engineering	{1.0/editor, 1.0/docu. 0.8/spread}	3
business	{1.0/docu, 1.0/spread}	3
editor	{1.0/emacs, 1.0/vi}	2
documentation	{0.1/emacs, 0.3/vi, 1.0/word}	2
spreadsheet	{0.1/word, 1.0/wright}	2
emacs	{1.0/emacs}	1
vi	{1.0/vi}	1
word	{1.0/word}	1
wright	{1.0/wright}	1

Fuzzy ISA hierarchies are too flexible of a structure to be used directly in database summarization. Thus, we provide a method to resolve a given fuzzy ISA hierarchy into a collection of fuzzy sets defined on the same domain, and a fuzzy set hierarchy that focuses on the complete inclusion relationships.

Definition 3 (Fuzzy Set Hierarchy). *A fuzzy set hierarchy* is a partially ordered set $(\Gamma, \subseteq)$, where Γ is a set of fuzzy sets defined on the domain D. The binary relation $\subseteq$ is the (complete) set inclusion relationship between two fuzzy sets. A fuzzy set f_1 is called a direct subset of another fuzzy set f_j if $f_i \subseteq f_j$ and there is no other f_k such as $f_i \subseteq f_k \subseteq f_j$.

Recall that a fuzzy set f_i is said to be (completely) included by a fuzzy set f_j in the same domain, if for each domain element x, $\mu_{fi}(x) \leq \mu_{fj}(x)$.

a. Transforming a Fuzzy ISA Hierarchy to a Fuzzy Set Hierarchy

In fuzzy set theory [21], the elements of a fuzzy set can themselves be fuzzy sets, rather than atomic domain values. Ordinary fuzzy sets whose elements are atomic values are called level-1 fuzzy sets. Fuzzy sets whose elements are level-(k-1) fuzzy sets are called level-k fuzzy sets. Table 3 depicts some level-k fuzzy sets. If two fuzzy sets have different levels, we cannot directly determine the inclusion relationship between them, since the domains are different. However, the level of a fuzzy set can be either upgraded or downgraded by some fuzzy set-theoretic treatments. Thus, if we want to determine the inclusion relationship between two fuzzy sets with different levels, we must adjust their levels to the same through upgrading or downgrading levels.

Upgrading the level of a fuzzy set is trivial, since a level-k fuzzy set can be simply rewritten as a level-$(k+1)$ singleton set whose unique element is the original level-k fuzzy set. For example, a level-2 fuzzy set editor in Table 3 can be thought as a level-3 fuzzy set such as {1.0/editor}.

Downgrading the level of a fuzzy set is done by a support fuzzification technique based on the extension principle [20]. Rather than spending a large space to explain support fuzzification precisely, we will explain it by example. Interested readers are recommended to refer to Zadeh's orginal paper [22].

The transformation procedure of a fuzzy ISA hierarchy to a fuzzy set hierarchy is composed of three steps as follows:

1. Downgrade several levels of fuzzy sets in a fuzzy ISA hierarchy to level-1 fuzzy sets.

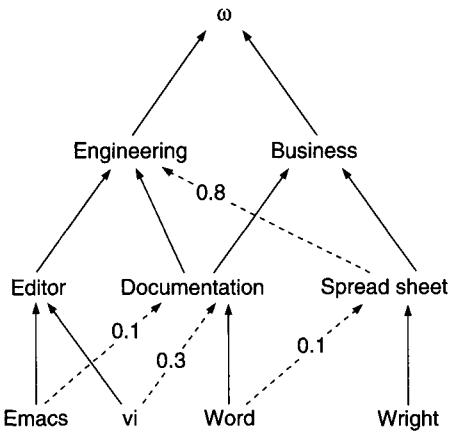


FIGURE 2 A fuzzy ISA hierarchy on computer programs.

2. By probing pairwise inclusion relationships, elicit partial order relation on those level-1 fuzzy sets obtained in the previous step.
3. Draw arrows between fuzzy sets and their direct subsets based on the partial order relation.

Let us demonstrate the transformation procedure step by step with an example. Figure 2 shows an example fuzzy ISA hierarchy on computer programs that could be used in computer usage analysis. Note that a fuzzy set f in a fuzzy ISA hierarchy is a level-k fuzzy set, if the maximal path length from f to terminal nodes is k-1. The several levels of fuzzy sets in the fuzzy ISA hierarchy of Fig. 2 are in Table 3.

In the first step, all level-k ($k>1$) fuzzy sets are downgraded to level-1 fuzzy sets through support fuzzification. For example, a level-3 fuzzy set engineering in Table 3 is transformed to a level-2 fuzzy set as follows:

$$
\begin{aligned}
\text{engineering} &= \{1.0/\text{editor}, 1.0/\text{documentation}, 0.8/\text{spreadsheet}\} \\
&= \{1.0/\{1.0/\text{emacs}, 1.0/\text{vi}\}, 1.0/\{0.1/\text{emacs}, 0.3/\text{vi}, 1.0/\text{word}\}, \\
&\quad 0.8/\{0.1/\text{word}, 1.0/\text{wright}\}\} \\
&= \{\otimes(1.0, 1.0)/\text{emacs}, \otimes(1.0, 1.0)/\text{vi}, \otimes(1.0, 0.1)/\text{emacs}, \\
&\quad \otimes(1.0, 0.3)/\text{vi}, \otimes(1.0, 1.0)/\text{word}, \otimes(0.8, 0.1)/\text{word}, \\
&\quad \otimes(0.8, 1.0)/\text{wright}\} \\
&= \{1.0/\text{emacs}, 1.0/\text{vi}, 0.1/\text{emacs}, 0.3/\text{vi}, 1.0/\text{word}, 0.1/\text{word}, \\
&\quad 0.8/\text{wright}\} \\
&= \{\oplus(1.0, 0.1)/\text{emacs}, \oplus(1.0, 0.3)/\text{vi}, \oplus(1.0, 0.1)/\text{word}, \\
&\quad 0.8/\text{wright}\} \\
&= \{1.0/\text{emacs}, 1.0/\text{vi}, 1.0/\text{word}, 0.8/\text{wright}\},
\end{aligned}
$$

where $\otimes$ and $\oplus$ are a T-norm and a T-conorm operator, respectively. In contrast with T-norm operators, T-conorm operators are used to obtain disjunctive combinations of membership degrees. Herein, we use MIN and MAX for T-norm and T-conorm operations just for illustration. However, there are also several alternatives for T-conorm operators.

TABLE 4 Level-1 Fuzzy Sets Obtained through Support Fuzzification from Level-k (k>1) Fuzzy Sets

Value	editor	docu	spread	engi	busi	ω
emacs	1.0	0.1	0.0	1.0	0.1	1.0
vi	1.0	0.3	0.0	1.0	0.3	1.0
word	0.0	1.0	0.1	1.0	1.0	1.0
wright	0.0	0.0	1.0	0.8	1.0	1.0

Note. The leftmost column enumerates atomic domain values and other columns represent membership degrees of atomic domain values for the fuzzy sets listed in the first row.

To envisage the implication of support fuzzification, let us consider why the membership degree of the element word is determined as 1.0. Since documentation is a member of engineering, and word is a member of documentation, word is also regarded as a member of engineering. By this transitivity, the membership degree of word to engineering is determined as $\otimes(\mu_{\text{engineering}}(\text{documentation}), \mu_{\text{documentation}}(\text{word})) = \otimes(1.0, 1.0) = 1.0$. Meanwhile, the alternative transitivity that spreadsheet is a member of engineering, and word is a member of spreadsheet, also implies that word is regarded as a member of engineering. If we follow the latter transitivity, the membership degree of word to engineering is determined as $\otimes(\mu_{\text{engineering}}(\text{spreadsheet}), \mu_{\text{spreadsheet}}(\text{word})) = \otimes(0.8, 0.1) = 0.1$. Note that as far as either of such two transitivity relationships exists, i.e., the disjunctive combination of two facts, the membership of word to engineering holds. Thus, the membership degree of word to engineering is concluded as $\oplus(1.0, 0.1) = 1.0$.

Note that the elements, "emacs," "vi," "word," and "wright" appearing in the final line of the above formula are not atomic domain values. They are level-1 fuzzy sets as depicted in Table 3. That is, they can be rewritten as {1.0/emacs},{1.0/vi},{1.0/word}, and {1.0/wright}. If we downgrade again the obtained level-2 fuzzy set engineering to level-1, we have the same looking fuzzy set such as {1.0/emacs}, {1.0/vi}, {1.0/word}, and {0.8/wright}. However now, the elements are atomic domain values. The reason we treat the terminal nodes in a fuzzy ISA hierarchy as level-1 singleton sets rather than just atomic domain values is in order to achieve a unified representation of domain concepts and domain values. As a result of this first step, we have a collection of level-1 fuzzy sets as in Table 4.

In the second step, we compare the obtained level-1 fuzzy sets in a pairwise manner to probe the inclusion relationships. Finally, we draw arrows between fuzzy sets and their direct subsets based on the obtained partial order relation. Figure 3 depicts the fuzzy set hierarchy obtained.

3. Data Characterization Process

Now we present a process for discovering qualified generalized records based on given fuzzy domain knowledge, i.e., fuzzy set hierarchies. In short, our summary discovery process looks for qualified generalized records in a top-down manner. It initially hypothesizes the most generalized record, i.e., a generalized record whose fuzzy sets are all ω's. The process specializes the most

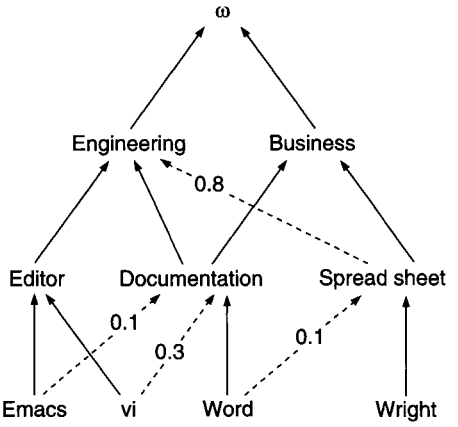


FIGURE 3 The fuzzy set hierarchy derived from a fuzzy ISA hierarchy.

generalized record, i.e., $\langle \omega, \ldots, \omega \rangle$, based on the given fuzzy set hierarchies to search for more specific generalized records while remaining qualified. The specialization is done minimally in the sense that only one fuzzy set is specialized into its direct subset. By evaluating support degrees of those specializations with respect to a given collection of database records, qualified generalized records are identified. At this point, human users can interact with the discovery process.

They might choose only some qualified generalized records for further consideration if they are not interested in the others or want to trade in the search completeness for reducing the search cost. The process minimally specializes again only user-chosen qualified generalized records. Those specializations become hypotheses for the next phase. After some repetitive steps of such specialization, the process yields a specialization hierarchy of qualified generalized records i.e., significant database summaries. Figure 4 diagrams the process, and Fig. 5 depicts the steps in detail.

Note that the monotonicity property of the support degree in Theorem 1 guarantees that any specializations of unqualified generalized records cannot be qualified, and as a result, the process never misses qualified generalized records, even though it does not specialize unqualified generalized records.

In Fig. 5, the support degree of each generalized record is computed from Line 5 to Line 7. This is the most time-consuming part of the process except the user interaction (Line 12), since the process must scan disks to read each database record. In Line 9, qualified generalized records with respect to τ are identified, and they are put into the result. After users choose only interesting generalized records in Line 12, they are minimally specialized in the sub-function specialize(). The process comes back to Line 4 with those specializations to repeat the steps.

Let us consider the efficiency of the process in terms of the number of disk accesses. Because it is hard to estimate how much time it takes to interact with human users, let us assume that users choose all qualified generalized records in Line 12 for the efficiency analysis. It is common to analyze the efficiency of most disk-based database applications in terms of disk access costs [23]. It is because the cost of disk accesses is much more expensive than that of in-memory

Cf. GR: generalized record

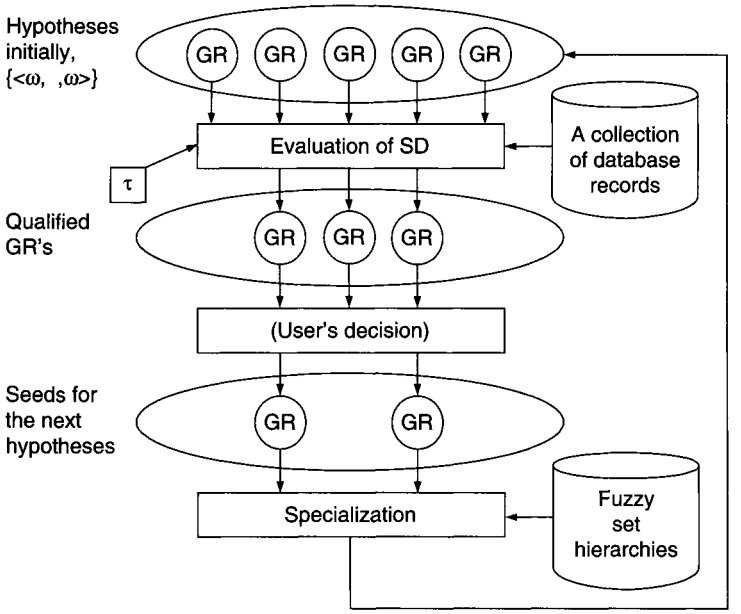


FIGURE 4 A brief diagram of the summary discovery process.

```
INPUT:
  (i) A collection of database records C. (ii) fuzzy set hierarchies for attributes,
  (iii) a support degree threshold value τ
OUTPUT:
  A specialization hierarchy of qualified generalized records
(1)   SPGR()
(2)   {
(3)      result = Ø ; curr = { < ω , ..., ω > } ;
(4)      while(curr = Ø ){
(5)            for each t in C
(6)               for each g in curr
(7)                  accumulate SD(g);
(8)            for each g in curr
(9)               if SD(g) > r, then result = result U g,
(10)              else curr = curr – g;
(11)           for each g in curr {
(12)              if the USER marks g then specialize(curr,g);
(13)              curr = curr – g;
(14)           }
(15)     }
(16)  }

(17)  specialize (set, g = < a1 ,..., am >)
(18)  {
(19)     for i = 1 to m
(20)        for each direct subset saj of aj
           (in the fuzzy set hierarchy for the jth attribute)
              set = set U < a1 , ... ,saj , ... am >)
(22)  }
```

FIGURE 5 A specialization process of generalized records (SPGR).

operations. The disk access cost actually determines the efficiency of a process if the number of in-memory operations does not increase exponentially along with the input size, and no external network-based communication is involved.

The number of generalized records whose support degrees are evaluated in one specialization phase, i.e., the size of "curr" in Line 6, has nothing to do with the number of database records. Rather, it is determined by the average fan-outs of given fuzzy set hierarchies. Empirically, we expect that the system memory buffer space can hold all those generalized records in one specialization phase.

Then the total number of disk accesses is the number of specialization phases multiplied by the number of disk accesses to read database records in C. Let us denote the maximal path length of a fuzzy set hierarchy on the jth attribute as l_j. Since the specialization of generalized records are done attribute by attribute (see line 16 to 21), the number of specialization phases cannot be greater than $\sum_j (l_j)$. As a result, the number of disk accesses is no greater than $\sum_j (l_j) \times p$, where p denotes the number of disk pages containing the collection C of database records. Note that like the size of curr, $\sum_j (l_j)$ is also determined by given fuzzy set hierarchies not by the number of database records. Thus, we claim that the average cost of our summary discovery process increases linearly along with the number of database records in a collection if we ignore a human user's interaction.

Let us see the behavior of the algorithm with an example. Suppose that we have a collection of computer usage records whose attributes are PROGRAM and USER as shown in Table 5. Given fuzzy ISA hierarchies on attributes, PROGRAM and USER are supposed to be resolved to fuzzy sets in Table 6 and fuzzy set hierarchies in Fig. 6. The fuzzy sets in Table 6 are represented in the form of semantic relations [23]. Semantic relations represent several fuzzy sets on the same domain in the relational form. If the domain is a continuous

TABLE 5 An Example Collection of Computer Usage Records

Program	User	Program	User
emacs	John	emacs	Tom
vi	Tom	gcc	John
emacs	Tom	wright	Steve
vi	John	emacs	Jane
word	Kimberly	emacs	Mary
emacs	John	tetris	John
word	Mary	emacs	Jane
emacs	John	ultima	Mary
word	Kimberly	emacs	John
word	Kimberly	emacs	John
emacs	John	ultima	Mary
word	Mary	emacs	Jane
emacs	John	tetris	John
word	Kimberly	emacs	Mary
vi	John	emacs	Jane
emacs	Tom	wright	Steve
vi	Tom	gcc	John
emacs	John	emacs	Tom

TABLE 6 Semantic Relations Representing Fuzzy Sets in the Fuzzy Set Hierarchies

Value	compiler	editor	docu	spread	engi	busi	game	ω
			For PROGRAM_01					
gcc	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
cc	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
f77	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
emacs	0.0	1.0	0.1	0.0	1.0	0.1	0.0	1.0
vi	0.0	1.0	0.3	0.0	1.0	0.3	0.0	1.0
word	0.0	0.0	1.0	0.1	1.0	1.0	0.0	1.0
wright	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
tetris	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
ultima	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0

Value	prog	writer	seller	account	develop	market	ω
			For USER_01				
John	1.0	0.0	0.0	0.0	1.0	0.0	1.0
Tom	1.0	0.0	0.0	0.0	1.0	0.0	1.0
Mary	0.2	0.8	0.0	0.0	1.0	0.8	1.0
Kimberly	0.0	1.0	0.1	0.0	1.0	1.0	1.0
Steve	0.0	0.0	1.0	0.0	0.0	1.0	1.0
Jane	0.0	0.0	0.4	1.0	0.0	1.0	1.0
Bob	0.0	0.0	0.0	1.0	0.0	1.0	1.0

interval, a semantic relation partitions the domain into disjoint subintervals and assigns a representative membership degree to each subinterval. Semantic relations are treated in the same way as ordinary database tables, and as a result, we do not have to deviate from the framework of conventional data models. Even though we adopt and recommend semantic relations as a proper representation of fuzzy sets for the benefit homogeneity, our discovery process is not tied to a specific fuzzy set representation in principle.

Figures 7 to 10 shows the behavior of the summary discovery process SPGR along with the specialization phases. The threshold value of support degrees is given as 0.4. Each figure corresponds to a single `while` loop in Fig. 5. In the first specialization, among six generalized records derivable from the root generalized record, only three depicted in Fig. 7 are qualified. If users are interested in the computer usage of developers, they would mark the middle one, i.e., ⟨-, developer⟩, for further consideration. By specializing the marked one and evaluating support degrees of derived hypotheses, the process yields qualified generalized records as shown in Fig. 8. Figures 9 and 10 show subsequent specializations.

From the final hierarchy of generalized records in Fig. 10, we can conclude that developers used engineering programs mostly. In particular, programmers have been using heavily executed editor programs.

4. Informativeness of Generalized Records

This section derives a measure for informativeness of generalized records based on Shannon's information theory [24]. Although the proposed data characterization process comes up with several qualified generalized records, the

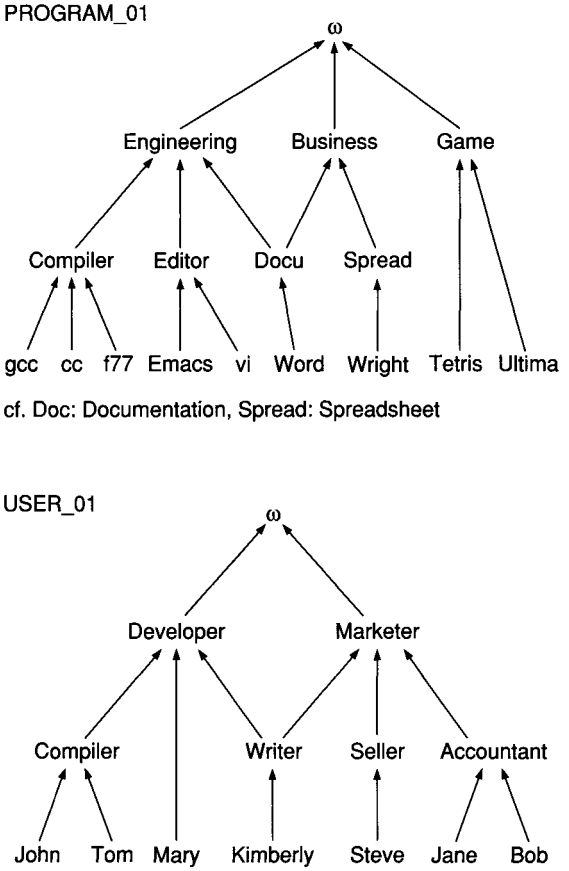


FIGURE 6 Fuzzy set hierarchies for PROGRAM and USER. As different users have different ISA relationships between domain concepts in mind, there can be several fuzzy set hierarchies for an attribute domain. Thus, we postfix"_number" to each fuzzy set hierarchy name to denote that it is chosen among several available ones.

quantity of information we can obtain from each generalized record may be different.

In the previous example, ⟨editor, programmer⟩ seems more informative than ⟨engineering, programmer⟩. The reason is that since a fuzzy set editor is more specific than a fuzzy set engineering, we have less uncertainty to figure out the original collection of database records from ⟨editor, programmer⟩ than from ⟨engineering, programmer⟩.

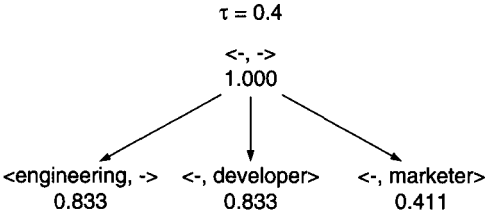


FIGURE 7 The first specialization of the root generalized record.

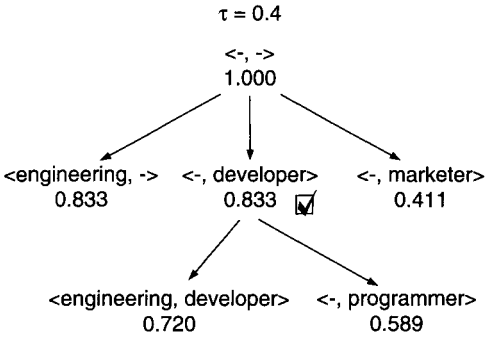


FIGURE 8 The second specialization of the generalized records.

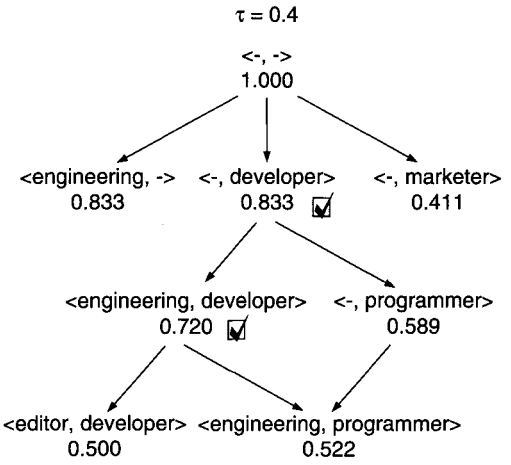


FIGURE 9 The third specialization of the generalized records.

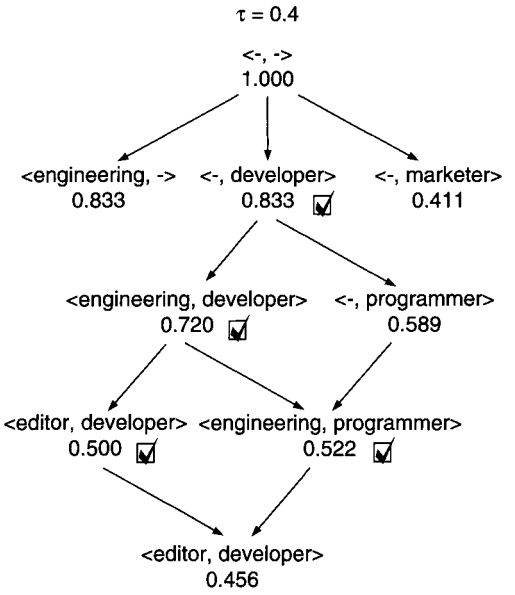


FIGURE 10 The final hierarchy of generalized records.

Along with the specificity of generalized records, support degrees also affect the information values. For example, ⟨editor, programmer⟩ with the support degree 0.9 could be regarded as more informative than the same generalized record with the support degree 0.3. It is because the former seems to give information about 90% of the original collection of database records, while the latter seems to give information about only 30% of them. Strictly speaking, this argument alone may mislead. Both the specificity and the support degree should be considered simultaneously to obtain more exact information value as detailed in the following sections.

a. The Notion of Shannon's Entropy

In Shannon's information theory, the amount of information carried by a message is measured *as* the amount of uncertainty reduced by the existence or the occurrence of that message [24]. When there are n possible alternatives in the system on hand, Shannon's entropy measures the amount of uncertainty of the system as $\log_2 n$. If a message arrives, the number of alternatives might be reduced to m ($m \leq n$) owing to the information carried by the message. Then the amount of uncertainty is reduced to $\log_2 m$. In this situation, we say that the message reduces uncertainty; in other words, it delivers information as much as $\log_2 n - \log_2 m = \log_2 n/m$. This notion of Shannon's entropy is adopted to analyze how much information content a generalized record delivers.

Let us denote the set of all possible data collections on a given attribute scheme as $\Omega(\bullet)$, and a set of data collections on the same attribute scheme that make it possible for a qualified generalized record g to be discovered as $\Omega(g)(\Omega(g) \subseteq \Omega(\bullet))$.

Then the amount of information that the generalized record g delivers is $\log_2\{|\Omega(\bullet)|/|\Omega(g)|\}$, where $|A|$ denotes the cardinality of a set A.

b. An Informativeness Measure for Generalized Records

Prior to deriving formulae for $\Omega(\bullet)$, $\Omega(g)$, and in turn $\log_2\{|\Omega(\bullet)|/|\Omega(g)|\}$, let us observe some characteristics of database summarization. Firstly, a data collection C for summarization can have *duplicates*. It holds even for relational database systems whose underlying data model represents a data collection as a relation. To justify this argument, let us consider an example. Suppose that a computer log relation has an attribute scheme such as ⟨PROGRAM, USER, TTY, DURATION, START_TIME⟩ as in the /var/adm/pacct file in the UNIX system and users want to obtain summaries on which users have executed which computer programs. They must project the relation onto those two attributes PROGRAM and USER before summarization. If duplicate records are eliminated after projection, unfit results may be obtained since several numbers of executions of a program by the same user look like just one execution in the duplicate eliminated table. Most of current relational database languages such as SQL permit duplicates in data tables [25].

Secondly, we assume that attribute dependencies such as functional dependencies and multivalued dependencies are not known in advance. Finally, we suppose that the cardinality or the range of each attribute domain is known. Note that the *support set* [20] of the fuzzy set ω on an attribute domain is equivalent to the attribute domain itself.

Now, we explain how to obtain $|\Omega(\bullet)|$ and $|\Omega(g)|$ in our context. Suppose that a given data collection C has an attribute scheme $(A_1, \ldots, A_m)$. Let $\Psi[j]$ denote the domain of the jth attribute. Then the set of all possible records that can be composed from the domains, denoted as Ψ, becomes $\Psi[1] \times \ldots \times \Psi[m]$, under the ignorance assumption of attribute dependencies.
Consequently,

$$|\Omega(\bullet)| = |\Psi|^{|C|}, \tag{1}$$

since duplicates in the data collection C are allowed.

With respect to a generalized record g, the given data collection C can be thought to being divided into two parts, denoted as C_g and $C_{g'}$. C_g is a set of database records supporting g, and $C_{g'}$ is its complement. Recall that the support degree implies the fraction of supporting database records to the total data collection, i.e., $|C_g|/|C|$. Thus,

$$|C_g| = |C|SD(g), |C_{g'}| = |C|(1\text{-}SD(g)). \tag{2}$$

Also, Ψ can be thought as being divided into two parts denoted as Ψ_g and $\Psi_{g'}$, with respect to g. Ψ_g is a set of records consistent with a generalized record g, and $\Psi_{g'} = \Psi - \Psi_g$. If we define the coverage degree, CV, of a generalized record g as $|\Psi_g|/|\Psi|$, to measure the fraction of Ψ that is consistent with g, Ψ_g and $\Psi_{g'}$ can be written as

$$|\Psi_g| = |\Psi|CV(g), |\Psi_{g'}| = |\Psi|(1 - CV(g)), \tag{3}$$

The coverage degree will be precisely defined in the next subsection. At the moment, let us assume that it is given from somewhere. It is obvious that the coverage degree of a generalized record is, by definition, greater than 0. Complete coverage degree, i.e., one, of a generalized record, implies that the generalized record is $\langle \omega, \ldots, \omega \rangle$. By definition, $\Omega(\langle \omega, \ldots, \omega \rangle)$ is the same as $\Omega(\bullet)$.

As depicted in Fig. 11, database records in C_g and $C_{g'}$ are thought to be selected from Ψ_g and $\Psi_{g'}$, respectively. Thus, we can formulate $|\Omega(g)|$ by (2)

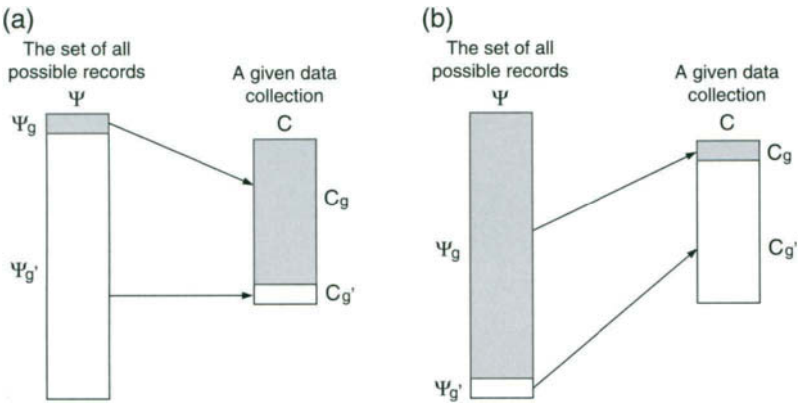


FIGURE 11 Record selection when a qualified generalized record g is known: (a) denotes a case where the coverage degree of g is very low but the support degree is high (low CV and high SD) and (b) denotes the opposite case (i.e., high CV and low SD).

and (3) as

$$|\Omega(g)| = |\Psi_g|^{|Cg|}|\Psi_{g'}|^{|Cg'|}$$
$$= |\Psi|CV(g)^{|C|SD(g)}|\Psi|(1 - CV(g))^{|C|(1-SD(g)}, \quad (4)$$

when $0 < CV(g) < 1$. If $CV(g) = 1$ then $|\Omega(g)| = |\Omega(\bullet)|$.

As a result, the information content of a generalized record g is formulated as

$$INFO(g) = \log_2\{|\Omega(\bullet)|/|\Omega(g)|\}$$
$$= \log_2[|\Psi|^{|C|}/\{|\Psi|CV(g)^{|C|SD(g)}|\Psi|(1 - CV(g))^{|C|(1-SD(g)}\}]$$

by (1) and (4)

$$= \log_2[1/\{CV(g)^{|C|SD(g)}(1 - CV(g))^{|C|(1-SD(g)}\}], \quad (5)$$

when $0 < CV(g) < 1$. If $CV(g) = 1$ then $INFO(g) = \log_2 1 = 0$.

Let us observe the behavior of *INFO*. Figure 12 depicts the values of *INFO* with respect to *SD* and *CV*. There are two areas where informativeness becomes high. One area include cases where *CV* is low and *SD* is high (see the right peak in Fig. 12). It shows that a specific generalized record with a high support degree delivers much information, because low *CV* and high *SD* implies that most records (i.e., $|C|SD(g)$) in the original data collection C come from a small number of alternatives covered by the generalized record (i.e., $|\Psi|CV(g)$): see Fig. 11a.

The other area includes cases where *CV* is high and *SD* is low (see the left peak in Fig. 12). It shows that although the support degree is low, a generalized record can deliver much information if it is very nonspecific. As Fig. 11b shows, it is also informative since the fact of high *CV* and low *SD* implies that most records (i.e., $|C|(1 - SD(g))$ in the data collection C come from a small number of alternatives not covered by the generalized record (i.e., $|\Psi|(1 - CV(g))$).

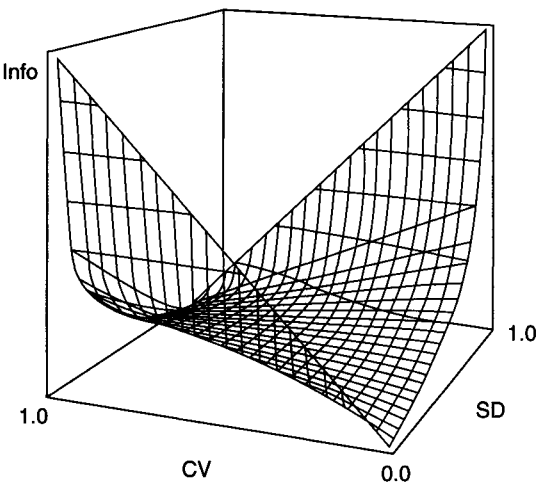


FIGURE 12 Informativeness for various SD–CV combinations.

c. The Coverage Degrees of Generalized Records

So far, we have assumed that the coverage degree of a generalized record is given from somewhere. Now let us consider how to actually obtain the coverage degree of a generalized record g.

Definition 4 (Coverage Degree). The *coverage degree* of a generalized record $g = \langle f_1, \ldots, f_m \rangle$ on an attribute scheme $(A_1, \ldots, A_m)$ is defined as

$$CV(g) = \left\{ \sum_{x1 \in \Psi 1, \ldots, xm \in \Psi m} \otimes[\mu_{f1}(x_1), \ldots, \mu_{fm}(x_m)] \right\} \Big/ \{|\Psi_1| \times \ldots \times |\Psi_m|\}.$$

Note that by substituting $\sum$ operations by $\int$ operations, the formula can be adapted to the case where continuous domains of attributes are involved. Let us see an example of the coverage degree computation: Suppose that we are considering two generalized records on an attribute scheme having two attributes, and attribute domains are $\{a, b, c\}$ and $\{\alpha, \beta\}$ for the first and second attributes. Those two generalized records are given as

$$g_1 = \langle f_{11}, f_{12} \rangle, \text{ where } f_{11} = \{0.1/a, 1.0/b, 0.5/c\}, \; f_{12} = \{0.4/\alpha, 1.0/\beta\},$$

$$g_2 = \langle f_{21}, f_{22} \rangle, \text{ where } f_{21} = \{0.1/a, 1.0/b, 0.0/c\}, \; f_{22} = \{0.1/\alpha, 1.0/\beta\}.$$

Then the coverage degrees of those generalized records are computed as follows:

$$|\Psi_1| \times |\Psi_2| = |\{a, b, c\}| \times |\{\alpha, \beta\}| = 3 \times 2 = 6 \tag{6}$$

$$\begin{aligned}
&\sum_{x1 \in \Psi 1, x2 \in \Psi 2} \otimes[\mu_{f11}(x1), \mu_{f12}(x2)] \\
&= \otimes[\mu_{f11}(a), \mu_{f12}(\alpha)] + \otimes[\mu_{f11}(a), \mu_{f12}(\beta)] + \otimes[\mu_{f11}(b), \mu_{f12}(\alpha)] \\
&\quad + \otimes[\mu_{f11}(b), \mu_{f12}(\beta)] + \otimes[\mu_{f11}(c), \mu_{f12}(\alpha)] + \otimes[\mu_{f11}(c), \mu_{f12}(\beta)] \\
&= \otimes[0.1, 0.4] + \otimes[0.1, 1.0] + \otimes[1.0, 0.4] \\
&\quad + \otimes[1.0, 1.0] + \otimes[0.5, 0.4] + \otimes[0.5, 1.0] \\
&= 0.1 + 0.1 + 0.4 + 1.0 + 0.4 + 0.5 \\
&= 2.5.
\end{aligned} \tag{7}$$

Similarly,

$$\sum_{x1 \in \Psi 1, x2 \in \Psi 2} \otimes[\mu_{f21}(x1), \mu_{f22}(x2)] = 1.3. \tag{8}$$

By (6) and (7), $CV(g_1) = 2.5/6 = 0.42$, and by (6) and (8), $CV(g_2) = 1.3/6 = 0.22$. As a result, we can say that g_1 and g_2 cover the Ψ as much as 42 and 22%, respectively. Figure 13 depicts how those generalized records cover Ψ graphically.

B. Bottom-Up Data Characterization

DBLEARN adopts an attribute-oriented induction method to extract database summaries [9]. In its attribute-oriented induction, each attribute value of a

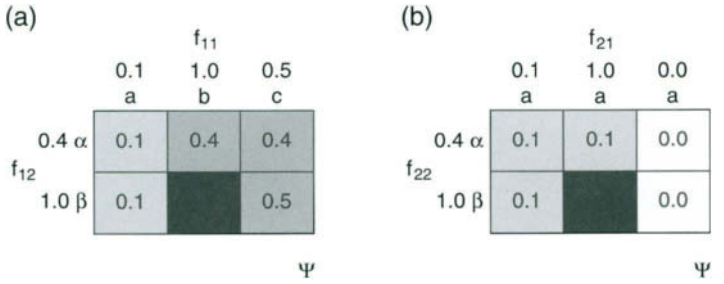


FIGURE 13 Coverage of generalized records: (a) is for $g_1 = \{f_{11}, f_{12}\}$, where $f_{11} = \{0.1/a, 1.0/b, 0.5/c\}$ and $f_{12} = \{0.4/\alpha, 1.0/\beta\}$. (b) is for $g_2 = \{f_{21}, f_{22}\}$, where $f_{21} = \{0.1/a, 1.0/b, 0.0/c\}$ and $f_{22} = \{0.1/\alpha, 1.0/\beta\}$. In each diagram, the whole six-block area represents Ψ_g, and the shaded area represents Ψ_{g1} (or Ψ_{g2}). The density of shade implies how completely the generalized record covers the corresponding area.

record is substituted with a more general concept. After one pass of the substitution, equivalent classes of generalized records are identified and each class is regarded as a candidate summary. This bottom-up procedure is repeated until satisfactory summaries are obtained.

Let us examine the procedure with an example. Suppose that we have a data table as in Table 7. Note that the last attribute, "Vote," is augmented to denote how many records are represented by the record. Initially, it is given as one for each record since only one record is represented by itself. In the process of summarization, the "Vote" value increases.

Also suppose that we have ISA hierarchies on the domains, "Major," "Birth Place," and "GPA" as in Fig. 14. Firstly, we remove the key attributes because it is meaningless to summarize such attributes. In Table 7, the first attribute, "Name," is removed since it is the key of the table. Each attribute value in the remaining table is substituted for its direct generalization in the given ISA hierarchies. This is called attribute-oriented substitution. After the first attribute-oriented substitution, we obtain the results in Table 8.

We can see the first and the fourth records are equivalent to each other. The second record is also the same as the third. Thus, each pair is merged into a single record as in Table 9.

Note that the "Vote" values for the first and second records in Table 9 are increased to 2. They represent that each of them represent two records in

TABLE 7 A Data Table, "STUDENT," To Be Summarized

Name	Major	Birth place	GPA	Vote
Lee	Music	Kwangju	3.4	1
Kim	Physics	Soonchun	3.9	1
Yoon	Math	Mokpo	3.7	1
Park	Painting	Yeosu	3.4	1
Choi	Computing	Taegu	3.8	1
Hong	Statistics	Suwon	3.2	1

Note. Each record represents a brief profile of a student.

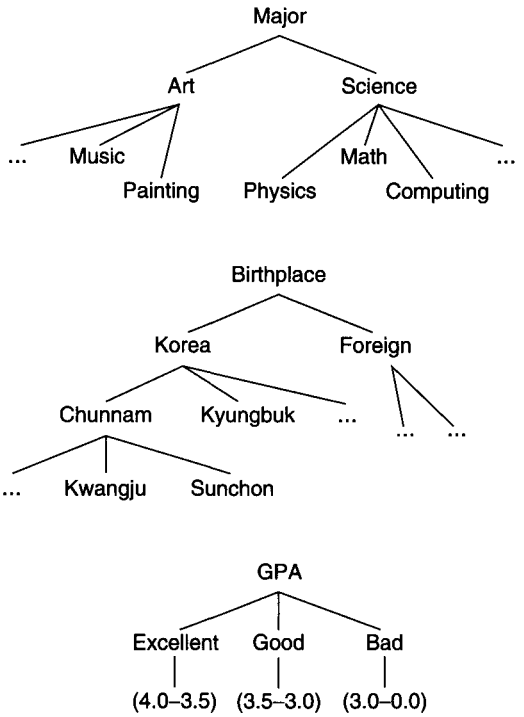


FIGURE 14 ISA hierarchies on the "Major," "Birth Place," and "GPA."

the original table. Commonly, users are interested only in records with high "Vote" values, since they represent the significant portions of the data table. Thus, the user must determine the acceptable minimum for the "Vote" values as the threshold value. If we determine the threshold value as 2 in this small example, the last two records should be removed from the result as in Table 10.

Since the data table contains small number of records and each ISA hierarchy has a few levels for simple illustration, there seldom remain more chances to summarize further. However, this substitution–merge process will be repeated several times in real situations where the data table has a large number of records and each ISA hierarchy has a significant number of levels.

TABLE 8 The Data Table, "STUDENT," after the First Application of Attribute-Oriented Substitution

Major	Birth place	GPA	Vote
Art	Chonnam	Good	1
Science	Chonnam	Excellent	1
Science	Chonnam	Excellent	1
Art	Chonnam	Good	1
Science	Kyungbuk	Excellent	1
Science	Kyunggi	Good	1

TABLE 9 The Data Table, "STUDENT," after Merging Redundant Records

Major	Birth place	GPA	Vote
Art	Chonnam	Good	2
Science	Chonnam	Excellent	2
Science	Kyungbuk	Excellent	1
Science	Kyunggi	Good	1

C. Comparison of Top-Down and Bottom-Up Approaches

This section has been devoted to introduce top-down and bottom-up approaches for data characterization. In the classification of inductive learning paradigms, the top-down approach belongs to the "model-driven generate-and-test" approach. Fuzzy set hierarchies given as the domain knowledge guide the learning process. The process generates hypothetical summaries and tests their validity over the given instance space. In addition, users can easily control the selection of search paths as well as the search depth. In actual database applications, where a great number of possible search paths exists, such kind of path control is quite useful.

Meanwhile, the bottom-up approach also has its own advantages. First of all, the size of data to be scanned is reduced as the characterization proceeds. Initially, the whole data records must be scanned. After each substitution–merge operation, several records are merged into a single reord, and some merged records are removed according to the given threshold value. Even though the size of original data set is very large, the size is reduced significantly after several steps of substitution–merge operations.

IV. CLASSIFICATION TECHNIQUES

Even though classification has been the well-known problem in machine learning, there still remain many areas to investigate in the context of data mining. It is due to the differences that the target database has much more records than the training set for machine learning algorithms, and the database also contains erroneous data and missing values.

Suppose that a marketing manager of a company is planning a direct mailing program to promote the sales of a new product. The company has already

TABLE 10 The Data Table, "STUDENT," after Removing Records with Low "Vote" Values

Major	Birth place	GPA	Vote
Art	Chonnam	Good	2
Science	Chonnam	Excellent	2

constructed the customer database containing the properties of each customer, such as address, gender, age, and job. It has been reported that the response ratio of direct mailing falls below 5% in most cases. It means that less than 5,000 customers purchase the new product in response to the direct mail when the company delivers 100,000 brochures for advertisement. If the expense of direct mailing is $1 per each customer, the total cost becomes $100,000. Thus, the marginal profit of each unit of sale must be over $200 for the direct mailing program to be regarded as beneficial since $100,000 ÷ 5,000 customers = $200/customer. Note that we do not consider extra expenses such as product delivery fee and sales employee salary for the sake of simplicity.

If the marketing manager can distinguish the customers who are likely to respond to the advertisement, the situation becomes quite different. As an extreme, if the manager can exactly identify in advance the 5,000 customers to purchase the product, it is still beneficial as long as the marginal profit of each unit of sale in the previous example is more than only $1. If the manager can identity the customers to purchase the product with 50% precision, he has to deliver only 10,000 brochures for advertisement for 5,000 units of sales. In this case, it is still beneficial as long as the marginal profit of each unit of sale is more than two dollars.

Classification techniques can remedy this situation. It means the activity of finding common properties of a given group of entities that distinguish them from other groups. In the previous example, the manager can determine through classification techniques common properties of those customers likely to purchase the product in terms of their addresses, gender, ages, jobs, and so on. Then he/she can choose the addresses of such customers from the whole customer database and deliver the advertisement brochures to them.

In the context of classification, a record consists of multiple descriptive attributes and its class label. The descriptive attributes represent several properties of each record, while the class label represents the class to which the record belongs. We need to have a training set consisting of preclassified records to build a specific classification facility. By being preclassified, we mean that the class label as well as descriptive attributes are already known. Once a classification facility is constructed, classification implies determining the class label of a new record that has only descriptive attributes. The goal is to devise a classification facility that will allow us effective and efficient classification. Effectiveness means how correctly the method classifies new records, and efficiency means how less classification cost is.

There have been several techniques for classification such as decision tree induction, artificial neural networks, and genetic algorithms. Each has its own relative advantages over and disadvantages against the others. Thus, it is important to understand them, and adopt proper techniques according to the application requirements.

A. Decision Tree Induction

A decision tree is a tree structure representing classification rules. Suppose that a database is given as in Table 11. The attributes A1, A2, and A3 of each record describe certain properties of each entity, and the final attribute C represents

TABLE 11 An Example Database for Classification

A1	A2	A3	C
a	d	k	1
a	e	r	2
b	f	m	3
b	g	o	3

the class label. From the table, a decision tree can be induced as in Fig. 15. Each node represented as a rectangle is called a decision node. The terminal nodes are called result nodes. The classification process begins from the root node. According to the decision in the decision node, the corresponding edge is followed to the next decision node. When the edge traversal reaches a terminal node, the classification result is obtained. For example, suppose that we have a new record ⟨a, e, k⟩ and that the decision tree classifies it into a class "2."

Note that given a database, there can be several decision trees. Thus, the issue of decision tree induction is how to construct an effective and efficient decision tree. Effectiveness means how correctly the decision tree classifies new records, and efficiency means how less the representation cost of the decision tree is. There are a variety of algorithms for constructing decision trees. Two of the most popular go by the acronyms CART and CHAID, which stand for "classification and regression trees" and "chi-squared automatic interaction detection," respectively. A newer algorithm C4.5 is gaining popularity and is now available in several software packages.

CART constructs a binary tree by splitting the records at each node according to a function of a single input field. The first task, therefore, is to decide which of the descriptive attributes makes the best splitter. The measure used to evaluate a potential splitter is "diversity." There are several ways of calculating the index of diversity for a set of records. With all of them, a high index of diversity indicates that the set contains an even distribution of classes, while a low index means that members of a single class predominate. The best splitter is the one that decreases the diversity of the record sets by the greatest amount.

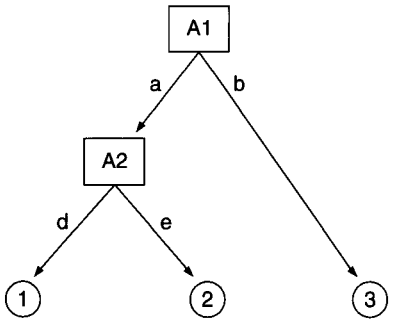


FIGURE 15 An example decision tree.

In other words, we want to maximize the value of the expression

$$\text{diversity(the original set)} - (\text{diversity(the set in the left child)} + \text{diversity(the set in the right child)}).$$

As the diversity function, several functions combining probability functions, logarithms, minimum operations, and so on have been introduced [4].

C4.5 is a descendent of the well-known classical decision tree induction method, ID3, which stands for "Iterative Dichotomiser 3" proposed by Quinlan and Rivest [10]. The primary difference between C4.5 and CART is that the former constructs m-ary trees rather than binary trees. Especially, when an attribute on a categorical domain is chosen as a splitter, C4.5 makes m-ary branches for them. The advantage of it is highly dependent on the domain of classification.

CHAID is the most widely used since it is distributed as part of popular statistics packages like SPSS and SAS. The largest difference between CHAID and the two decision tree induction algorithms that we have mentioned is that it attempts to stop growing the tree before overfitting occurs, rather than first overfitting the data, then pruning. Another difference is that CHAID is restricted to categorical variables. Continuous variables must be broken into ranges or replaced with classes such as high, low, medim [4].

The attractiveness of tree-based methods for classification is due in large part to the fact that, in contrast to artificial neural networks, decision trees represent rules in human-understandable form [4].

B. Artificial Neural Network-Based Classification

Artificial neural networks are computer programs whose behaviors imitate the human brain. It is known that the human brain consists of a great number of neurons [26, 27]. Transmission of electric signals among neurons causes complex brain activities such as decision and recognition. Artificial neural networks are also composed of artificial neurons that are actually parts of programs. Figure 16 depict a structure of an artificial neuron. Each input has its own weight. The left part of the neuron denoted as "$\sum$" is a combination function, which combines all the inputs into a single value. The weighted summation is commonly used as a combination function. The right part of the neuron denoted as "$\int$" is a transfer function, which calculates the output value from the result of the combination function. The output is exactly one value usually taken from 0 to 1.

Neurons are connected together to construct an artificial neural network. Figure 17 illustrates some types of artificial neural networks. An artificial neural

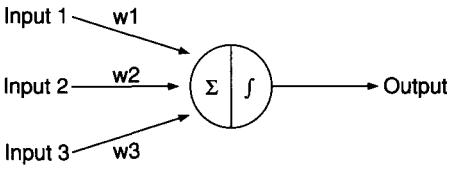


FIGURE 16 Structure of an artificial neuron.

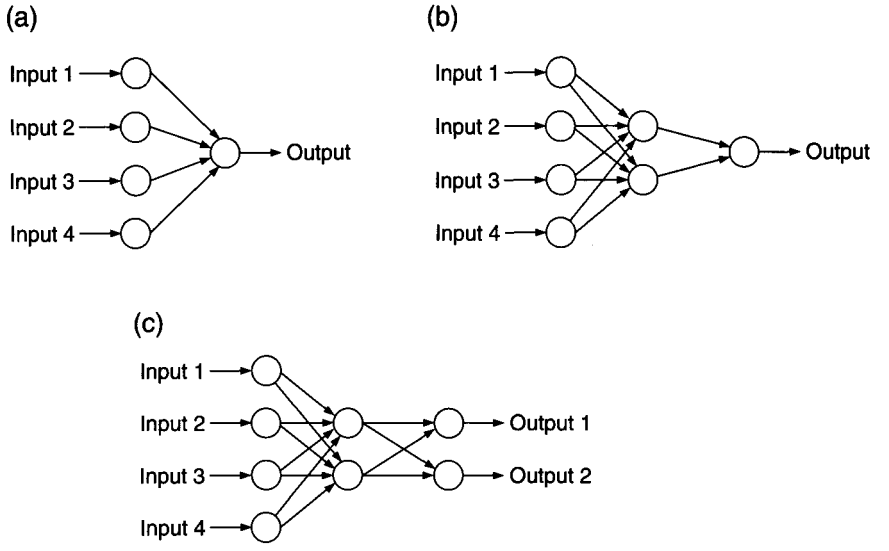


FIGURE 17 Several types of artificial neural networks: (a) a simple network, (b) a network with a hidden layer, and (c) a network with multiple outputs.

network commonly has three layers, input layer, ouput layer, and hidden layer. The first type (a) is a very simple neural network with only input and output layers. The second type (b) also has a hidden layer. The hidden layer makes the network more powerful by enabling it to recognize more patterns. The third type (c) can produce multiple output values.

Classification with artificial neural networks consists of the following steps. Firstly, we must identify the input and output features of records. Even though descriptive attributes become input features and the class label becomes the output feature, we must encode them so their range is between 0 and 1. In the second step, we must set up a network with an appropriate topology. For example, the number of neurons in the hidden layer must be determined. In the third step, it is required to train the network on a representative set of training examples. In the fourth step, it is common to test the performance of the network with a set of test data. Note that the test data must be strictly independent from the previous training data. If the result of the test is not acceptable, we must adjust the network topology and parameters. Lastly, we apply the network to classify new records.

The training step is one of the most important parts of neural network-based classification. Actually, it is the process of setting the best weights on the inputs of each neuron. So far the most common technique is back-propagation originally developed by Hopfield [26]. In back-propagation, after the network calculates the output using the existing weights in the network, it calculates the error, i.e., the difference between the calculated result and the expected. The error is fed back through the network and the weights are adjusted to minimize the error.

As compared to decision tree-based classification, artificial neural networks are reported much more powerfully especially in complicated domains. They

incorporate nonlinear combinations of features into their results, not limiting themselves to rectangular regions of the solution space. In addition, it is also an advantage that artificial neural networks are already available in many off-the-shelf software packages.

However, the primary shortcoming of artificial neural networks is that they cannot explain results in a human-understandable way. The internal behavior of a network is regarded as a black box, and human users can examine only input and the corresponding output values. Furthermore, it is also regarded as the limitation that all inputs and outputs must be encoded as numbers between 0 and 1. This requires additional transforms and manipulations of the input data.

V. ASSOCIATION RULE DISCOVERY

An association rule implies the likelihood of co-occurrence of several events. Since it was first introduced [15], there have been many active research efforts in a variety of applications such as supermarket sales analysis, telephone network diagnosis, and Internet browsing pattern analysis [18].

A. Definition of the Association Rule

Given a set of events $I = \{i_1, i_2, \ldots, i_m\}$, an association rule $X \Rightarrow Y$ is defined, where X and Y are subsets of I. It is interpreted as "when events in X occur, events in Y are likely to occur simultaneously." When we have a transaction database, where each record, i.e., transaction, represents a set of events occurring simultaneously, we can evaluate the validity of a certain association rule twofold. Firstly, the support degree of an association rule, $X \Rightarrow Y$, indicates how much portion of the database contains the set of events in $X \cup Y$. In other words, it represents the statistical significance of the rule. Secondly, the confidence of an association rule, $X \Rightarrow Y$, indicates how many records also contain the set of events in Y among the records containing the set of events in X. It implies the strength of the rule. The goal is to determine association rules with high support degree and confidence with respect to a given transaction database.

B. Association Rule Discovery Algorithms

The first algorithm for association rule discovery is called Apriori [15]. Although there have been many variations of the algorithm to reduce the execution time or to enhance the functionality, we introduce Apriori herein since it shows the fundamental issues of association rule discovery effectively. Suppose that we have a transaction database as in Table 12.

The discovery of association rules is composed of two tasks. The first task is to determine the sets of events with high support degrees. We call those event sets with support degrees higher than a given threshold value frequent event sets. Once we have the frequent event sets, it is straightforward to elicit association rules with high confidence from them. Thus, most efforts for association rule discovery have focused on the first step.

TABLE 12 An Example Transaction Database for Association Rule Discovery

Transaction ID	Events
101	A, C, D
102	B, C, E
103	A, B, C, E
104	B, E

Figure 18 depicts the first task on the database in Table 12 step by step. Suppose that the given threshold value for support degree is 40%. Firstly, by scanning the whole database, we produce event sets, each of which contains a single event. The result becomes C1. The attribute label as "Support" indicates how many transactions in the original database contain the corresponding event. Since the number of transactions in the database is four and the threshold value is 40%, the event set should have a "Support" of value more than 1. Thus, the fourth event set is removed as in L1. By combining event sets in L1, we produce event sets, each of which contains exactly two events. The result becomes C2. We count how many transactions in the database contain each event set by scanning the whole database again. The only event sets with a

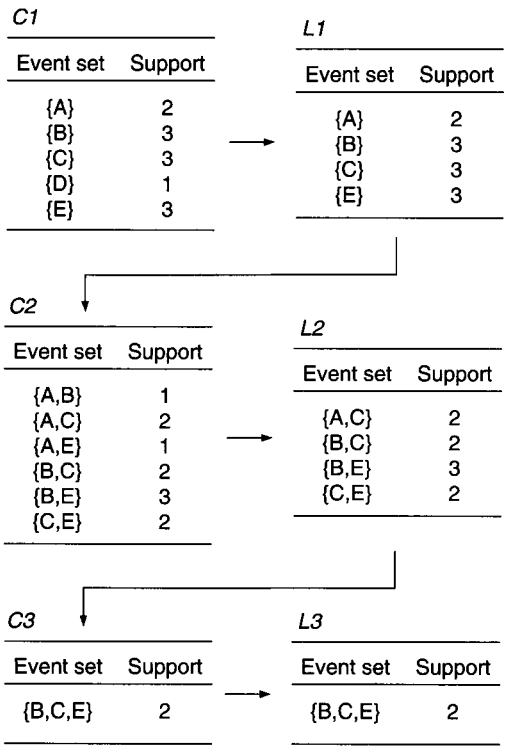


FIGURE 18 Step-by-step process of determining frequent event sets.

"Support" value more than 1 are preserved as in L2. Again, we produce event sets, each of which contains exactly three events by combining event sets in L2. It becomes C3. This process is repeated until no more operations are possible. In this example, we obtain frequent event sets as L1 ∪ L2 ∪ L3.

Let us examine the second task, i.e., eliciting association rules with high confidence in an instance. Since the "Support" values of the event sets {B, C, E} and {B, E} are 2 and 3 respectively, the confidence of the association rule {B, E} => {C} becomes 67%. It means that 67% of event sets also contain the event C among event sets containing B and E.

Several variations of the Apriori algorithm have tried to reduce the execution time by exploiting hashing, database partitioning, and so on [16, 30]. In addition, there are variations for discovering multilevel association rules. For example, it may be unnecessarily detailed to have an association rule such as "if a customer purchases a 160-ml Jones Milk (Product ID: 78-2456), then she is likely to purchase a 200-g Hayes Bread (Product ID: 79-4567)." Instead, it may be more useful just to know that milk customers are apt to also be bread customers. There have been several research efforts to devise effective algorithms to discover such generalized rules as well as detail ones [28–30].

VI. CONCLUDING REMARKS

This chapter has discussed several representative data mining techniques such as data characterization, classification, and association. In addition to the techniques discussed here, there are a lot of useful techniques such as clustering, sequential pattern discovery, and time series search. Likely to the SQL case where application demands lead to the development of new features, it is expected that new techniques will be continuously introduced according to the application requirements.

There have been many research projects and experimental or commercial systems for data mining. The QUEST project at IBM Almaden Research Center has investigated various data mining techniques including association rules [15]. IBM recently announced and began to deliver its data mining system, Intelligent Miner, whose theoretical background is strongly based on the QUEST project. Bell Laboratory has also developed IMACS (Intelligent Market Analysis and Classification System) [31]. The project emphasizes the aspect of human interaction in the data mining process, and suggests the term "data archeology" to express its data mining paradigm. GTE Laboratory has developed KDW (Knowledge Discovery Workbench), which incorporates a variety of data mining techniques in a unified framework [32]. Professor J. Han at Simon Fraser University in Canada has also been devoting active efforts to data mining. Recently, he announced DBMiner, GeoMiner, and WebMiner as a series of data mining systems [33, 34].

Meanwhile, there has come a new wave of information management called "data warehousing" in the information technology recently. Data warehousing is a large-scale project where a large volume of data is extracted from disparate sources of operational databases, and organized in the forms appropriate for enterprise-wide decision making. Figure 19 depicts the architecture of a data

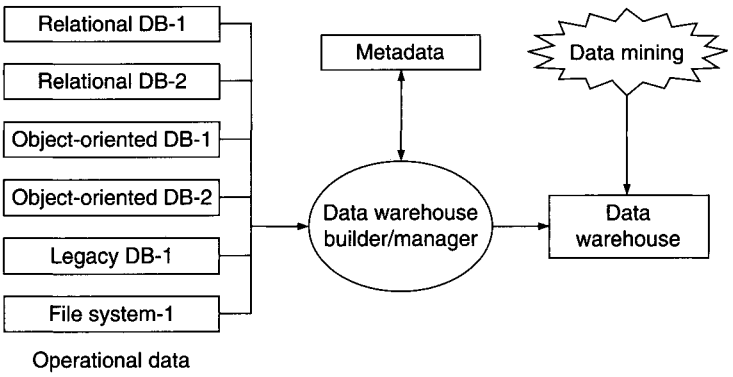


FIGURE 19 Architecture of a data warehousing system.

warehousing system. Since operational databases have been built for their own operational purposes, it is rare that their structures fit together in the enterprise-wide decision making. Furthermore, their schema are quite different to each other and some information is apt to be duplicated. A data warehouse builder must rearrange and adjust all the heterogeneity in the disparate data sources. When the content of a data source changes, it should be propagated to the data warehouse properly. It is under the responsibility of the data warehouse manager. Since Metadata represent the global schema information of the data warehouse, all queries on the data warehouse must refer to Metadata. Additionally, some systems adopt data marts, which are small-scale data warehouses for specific portions of the decision making.

As indicated in Fig. 19, data mining techniques can reveal their strength strongly in data warehousing systems since the data warehouse already contains proper data sources for valuable data mining. Although online analytical processing (OLAP) provides the querying interface for data warehouses currently, the application coverage of it is so limited. Thus, there have been active efforts for integrating data mining techniques and data warehousing systems.

REFERENCES

1. Frawley, W., Piatetsky-Shapiro, G., and Matheus, C. Knowledge discovery in databases: An overview. In *Knowledge Discovery in Databases,* pp. 1–27. AAAI Press, Menlo Park, CA, 1991.
2. Silberschatz, A., Stonebraker, M., and Ullman, J. Database systems: Achievement and opportunities. The Lagunita Report of the NSF Invitational Workshop, TR-90-22, Department of Computer Science, University of Texas at Austin, 1990.
3. Cohen, P., and Feigenbaum, E. *The Handbook of Artificial Intelligence,* Vol. 3, pp. 411–415. Kaufmann, Los Altos, CA, 1982.
4. Berry, M. J. A., and Linoff, G. *Data Mining Techniques: For Marketing, Sales, and Customer Support.* Wiley, New York, 1997.
5. Cabena, P., Hadjinian, P., Stadler, R., Verhees, J., and Zanasi, A. *Discovering Data Mining: From Concept to Implementation.* Prentice-Hall PTR, Englewood Cliffs, NJ, 1998.
6. Lee, D. H., and Kim, M. H. Discovering database summaries through refinements of fuzzy hypotheses. In *Proc. 10th Int'l Conf. on Data Engineering,* pp. 223–230, 1994.

7. Lee, D. H., and Kim, M. H. Database summarization using fuzzy ISA hierarchies. *IEEE Trans. Systems Man Cybernet.* 27(4): 671–680, 1997.
8. Yager, R. On linguistic summaries of data. In *Knowledge Discovery in Database,* pp. 347–363. AAAI Press, Menlo Park, CA, 1991.
9. Han, J., Cai, Y., and Cercone, N. Knowledge discovery in databases: An attribute-oriented approach. In *Proc. of Int'l Conf. on Very Large Databases,* pp. 547–559, 1992.
10. Quinlan, J., and Rivest, R. Inferring decision trees using the minimum description length principle. *Inform. Comput.* 80: 227–248, 1989.
11. Yasdi, R. Learning classification rules from database in the context of knowledge acquisition and representation, *IEEE Trans. Knowledge Data Eng.* 3(3): 293–306, 1991.
12. Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., and Swami, A. An interval classifier for database mining applications. In *Proc. of Int'l Conf. on Very Large Databases,* Vancouver, Aug. 1992, pp. 207–216.
13. Ng, R., and Han, J. Efficient and effective clustering methods for spatial data mining. In *Proc. of Int'l Conf. on Very Large Databases,* pp. 144–155, 1994.
14. Xu, X., Ester, M., Kriegel, H.-P., and Sander, J. A distribution-based clustering algorithm for mining in large spatial databases. In *Proc. of Int'l Conf. on Data Engineering,* pp. 324–333, 1998.
15. Agrawal, R., Imielinski, T., and Swami, A. Mining associations between sets of items in massive databases. In *Proc. of ACM SIGMOD Conf.,* Washington D.C., May 1993.
16. Agrawal, R., and Srikant, R. Fast algorithms for mining association rules in large databases. In *Proc. of Int'l Conf. on Very Large Databases,* Santiago, Sep. 1994, pp. 487–499.
17. Agrawal, R., and Srikant, R. Mining sequential patterns. In *Proc. of Int'l Conf. on Data Engineering,* Taipei, Mar. 1995, pp. 3–14.
18. Chen, M.-S., Park, J. S., and Yu, P. S. Data mining for path traversal patterns in a web environment. In *Proc. ICDCS,* pp. 385–392, 1997.
19. Zadeh L. Fuzzy Sets. *Inform. Control* 8: 338–353, 1965.
20. Zimmermann, H. *Fuzzy Set Theory and Its Applications.* Kluwer–Nijhoff, Dordrecht, 1985.
21. Klir, G. J., and Folger, T. A. *Fuzzy Sets, Uncertainty, and Information,* pp. 260–265. Prentice-Hall, Englewood Cliffs, NJ, 1988.
22. Zadeh, L. A fuzzy set theoretic interpretation of linguistic hedgese. *J. Cybernet.* 2(2): 4–34, 1972.
23. Ullman, J. *Principle of Databases and Knowledge-Base Systems.* Computer Science Press, New York, 1988.
24. Shannon, C. The mathematical theory of communication. *Bell System Tech. J.* 27: 379–423, 623–656, 1948.
25. ISO 9075: Information Processing Systems–Database Language SQL, 1992.
26. Bigus, J. *Data Mining with Neural Networks.* McGraw-Hill, New York, 1996.
27. Lu, H., Setiono, R., and Liu, H. NeuroRule: A connectionist approach to data mining. In *Proc. of Int'l Conf. on Very Large Databases,* Zurich, Sep. 1995, pp. 478–489.
28. Srikant, R., and Agrawal, R. Mining generalized association rules. In *Proc. of Int'l Conf. on Very Large Databases,* Zurich, Sep. 1995, pp. 407–419.
29. Han, J., and Fu, Y. Discovery of multiple-level association rules from large databases, In *Proc. of Int'l Conf. on Very Large Databases,* Zurich, Sep. 1995, pp. 420–431.
30. Park, J.-S. and Fu, Y. An efficient hash based algorithm for mining association rules In *Proc. of ACM SIGMOD Conf.,* pp. 175–186, 1995.
31. Brachman, R., and Halper, F. Knowledge representation support for data archeology. In *Proc. of Int'l Conference on Information and Knowledge Management,* pp. 457–464, 1992.
32. Piatetsky-Shapiro, G., and Matheus, C. Knowledge discovery workbench for exploring business databases. *Int. J. Intell. Syst.* 7(7): 675–686, 1992.
33. Han, J., *et al.,* DBMiner: A system for mining knowledge in large relational databases. In *Proc. of Int'l Conf. on Knowledge Discovery in Databases,* 1996.
34. Han, J. *et al.,* GEOMiner: A system prototype for spatial data mining. In *Proc. of ACM SIGMOD Conf,* 1997.

3

OBJECT-ORIENTED DATABASE SYSTEMS

HIROSHI ISHIKAWA

Department of Electronics and Information Engineering, Tokyo Metropolitan University, Tokyo 192-0397, Japan

I. INTRODUCTION

Initially, complex, large-scale database applications such as CAD [15], hypermedia [14], and AI [11, 12] spawned object-oriented databases (OODBs). Moreover, OODBs have gone beyond them toward advanced applications such as networked multimedia applications [24]. In this paper, we describe how we designed and implemented an object-oriented DBMS called Jasmine [13, 15, 16, 18, 23]. (Note that Jasmine throughout this paper is not a product name but a prototype code name of Fujitsu Laboratories Ltd.) We also discuss how we applied Jasmine to advanced multimedia applications to verify its basic validity and how we extended Jasmine for such applications.

This paper has the following contributions. First, we focus on the impact of the design of its object-oriented model and language on database implementation technology. We describe what part of traditional relational database technology we extend to handle object-oriented features such as object identifiers, complex objects, class hierarchies, and methods. We introduce nested relations to efficiently store and access clustered complex objects. We use

Database and Data Communication Network Systems, Vol. 1

hash-based methods to efficiently access nonclustered complex objects. We provide user-defined functions directly evaluated on page buffers to efficiently process method invocation. We devise object-oriented optimization of queries including class hierarchies, complex objects, and method invocation. We incorporate dedicated object buffering to allow efficient access to objects through object identifiers. Second, we describe nontrivial applications of Jasmine in detail and discuss the validity of object-oriented databases. We focus on networked multimedia databases, which can be basically implemented by taking advantage of the extensibility of Jasmine. Further, we focus on how we extend Jasmine to implement its advanced facilities such as real time video play.

This paper is organized as follows. Section II describes the object model and the object manipulation language of Jasmine. Section III describes the implementation of Jasmine. Section IV discusses an object-oriented database approach to networked multimedia applications. Section V compares our system with other related work and gives concluding remarks.

II. FUNCTIONALITY

A. Data Model

In this section, we will briefly describe Jasmine's object model. (See [18] for the formal semantics.) Objects are a collection of attributes, which are categorized into properties (enumerated attributes) and methods (procedural attributes). Properties are object structures and methods are operations on objects. Objects are categorized into instances and classes. Instances denote individual data and classes denote types (i.e., structures) and operations applicable to instances of the class. Instances consist of a collection of attribute names and values. Classes consist of attribute names, definitions, and associated information such as demons. Objects are uniquely identified by values of the system-defined attribute object identifier (OID). On the other hand, values such as numbers and character strings have no OIDs but do have classes defined by the system. Objects with OIDs are called reference objects and values with no OIDs are called immediate objects. Objects can include other objects (i.e., OIDs) as attribute values. This enables the user to directly define complex objects (composite objects) [26], which supports aggregation directly.

Classes are organized into a hierarchy (more strictly, a lattice) by generalization relationships. This hierarchy is called a class hierarchy. Classes (i.e., subclasses) can inherit attribute definitions from their superclasses. The user can make instances (i.e., instantiate) from any class in a class hierarchy unlike Smalltalk-80 [9]. Such instances are called intrinsic instances of the class. Classes not only define object types and methods, but they are also interpreted as a set of instances, which supports classification. That is, the instances of a class is the union of all the intrinsic instances of itself and all its subclasses. This differentiates Jasmine from other OODBs such as GemStone [32] where the user must define separate classes both as type and as a set. Objects can have a set of objects as well as a singleton object as an attribute value. The former are called multiple-valued attributes and the latter single-valued attributes.

Specialized functions called demons can be attached to attributes. Constraint demons are checked before values are inserted into attributes. The values are only inserted if the demon returns true. If-needed, if-added, if-removed, and if-updated demons are invoked when values are referenced, inserted, deleted, and replaced. Before and after demons are invoked before and after the procedural attributes they are attached to are invoked. The user can combine these demons to flexibly implement active databases [19, 33]. Unlike other systems, Jasmine allows the user both to specify system-defined integrity constraints, such as mandatory and multiple, and to specify user-defined integrity constraints as demons.

Consider the class PATIENT as an example (see Fig. 1). The keyword Enumerated is followed by the definition of user-supplied enumerated attributes. The name facet such as Doctor denotes the name of an attribute. The class facet before the attribute name denotes the range class such as FLOAT of Height. The value of the attribute of an instance must be an instance of the range class (see Fig. 2). The domain of the attribute is the class being defined, PATIENT. The multiple facet denotes that the attribute is multiple-valued such

```
PATIENT
   Db              MEDICAL
   Super           PERSON
   Enumerated      DOCTOR     Doctor mandatory
                              STRING Category default  outpatient
                              INTEGER Cardinality common
                              FLOAT Temperature multiple
                                    constraint {(value > 34.0 && value <43.0)}
                              FLOAT Weight mandatory constraint {(value > 0.0)}
                              FLOAT Height constraint {(value > 0.0)}
                        If-needed
                          { int h;
                            h = self.Weight;
                            return h + 100.0;}
   Procedural      MEDICAL_CERTIFICATE make-medical-certificate (date)
        STRING date;
       { MEDICAL_CERTIFICATE  mc;
                 mc = <MEDICAL_CERTIFICATE>.instantiate ();
         mc.Patientname = self.Name;
         mc.Doctorname = self.Doctor.Name;
         mc.Diseasename = self.Disease.name;
         mc.Date = date;
         return mc;}
```

FIGURE 1 Example of class definition.

MedicalPatient007

Sex	male
Age	36
Name	James Bond
Address	Tokyo
Doctor	MedicalDoctor010
Category	inpatient
Temperature	37.5 37.3 38.1
Weight	76.0
Height	176.0

FIGURE 2 Example of an instance.

as Temperature. The mandatory facet denotes that the attribute allows no null value such as Doctor and Weight. The mandatory attribute must be specified its value at instantiation. The common facet denotes that the attribute value is common to all the instances as the domain objects. The common attribute is not necessarily a constant, such as Cardinality. The default facet contains a default value referenced when the attribute value is not yet specified, such as Category of PATIENT. The if-needed demon, invoked if the referenced attribute has a null value, computes a value such as Height of PATIENT. The keyword Procedural is followed by the definition of procedural attributes. Procedural attributes such as make-medical-certificate also have facets. The class facet such as MEDICAL_CERTIFICATE denotes the range class of the procedural result.

A superclass in a class hierarchy is denoted by the system-defined attribute Super. The superclass, for example, PERSON, includes its subclasses, PATIENT, as a set. The attributes of the superclass are inherited to the subclass, such as Age of PATIENT. An attribute can be newly defined in the subclass such as Doctor of PATIENT. Intrinsic instances of a nonleaf class can represent incomplete knowledge of the domain. For example, PERSON intrinsic instances directly denote a set of persons known to be neither a patient nor a doctor.

A class can be divided into disjoint subclasses (see Fig. 3). Those subclasses are collectively called a partition. Each subclass is called a member of

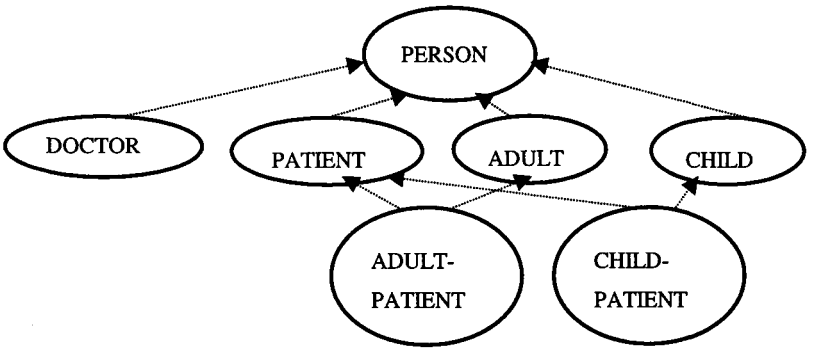


FIGURE 3 Example of a class lattice.

the partition. For example, PERSON has a partition consisting of DOCTOR and PATIENT. A partition denotes a categorization based on one viewpoint. Different viewpoints generate different partitions. PERSON has another partition of ADULT and CHILD. Members of distinctive partitions may not be disjoint, such as PATIENT and ADULT. Categorization conditions can be explicitly specified to make the partition semantics clear such as "Age <18" of CHILD. Then the attribute Age is called a categorization attribute. The categorization conditions are basically specified by a subset of the database programming language described in Section II.B. They can be used by the query optimization described in Section III.

Jasmine allows a class to have multiple superclasses. They must be nondisjoint members of different partitions of a class. For example, ADULT-PATIENT inherits Doctor from PATIENT and Occupation from ADULT (multiple inheritance [35]). Multiple superclasses sometimes have properties of the same name. We must resolve such conflicts. Assuming that the class of a property of a subclass should be either the class of or its subclass of the same property of a superclass, the rules are as follows: (1) A property with the most specific class is chosen if there is only one such property. (2) Otherwise, a property with the most specific class first found by a system-defined search (i.e., depth-first search) is chosen. (3) A property other than that determined by rule (1) or (2) can be chosen by explicitly specifying the superclass in the property definition if necessary.

B. Database Programming Language

This section describes an object manipulation language called Jasmine/C [18] as a database programming language which integrates a general-purpose programming language (C) and a database language in an object-oriented context, and which allows the user to program advanced applications. In Jasmine, the user manipulates objects by sending messages to objects just as in object-oriented programming languages. This type of access is called singleton access or individual access. The user can assign values to attributes and reference attribute values. Jasmine allows set-oriented access in addition to singleton access. Set-oriented access is done by a query on objects. The query language of Jasmine has the following features different from those of SQL [3]. The semantics can be formally defined through query translation by object operators as described in [18].

The basic unit of a query expression consisting of target and condition parts is an object expression, a class name followed by a series of attribute names. The target part is an object expression, or a list of object expressions. The condition part consists of a logical combination of predicates which compare object expressions. A query on a class returns all the instances of the class and its subclasses, so the user can retrieve by a single Jasmine query what would take multiple relational database queries to retrieve. The object expressions denote object joins. The object expressions can also contain methods, so the user can manipulate objects set-theoretically and filter a set of objects procedurally. If a superclass is specified with a method in a query, methods dedicated to instances of the class and its subclasses can be invoked simultaneously. This facilitates polymorphism [35] in a set-oriented manner. The system-defined methods such

as put and delete specified in a query can modify a set of objects. A query can invoke demons which implement integrity facilities introduced by QBE [38]. The user can specify multiple-valued attributes in a query. The user can control unnesting of multiple values and apply aggregate functions correctly. Multiple-valued attributes are existentially or universally quantified.

The integration of query and programming facilities is another important feature for advanced applications. First, the user can specify methods in a query as described above. The user can extend the functionality of the query language just by defining and specifying a method in a query, without modifying the query language processor. The user can develop application programs more compactly without specifying details such as iteration variable declaration and control structures. Making this type of iteration implicit can increase physical data independence [3] of application programs by allowing the system to optimize the query expression. Second, the user can also define methods by specifying a query for them. This can define so-called virtual attributes and increase logical data independence [3] of application programs when applications evolve. Third, the fact that the user invokes a query from programs is one of the salient aspects of advanced applications. We introduce set variables to solve the impedance mismatch problem [3] between the query and programming languages. The set variable has a class defined by an object model as its type and can contain a set of objects returned by a query as its value. The user can fetch an object by sending the scan message to the set variable and operate on the object by sending a message to the object in an object-oriented programming manner.

Class objects can also be operated set-theoretically for advanced applications. Basic database functions such as transactions, locking, and logging can be provided through system-defined classes. Multimedia data types and operations are provided by implementing them from system-defined primitive classes in a bootstrap manner.

Now we describe the syntax and semantics of a query through examples. The query has the syntax

"["object_expression(s)"]" [where condition] [groupby object_expression(s)],

where the object expression has the form class_name ["." attribute_name ["." attribute_name]...].

The query expression evaluates to a set of the target objects satisfying the condition. The elements of the constructed set are objects (OIDs), or values belonging to the database, or newly constructed tuple values. The result type is determined by the query. For example, to find the name and address of outpatients, the user forms a query as

(Query 1) [PATIENT.Name, PATIENT.Address]
where PATIENT.Category == "outpatient".

The tuple operator "[ ]" allows the construction of tuple values, corresponding either to the projection or to the join of relations. Like this example, the query corresponds to projection only if the target list has the form

[common_object_expression.attribute1, common_object_expression.attribute2, ...].

Immediate objects are compared by ==, !=, >, >=, < , and <=, based on values.

In general, joins are categorized into implicit and explicit joins. Jasmine supports implicit joins as

(Query 2) PATIENT.Doctor.Name where PATIENT.Name == "James Bond".

This finds the name of doctors who are in charge of James Bond. The operator "[ ]" can be omitted only if the target list contains only one object expression. Assuming that C is a class and *Ai* is an attribute and *Oi* is an object, an implicit join denoted by an object expression *C.A1*. . . . An has the following semantics:

{ On |O0 belongs to C and for all $i = 1, \ldots, n$, either of the following holds:

(1) *Oi* is equal to *Ai* of *Oi*-1 if *Ai* is single-valued
(2) *Oi* belongs to *Ai* of *Oi*-1 if *Ai* is multiple-valued}.

Nested sets generated by multiple-valued attributes are automatically flattened unless the user prohibits that. Jasmine can also support explicit joins as

(Query 3) [PATIENT.Name, DOCTOR.Name] where
PATIENT.Age == DOCTOR.Age.

This retrieves pairs of names of patients and doctors who happen to be of the same age. "[ ]" in this case corresponds to join. Reference objects can also be compared by == and != based on OIDs. For example, assume Disease and Specialty are reference attributes (see Fig. 4):

(Query 4) [PATIENT.Name, DOCTOR.Name] where
PATIENT.Disease == DOCTOR.Specialty.

This query finds the names of patients and doctors who specialize in their disease.

The object expression with multiple-valued attributes evaluates to a set of sets. However, multiple-valued attributes are automatically unnested unless the user specifies the prohibition of unnesting by a special operator described later. Therefore, the following query retrieves a flattened set of temperatures of serious patients:

(Query 5) PATIENT.Temperature where PATIENT.Condition == "serious"

A condition on multiple-valued attributes is interpreted as at least one value satisfying the condition, that is, existentially. Universally quantified multiple

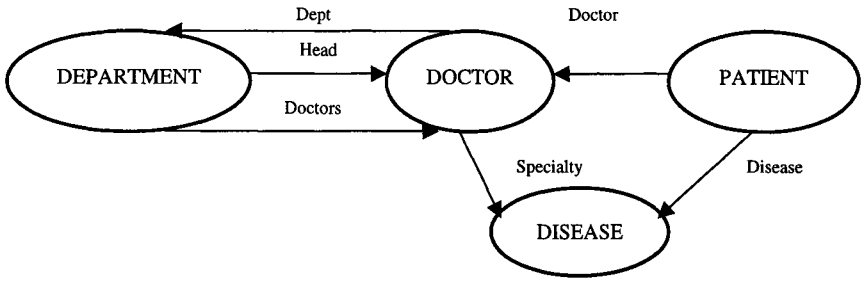


FIGURE 4 Part of a medical database structure.

attributes can also be specified as described later. The following retrieves the names of patients who ran a temperature of higher than 37.5°C at least once:

(Query 6) PATIENT.Name where PATIENT.Temperature >37.5.

Any class, leaf or nonleaf, in a generalization lattice can be specified in a set-oriented query. According to the interpretation of a class, the intrinsic instances of a nonleaf class and the instances of its subclasses can be retrieved at the same time. For example, to find persons who live in Kyoto:

(Query 7) PERSON where PERSON.Address == "Kyoto".

This causes a query be specified compactly because several queries against subclasses such as PATIENT and DOCTOR can be formulated in a single query.

Objects can be retrieved without precise specification since a general class can be specified in a query together with an attribute defined in its subclasses. In general, assuming that C and C' are classes and A is an attribute, C is systematically translated into C' in a query only if the following set is not empty: $\{C' | C'$ is a subclass of C and A is defined or inherited by $C'\}$. The original query usually generates multiple queries. Note that Query 7 is a special case where A (e.g., Address) is defined or inherited by all classes in a class hierarchy with C (e.g., PERSON) as its top. For example, to find the names of persons whose disease is a cancer:

(Query 8) PERSON.Name where PERSON.Disease.Name == "cancer"

The class PERSON is automatically specialized to the subclass PATIENT with the attribute Disease defined. In the extreme case, OBJECT can be used in a query. This mechanism fits with how we defined some concepts by differentiating a general concept by providing specializing attributes. The user can thus formulate a query without knowing specificity like a natural language query.

A condition can be imposed on the categorization attribute of a general class with a partition. If the specified condition matches some of the categorization conditions of the partition, the specified class can be specialized to some of the partition members. In general, assuming that C and C' are classes, C is systematically translated into C' only if the following set is not empty: $\{C' | C'$ is a subclass of C and the condition of a query and the categorization condition of C' are not exclusive $\}$. For example, to find infants (i.e., younger than seven):

(Query 9) PERSON where PERSON.Age <7

The class PERSON is automatically specialized to CHILD with its categorization condition Age <18.

The user can do operations other than retrieval set-theoretically by using procedural attributes, which can be specified in any part of an object expression of a query. The additional parameters of the procedural attribute are given in parentheses. In general, the object expression has the following form: receiver. method (parameter, parameter, ...). A receiver is an object expression and a parameter is an object expression, an object, or a value. The result is also an object, a value, or a set of objects or values. For example, to make and print a copy of serious patients' medical certificates dated February 11, 1999, the user formulates the query

(Query 10) PATIENT.make-medical-certificate("19990211").print()
where PATIENT.Condition == "serious".

If we operate on objects set-theoretically in a setting other than Jasmine, we then must retrieve a set of objects and scan and operate on each object in an iteration construct. In contrast, Jasmine makes this type of iteration implicit and iteration variable declaration unnecessary and allows the user to compactly specify a query without knowing the details.

We can also specify procedural attributes in incomplete knowledge access. If we specify a general class whose subclasses have procedural attributes of the same interface which have different implementations, the different attributes are invoked in a single query at the same time. In general, assuming that C and C' are classes and M is a method, C is systematically translated into C' only if the following set is not empty: {C'|C' is a subclass of C and M is defined or inherited by C'}. M has a different implementation, depending on C', so this realizes polymorphism in a set-oriented manner. For example, to display all heterogeneous media objects belonging to James Bond, such as X-ray and CT, the user specifies the query

(Query 11) MEDIA.display() where MEDIA.Patient.Name == "James Bond".

Both system-defined and user-defined procedural attributes can be specified in the same way unlike other systems such as [6]. The system-defined procedural attributes include print and object modification operations such as put, replace, delete, and destroy. Of course, they can be invoked in a set-oriented query. In other words, the user can extend the query language without changing the parser or the code generator. For example, the following query adds 38.0°C to James' temperature (multiple-valued attribute):

(Query 12) PATIENT.put("Temperature", 38.0)
where PATIENT.Name == "James Bond".

Attributes taking a set of objects and giving a singleton, called aggregate functions, can be specified in a set-oriented query. They include the system-defined attributes such as count, average, sum, max, and min. Since a set in our context allows duplication of objects, the user can use the aggregate functions naturally. For example, to find the number of outpatients who are under 7 years of age, the user forms a query as

(Query 13) PATIENT.count() where PATIENT.Age < 7
and PATIENT.Category =="outpatient".

The following query finds the average of the ages of inpatients:

(Query 14) PATIENT.Age.average() where PATIENT.Category == "inpatient".

In general, the aggregate functions take as input the whole flattened set retrieved just before the function evaluation. This can cause subtle problems when the user applies the aggregate functions to the multiple-valued attribute. Assuming that the attribute Temperature is multiple-valued, consider the following:

(Query 15) PATIENT.Temperature.average() where
PATIENT.Category == "inpatient".

This evaluates to the average of an automatically normalized set of objects as the values of Temperature of more than one inpatient. Therefore, if the user wants to apply the average to Temperature values of each PATIENT object, the user specifies a special operator "{}" to prohibit automatic unnesting of multiple-valued attributes as

(Query 16) PATIENT.{Temperature}.average() where
PATIENT.Category == "inpatient".

Multiple-valued attributes can be universally quantified by specifying "All" before comparison operators. The following query retrieves the names of patients whose temperatures are all over 37.5 in contrast to Query 6.

(Query 17) PATIENT.Name where PATIENT.{Temperature} All >37.5.

Grouping is allowed. The following calculates average temperatures for each group of inpatients of the same age:

(Query 18) PATIENT.Temperature.Average() where
PATIENT.Category =="inpatient" groupby PATIENT.Age.

Procedural attributes can also be specified in object expressions in the condition part of a query to filter objects procedurally. This is powerful in a variety of applications. A content-based search of multimedia data can be done by defining an attribute such as like. The following finds a patient whose X-ray looks like sample-1 containing some disease although content-based multimedia retrieval will be discussed in Section IV in detail:

(Query 19) X-ray.Patient where X-ray.like(sample-1) == true.

Usually the syntactically same object expressions in a query have the same semantics. However, making aliases of object expressions is possible anywhere in a query by qualifying them if necessary, for example, to do self-join. The following query retrieves pairs of patients who suffer from the same disease. Note that the second condition eliminates duplication:

(Query 20) [P1:PATIENT, P2: PATIENT] where P1: PATIENT.Disease == P2:
PATIENT.Disease and P1: PATIENT.Id < P2: PATIENT.Id.

Now we can define the semantics of a query by using the semantics of object expressions defined earlier. First, the condition part is evaluated as follows: If simple conditions comparing object expressions are evaluated to be true, based on the values of the object expressions, the Boolean combination of them is evaluated in the usual way. If the result is true, we then evaluate the target part and we get a tuple of objects or values as a query result.

It is necessary to individually access a set of retrieved objects in application programs. To this end, we introduce a variable that can be bound to a set of objects. The variable is called a set variable. The set variable is usually set to the result of a set-oriented query. The user can access objects one by one by scanning the set variable. The instance variable is also introduced to hold a singleton. The instance variable holds a single object like usual variables in a conventional programming language. The instance variable and set variable

constitute the object variable. The object variable integrates set-oriented access of a database system and singleton access of a programming language. The existence of a multiple option at declaration specifies that the object variable is a set variable. For example,

PATIENT *ps* multiple, *p*;

ps and *p* are declared as set variable and instance variable of PATIENT type. In general, the set variable is set to the result set of a set-oriented query at the right-hand side of a statement. The user can access objects individually by using the system-defined procedural attributes as

```
ps = PATIENT where PATIENT.Diseasee == "cancer";
ps.openscan();
while (p = ps.next())
    {...}
ps.closescan();
```

The procedural attribute next returns an object at each invocation, which is set to the instance variable *p* for further use.

Procedural attributes can include set-oriented queries. The following attribute of the class DEPARTMENT defines interns who work in a department:

```
Procedural DOCTOR intern() multiple
  { self.Doctor where self.Doctor.Status =="internship" }.
```

This can be specified in a query to retrieve interns in the pediatrics department as

```
DEPARTMENT.intern() where DEPARTMENT.Name == "pediatrics".
```

We do not provide a special syntax for nesting queries. Instead, nested queries can be simulated by procedural attributes defined by queries like the above example. Correlated queries can be formulated explicitly by passing object expressions as parameters to the procedural attributes or implicitly through the system-defined variable self.

III. IMPLEMENTATION

Relational databases have already accumulated large amounts of implementation technology. We don't think that it is clever to throw it away and to build object-oriented databases from scratch. Relational technology provides basically applicable techniques such as storage structures, access methods, query optimization, transaction and buffer management, and concurrency control. Therefore, we take a layered architecture consisting of object management and data management and use relational technology as data management (see Fig. 5). However, traditional relational technology has limitations in efficient support for object-oriented concepts including object identifiers, complex objects, class hierarchies, and methods. We extend relational technology to overcome such limitations. In addition to flat relations, we incorporate nested relations to efficiently store and access clustered complex objects. We support

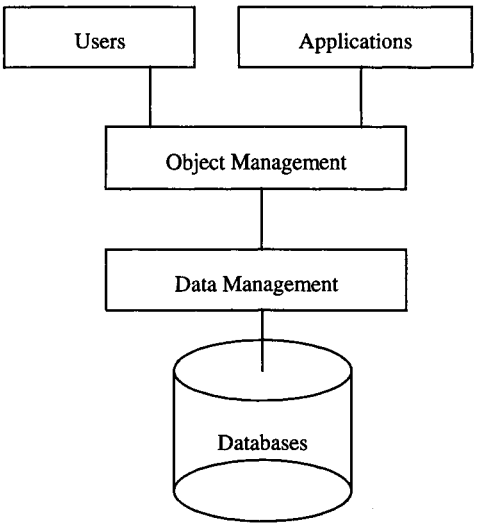


FIGURE 5 System architecture.

both hash and B-tree indexes to efficiently access objects through object identifiers. In addition to a nested-loop join and a sort-merge join, we provide a hash join to efficiently process nonclustered complex objects in queries. We extend query optimization to process object-oriented queries including class hierarchies and method invocation. Note that such optimization is done not by the data management subsystem but by the object management subsystem. We provide user-defined manipulation and predicate functions directly evaluated on page buffers. Methods are compiled into them and efficiently processed. We devise object buffering in addition to page buffering and integrate these two schemes to evaluate queries. In a word, our approach is to provide an object-oriented model and language interface to an extensible database kernel [37], such as GENESIS [1] and EXODUS [2].

Of course, there are alternatives to our extended relational approach to object-oriented database implementation. A pure relational approach such as Iris [31] has drawbacks as described above. Another approach uses WiSS (Wisconsin Storage System) such as O2 [5], which provides record-based, single-relation operators. This makes it difficult for us to focus on query optimizations based on set-oriented relational operators. In the extreme case, monolithic architectures could be considered in contrast to our layered approach. This would be less flexible to further tuning and extension. In this section, we will explain the function and implementation of the data management subsystem, and storage of objects and implementation of the object manipulation language.

A. Data Management Subsystem

1. Data Structures

Advanced applications of OODBs require a variety of indexes such as hash and B-tree indexes, clustered and nonclustered indexes, and extended

data dictionaries. Such indexes and data dictionaries are usually implemented as special data structures in relational database systems because of access efficiency. The conventional approach using special data structures makes the system less compact and less flexible to future extension. So the data management subsystem as a database kernel supports only relations (sequential, B-tree, hash, and inner relations) to allow the user of this subsystem to customize data dictionaries and indexes by using relations.

Only fixed-length and variable-length data are supported as field types of tuples by the data management subsystem. The data management subsystem makes no interpretation of field values except for TIDs and inner relations. Any type of data can be stored such as an array, a list, and a relation. Inner relations can be implemented as variable-length fields. Inner relations can have other inner relations as field values, so nested relations can be recursively defined. The length of a tuple must be less than the page size for efficient access and simple implementation. The length and number of fields in a tuple are subject to this limit.

The data management subsystem supports four types of relations as follows:

1. Sequential relations have pages that are sequentially linked. Tuples are stored in the order of insertion. The location of inserted tuples is fixed, so an index can be created on sequential relations.
2. B-tree relations have B-tree structures. Tuples are stored in the leaf pages in the order specified by user-defined order functions. This allows new access methods to be assimilated by supplying dedicated comparison and range functions. B-tree relations consist of key fields and nonkey fields. B-tree relations used as an index on sequential relations consist of several key fields and one TID field. This corresponds to a nonclustered index. B-tree relations that contain the whole data can be viewed as relations with a clustered index.
3. Hash relations use a dynamic hashing scheme called linear hashing with partial expansion [30], an extension to linear hashing. We choose this scheme because the space required to store data is proportional to the amount of data and the space utilization ratio is adjustable and high. Hash relations also consist of key fields and nonkey fields. The hash function is supplied by the user.
4. Inner relations for realizing nested relations are stored in variable-length fields of tuples. Tuples of inner relations are sequentially inserted. Nested relations can be recursively implemented by storing inner relations into fields of another inner relation. We provide nest and unnest operations for nested relations in addition to retrieval, insertion, deletion, and update. Retrieved inner relations can be operated as sequential relations. Update of inner relations can be done by retrieving inner relations, updating them as sequential relations, and replacing old ones by new ones. We provide the functions interpreting the variable-length fields according to the nested relation schemes to operate on inner relations. Note that a theoretical basis for the nested relational model was provided by Kitagawa and Kunii [27].

Tuple structures are uniform independently of relation types (see Fig. 6). The first two bytes of a tuple contains the tuple length. The tuple consists of fixed and variable parts. Fixed-length fields are stored in the fixed part.

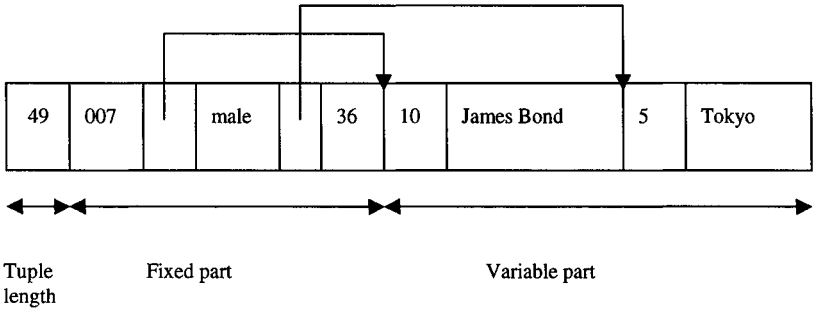


FIGURE 6 Tuple structure.

Variable-length fields are stored in the variable part. The offsets of the variable-length fields from the top of the tuple are stored in the fixed part. Any data can be accessed in a constant time, although this tuple structure does not allow null-value compression. Modification of the variable-length data can be done without affecting the fixed-length data.

TIDs, which can be stored in fixed-length fields, act as pointers to tuples. A variety of data structures can be implemented by using TIDs. For example, a nonclustered index can be implemented by defining an index key field and a TID field in B-tree or hash relations (see Fig. 7).

Access to fields must be efficiently processed since it is a frequent operation. We provide pointer arrays for field access (see Fig. 8). Each pointer points to the corresponding field in a tuple on page buffers. Simple tuple structures allow efficient construction of pointer arrays. One alternative is to copy field values to different areas. The alternative is good for data protection, but is rather time-consuming. Field pointer arrays are passed to user-defined functions such as manipulation and predicate functions for field access.

To efficiently access data, we move as few data as possible and fix tuples in buffers if possible. Internal sorting uses pointer arrays for tuples to be sorted (see Fig. 9). Such pointers are moved instead of tuples. Similarly, when a hash table is created for internal hashing, pointers to tuples are linked instead of tuples.

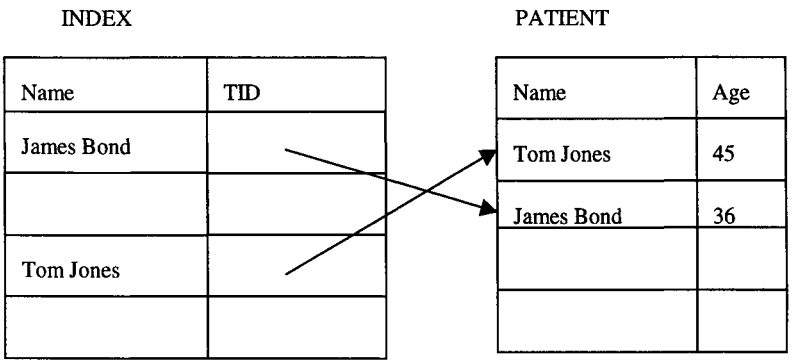


FIGURE 7 Nonclustered index using a B-tree relation.

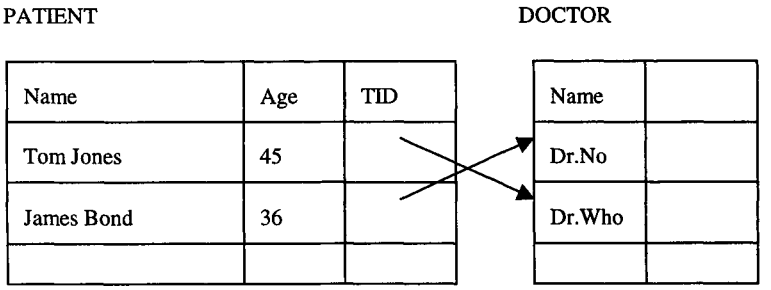


FIGURE 8 Precomputed join.

2. Hash-Based Processing

Set operations such as set-difference and duplicate elimination require OID-based access. Object-oriented queries usually equi-joins based on OIDs. If either of two joined relations can be loaded into main memory, we can use the hash join method [36]. Even if neither of them can be loaded into main memory, the hash join method generally requires less CPU time and I/O times than the sort-based method. We adopted the hash-based method for equi-joins and set operations. Unlike Jasmine, other object-oriented systems such as ORION use nested-loop and sort-merge joins.

The internal hash join is used when either of two input relations for joins can be loaded into main memory. Recursion is not used in the internal hash join. Only one relation is partitioned into subrelations. The other relation is only scanned tuple by tuple. It is not necessary to load both of the relations entirely. We describe the outline of the algorithm. Only main memory is used during processing.

(1) Determine which input relation is to be partitioned. Let the partitioned input relation be A.
(2) Determine a partition number p and a hash function h.

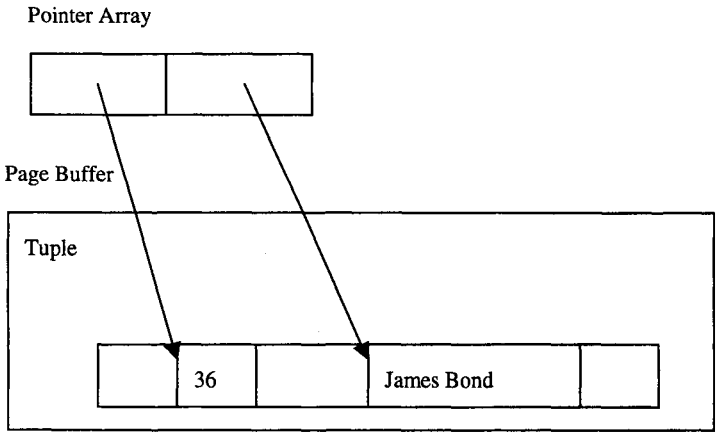


FIGURE 9 Pointer array for field access.

(3) Partition the relation A into p subrelations $Ai = \{a$ belongs to $A|h$ (key of $a) = i\}$ $(i = 0, \ldots, p\text{-}1)$.
(4) For each tuple b of the other relation B, compute $k = h$ (key of b) and compare each tuple of Ak with b on the join key. When they match, make a new tuple from them and output it to the output relation C.

The external hash join is used when neither of two input relations can be loaded into main memory. The essential difference between the external hash join and the internal hash join is the use of recursion and the partitioning of both of the input relations. The outline of the algorithm is as follows:

(1) Determine a partition number p and a hash function h.
(2) Partition the relation A into p subrelations $Ai = \{a$ belongs to $A|h$ (key of $a) = i\}$ $(i = 0, \ldots, p\text{-}1)$, and partition B into p subrelations $Bi = \{b$ belongs to $B|h$ (key of $b) = i\}$ $(i = 0, \ldots, p\text{-}1)$. Each subrelation is stored in secondary memory.
(3) For each $i = 0, \ldots, p\text{-}1$, if either Ai or Bi can be entirely loaded into main memory, Ai and Bi are joined using the internal hash join. Otherwise, steps (1) through (3) are executed recursively.

3. User-Defined Functions

Methods are often specified in target and condition parts of object-oriented queries. Conventionally, applications retrieve all data and filter and manipulate them. This approach is rather inefficient because it requires extra data transfer and conversion between buffers and applications. System-defined comparators are also inefficient because they interpret any data according to data types. So we implement application-specific parts such as methods as user-defined functions and embed them into the the data management subsystem. The user can specify application-specific parts as follows.

(1) A predicate function specifies a retrieval condition of selection or join operators.
(2) A manipulation function specifies operators on each tuple satisfying the predicate in selection or join.
(3) An order function specifies the order used by sorting or B-tree relations.
(4) A range function specifies a search condition of B-tree relations such as $a <= x < b$ or $c < x$.
(5) A static hash function is used by hash-based relational operators such as join, union, and difference.
(6) A dynamic hash function is used by hash relations.

To separate application-specific parts by providing user-defined functions allows both flexible customization by the user and efficient execution by compiling. For example, the following predicate function filters tuples by using a simple condition with the system-defined comparison operator <:

```
predicate 1 (flag, OID, condition, age)
{if (flag == MAIN)
  {if (age < 13) return true
    else return false}}
```

Whether this predicate is invoked for PREprocessing, MAINprocessing, or POSTprocessing depends on the variable flag. Preprocessing and postprocessing are done only once while the main processing is done tuple-wise. Control is transferred to manipulation functions which manipulate filtered data only if the predicate function returns true. The predicate functions are compiled into operations on tuples in the buffer and are efficiently processed because no type is dynamically checked and no data are interpreted. The user-defined complex condition is also defined as a predicate function. If no predicate is explicitly supplied, control is always passed to manipulation functions.

Manipulation functions are compiled to operate directly on tuples in the buffer and are invoked only if predicate functions return true. For example, the following function operates on tuples of a relation by using a make-medical-certificate program and inserts the result into another relation tmp3:

```
manipulate3 (flag, OID, doctor, name, disease)
{if (flag == PRE) openinsert(tmp3);
  else if (flag == MAIN)
    {result = make-medical-certificate("19990211");
      insert(tmp3, result);}
  else if (flag == POST) closeinsert(tmp3);}
```

In general, selection and join operators described below require predicate and manipulation functions. Data are thus efficiently filtered by predicate functions and operated on by manipulation functions.

4. Architecture

The data management subsystem has a layered architecture consisting of relational, tuple, and storage layers (see Fig. 10). All of these are open to the user. The data management provides neither a query parser nor an optimizer because they are rather high-level and application-dependent. It is just an executer of operators provided by the three layers.

The relational layer provides functions that execute set operations as an extended relational algebra. For example, (1) select (rb, pb, mb) extends

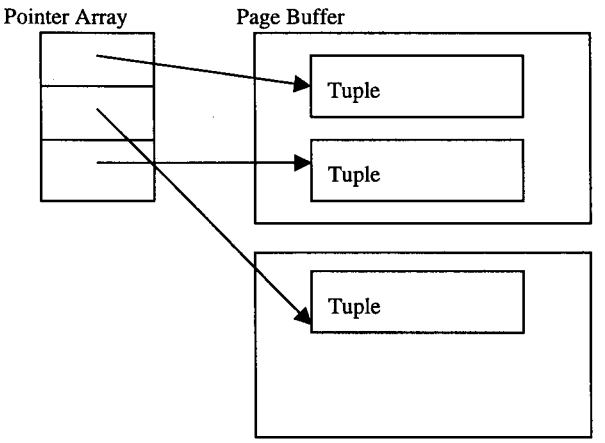


FIGURE 10 Pointer array for internal sorting.

selection of relational algebra. It has three parameters rb, pb, and mb. rb is the data block that specifies the source relation. pb and mb specify user-defined predicate and manipulation functions, respectively. (2) hjoin(rb1, rb2, mb, hb) performs an equi-join of relations specifying rb1 and rb2. mb is performed on each pair of tuples which match on join fields. This operation is based on a hash function specified by hb. (3) join(rb1, rb2, pb, mb) performs a general join of relations rb1 and rb2. (4) tjoin(rb1, rb2, TID, mb) joins each tuple of rb1 with a tuple of rb2 pointed by its TID field and performs mb on such a pair of tuples. (5) sort(rb1, rb2, ob) sorts rb1 and stores the results into rb2. The order function is specified by ob. (6) unique(rb1, rb2, hb) eliminates duplicates of rb1 and stores the result into rb2. This operation is hash-based. (7) nest(rb1, rb2, fid, hb) generates a nested relation rb2 from a flat relation rb1 with fields specified by fid. This operation is also hash-based. (8) unnest(rb1, rb2, fid) generates a flat relation rb2 from a nested relation rb1.

Functions of the tuple layer operate on four types of relations. The operators are as follows: (1) Scan scans a relation sequentially and finds a tuple satisfying the specified predicate. (2) Raster scans a relation sequentially fixing scanned pages on buffers. It is used in internal sorting or making internal hash tables. (3) Access directly accesses a tuple satisfying the specified predicate. (4) Fetch, (5) delete, and (6) update directly accesses, deletes, and updates a tuple specified by a given TID, respectively. (7) Insert inserts a tuple or a group of fields. (8) Clear deletes all tuples. (9) Flac constructs a field pointer array for the specified fields.

The functions of the storage layer include disk I/O, page buffers, transactions, concurrency control, and recovery. Disk I/O management includes allocation and deallocation of subdatabases (segments) and pages. A database consists of two types of subdatabases. One is a subdatabase that is permanent and recoverable. The other is a subdatabase which is used as a workspace for keeping temporary relations, and is only effective in a transaction. This is not recoverable. Subdatabases are composed of a number of pages.

The storage layer supports variable-length pages sized $2i$ KB ($i = 2, \ldots, 8$), consisting of several 4 KB physical pages, which form a virtually continuous page on buffers. We use the buddy system for buffer space allocation. The page length can be specified for each relation because multimedia data and inner relations may exceed 4 KB.

We use concurrency control based on granularity and two-phase locking. Deadlock detection is done by examining a cycle in the Wait-For-Graph. One of the deadlocked transactions in the cycle in the graph is chosen as the victim for rollback. ORION uses deadlock detection based on timeouts. Our transaction recovery is based on shadow-paging for simplicity.

B. Object Management Subsystem

1. Object Storage

We efficiently store nested structures of objects by use of nested relations supported by the data management subsystem unlike other systems. Storage structures differ from instance to class. Translation of objects to relations is automatically done by the system. Information about the translation is held by classes.

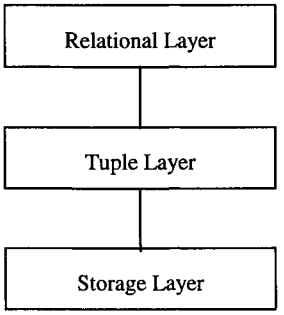


FIGURE 11 Architecture.

All intrinsic instances of a class are stored in a relation, corresponding an instance to a tuple and an attribute to a field (see Fig. 11). Instances of different classes are stored in separate relations. Multiple-valued attributes such as Temperature are stored as a multiple-valued field, the simplest form of nested relations. We resolve attribute inheritance before storing instances. We treat inherited attributes such as Age and newly defined attributes such as Weight uniformly. We store intrinsic instances of a superclass and those of a subclass in separate relations. If we instantiate or destroy intrinsic instances of a class, we do not have to propagate any modification to its superclass or subclass.

We store fixed-length strings and numbers in fixed-length fields and variable-length strings in variable-length fields. As for reference attributes such as Doctor we store only OIDs in fixed-length fields. This implements nonclustered complex objects. Nonclustered complex objects are needed by CAD applications where complex objects are created bottom-up; that is, component objects are reused. Of course, the user can enforce complex object integrity on nonclustered complex objects by attaching demons to methods of whole objects such as instantiate [14]. We correspond relation objects in attributes to inner relations of nested relations and their attributes to fields of the inner relations. Inherently, component objects of relation objects cannot exist without their whole objects. Clustered complex objects, implemented by relation objects, can be managed as a unit both logically and physically. Nested relations naturally realize clustering of complex objects, although component objects must be hierarchically accessed from root objects.

The OID attribute also corresponds to a field. An OID consists of a database id-number, a class id-number, and an instance id-number, so an OID alone can inform us of its database and class information directly (see Fig. 12). Unlike the OID of O2 [5], the OID of Jasmine is logical in that it contains no physical addresses in secondary memory. No OIDs need to be changed even at data reorganization, although OIDs need to be changed if objects are migrated to another class. The user can choose among sequential, B-tree, and hash relations as instance storage. The user can attach indexes to attributes.

As classes are instances of the system-defined CLASS in Jasmine, we store all classes in one nested relation and facilitate efficient associative access of class objects. Basically we correspond one class object to one tuple. Since attribute categorization, such as enumerated and procedural, is common to all

PATIENT (Instance Table)

Oid	Sex	Age	Name	Doctor
MedicalPatient007	male	36	James Bond	MedicalDoctor000

	Category	Temperature	Weight	Height
	inpatient	36.5 37.3 38.1	76.0	181.0

CLASS (Class Table)

Name	Db	Super	Property		
			name	Class	if-needed
PATIENT	MEDICAL	PERSON	Doctor	DOCTOR	
			Weight	FLOAT	
			Height	FLOAT	DEMON030

	Method			
	name	Class	Main	
	Make-medical-certificate	Medical-certificate	METHOD001	

FIGURE 12 Examples of an instance table and a class table.

classes and attributes have a fixed set of facets, we store enumerated and procedural attributes in different inner relations and facets in the fields of the inner relations. Procedural attribute (method) definitions are also stored in relations and are retrieved and utilized during query optimization. The system-defined attributes such as Super are stored in separate fields (see also Fig. 12). To store heterogeneous classes in one relation makes set-oriented access to them efficient.

2. Set-Oriented Access Support

We compile both set-oriented access and singleton access to do early binding and reduce run-time overhead. The Jasmine compiler is implemented using

a C compiler. The application programs written in Jasmine/C are precompiled to C programs, which are compiled and linked with the run-time support library. Preprocessing is used to take maximum advantage of portability and code optimization of the C compiler. An interactive query needed by application development is processed by the interpreter, not by the compiler.

Application programs are translated into C in three phases: query graph generation, access plan generation, and code generation. The query graph generation phase makes a query graph corresponding to the user query by referencing the object model. The query graph is a subgraph of the object model annotated with the target and condition information. For example, consider the query

PATIENT.make-medical-certificate("19990211"),

where PATIENT.Age < 13 and PATIENT.Doctor.Name == "Dr. No".

This makes the following query graph:

CHILD-PATIENT (make-medical-certificate("19990211"),
Age < 13, DOCTOR (Name == "Dr. No")).

Note that the user query containing incomplete knowledge access is transformed into a more efficient one during this phase. In Jasmine, the user can form a query by specifying a general class such as PATIENT instead of the specific class CHILD-PATIENT. Then it is necessary to restrict the general class to its appropriate subclass.

Then the access plan generation phase translates the query graph into an optimal plan of a series of extended relational operators using the object model information such as statistics, access methods, and mapping from class to relation. This phase uses rule-based optimization. Rules are grouped into sets of rules called rulesets [12] according to patterns of queries. This increases the modularity of rules. Rulesets can be more efficiently processed than flat rules because of this modularity. They are also easier to maintain and extend. Object-oriented query optimization is fully described later. Here we just illustrate the overall flow of query processing. Object-oriented features such as complex objects, class hierarchies, and methods (procedural attributes) constitute patterns of rulesets. Their occurrences in queries invoke associated rulesets. For example, the object expression is processed by different rulesets depending on whether it contains a procedural attribute.

First, the case where object expressions contain no procedural attributes is considered. In general, the object expression generates equi-joins between instance relations corresponding to a functional join. In addition, the conditions generate selections and explicit joins. For equi-join, a predicate function which joins two instance relations by an attribute field of one relation and an OID field of the other is generated. For selection, a condition concerning one instance relation is generated as a predicate function of a select operator. For an explicit join, a join predicate function which may contain a non-equi-join is generated. Manipulation functions are generated to project fields for later operation.

The query results in a set of OIDs, values, or tuples. A query against a nonleaf class evaluates to a relation containing OIDs of instances of several classes. As OIDs have the same structure for all objects, they can be stored in

one relation. Each scan returns an object, by scanning the result relation and then selecting the base instance relation by the OID. At that time, if an object is already on core, it is used.

Usually, if a selection predicate of the sequential relation can use an index, selection by index, which selects a B-tree relation for the index by the key condition, sorts the result relation containing TIDs, and then joins the result relation and the original sequential relation by using TIDs, is chosen. The rest of the selection condition is evaluated at the same time. For B-tree and hash relations, if a predicate concerns the key fields, key-based searching is done. Note that if a whole relation of any type is small enough to be contained within one page, sequential access is chosen.

If one of two relations being joined is small enough to be contained within a page and the join key is indexed by the other relation, tuple substitution is chosen. If one of two relations is contained within a page and no index is provided, nested loop is chosen. Otherwise, hash join is chosen. For B-tree and hash relations, the join is similarly processed. In case of a join of several relations, the order of join is dynamically determined by the size of the intermediate result relations. First, we choose the smallest relation and the second smallest one among relations to be joined. Then we join them to obtain an expectedly small relation as a result. We add the result to relations to be joined and repeat this process. This dynamic scheme based on exact relation sizes is expected to be more efficient than static schemes based on database statistics.

Next, consider the case where object expressions contain procedural attributes. User procedural attributes appearing in the target part are translated into manipulation functions. Procedural attributes in conditions are translated into predicate functions. For example, the above query graph generates the relational operator sequence and the predicate and manipulation functions as follows:

```
select(DOCTOR, predicate1, manipulate 1);
select(CHILD-PATIENT, predicate 2, manipulate 2);
if (within-page(tmp1)||within-page(tmp2)) join(tmp1, tmp2, predicate3,
  manipulate3);
else hjoin(tmp1, tmp2, predicate3, manipulate3, hashfunc);

predicate 1 (flag, OID, name)
  {if (flag == MAIN)
    {if (name == "Dr. No") return true
      else return false }}

manipulate1 (flag, OID)
  {if (flag == PRE) openinsert (tmp1);
    else if (flag == MAIN) insert(tmp1, OID);
    else if (flag == POST) closeinsert(tmp1);}
predicate2 (flag, OID, condition, age)
  { if (flag == MAIN)
    {if (age < 13) return true
    else return false}}
```

```
manipulate2 (flag, OID, doctor, name, disease)
  {if (flag == PRE) openinsert(tmp2);
    else if (flag == MAIN)
      {Acertificate = make-medical-certificate ("19990211");
      insert(tmp2, doctor, Acertificate);}
    else if (flag == POST) closeinsert(tmp2);}

predicate3 (flag, OID, doctor)
  {if (flag == MAIN)
    {if (OID == doctor) return true
      else return false }}

manipulate3 (flag, Acertificate)
  {if (flag == PRE) openinsert(tmp3);
    else if (flag == MAIN) insert(tmp3, Acertificate);
    else if (flag == POST) closeinsert(tmp3);}
```

If an index on Name of DOCTOR is available, the first select will be replaced by the sequence select–sort–tjoin (selection by index). The system-defined procedural attributes such as aggregate functions and update functions are also translated into manipulation functions. In particular, the update functions are translated into relational update operators. If demons are defined and the option is specified, they are integrated into manipulation and predicate functions. Lastly, the C codes for the given query are generated to feed the C compiler.

3. Object-Oriented Query Optimization

The features of query optimization in an object-oriented database are different from those of query optimization in a relational database because multiple-valued attributes, implicit joins, procedural attributes (methods), and nonleaf classes in a class hierarchy are specified in a query. First, we describe processing of multiple-valued attributes. As for non-clustered complex objects, reference attributes contain only OIDs and multiple-valued attributes contain only elements of a set. Then multiple-valued attributes contain a set of OIDs or values. Since only sequential access is supported for inner relations of nested relations, multiple-valued attributes are unnested into flat relations and are optimized conventionally except for application of aggregate functions. As for clustered complex objects implemented by nested relations, predicate and manipulation functions of inner relations of nested relations are nested into those of outer relations of the nested relations. They are recursively evaluated from outer relations to inner relations.

Next, we describe implicit joins of relations generated by object expressions such as DOCTOR.Patient.Age. If there is no available index on the OID field of the relation for the class of the attribute (e.g., PATIENT), the join is processed by hash joins. The order of more than one join is dynamically determined by the size of the intermediate result relations. If there is an index available on the OID, the join is processed by TID joins. In case of several joins, they

are processed from left to right in the object expression. Section predicates, if any, are evaluated during join processing. Note that there are methods for precomputing joins. For example, to process the query (DOCTOR.Patient.Age where DOCTOR.Patient.Age > 30), an index with Age as a key value and the OID of DOCTOR as a pointer value is created. Other systems such as ORION use this approach. However, it is rather difficult to maintain such an index properly.

We describe how to process queries containing nonleaf classes in a class hierarchy. We assume that PATIENT has ADULT-PATIENT and CHILD-PATIENT as subclasses. Consider the following examples,

(Query 21) PATIENT.Name where PATIENT Age > 12
and PATIENT.Age < 20
(Query 22) DEPARTMENT.Doctor.Patient.Name where
DEPARTMENT.Name == "pediatrics".

For Query 21, the system generates two subqueries:

result = ADULT-PATIENT.Name where ADULT-PATIENT.Age < 20
result = result + CHILD-PATIENT.Name where CHILD-PATIENT.Age > 12.

The two query results are inserted into the same output relation.

For Query 22, the join of DEPARTMENT and DOCTOR is processed first. During the join processing, the intermediate output relations are switched according to the class of the OID for DEPARTMENT.Doctor.Patient. The class can be determined just by looking at the OID. The pseudo queries are

adult-intermediate = DEPARTMENT.Doctor.Patient
where DEPARTMENT.Name == "pediatrics" and
DEPARTMENT.Doctor.Patient.Class == <ADULT-PATIENT>
child-intermediate = DEPARTMENT.Doctor.Patient where
DEPARTMENT.Name == "pediatrics" and
DEPARTMENT.Doctor.Patient.Class == <CHILD-PATIENT> .

The switching is done during a single-join operation. The code for the switching is translated into the manipulation function of the join operator. Then a pair of adult-intermediate and ADULT-PATIENT and a pair of child-intermediate and CHILD-PATIENT are joined, and the results are merged. As described above, the intermediate result of selection or join operations is switched to separate relations containing only OIDs relevant to successive joins. This can establish optimal preconditions for the joins by avoiding unnecessary search.

Classes (e.g., PERSON, DOCTOR, and PATIENT) in a class hierarchy share inherited attributes such as Age. Basically there are two methods for creating indexes on classes in a class hierarchy. One method is to create only one index on a whole class hierarchy, called a class-hierarchy index. The other is to create a separate index, called a single-class index, on each class. Jasmine uses single-class indexes. Other systems such as ORION and O2 use class-hierarchy indexes. The class-hierarchy index has an advantage that the total size of index pages and the total number of accessed index pages are smaller than those of the single-class index. However, it is not always optimal when a class hierarchy

is partially specified in a query. Moreover, it is rather difficult to maintain such class-hierarchy indexes.

In some cases, semantic information such as categorization can be used to specialize nonleaf classes to specific ones. When a condition on the categorization attribute such as (Age < 7) is specified in a query containing a nonleaf class PATIENT, if the condition matches one of the categorization conditions of partition classes (Age < 18 for CHILD-PATIENT), the nonleaf class (PATIENT) is specialized into the subclass (CHILD-PATIENT) with the matched categorization condition. This can reduce the size of the search space for the query.

The system translates methods such as make-medical-certificate of PATIENT specified in a query into the manipulation and predicate functions of selection or join operators, and processes them on page buffers, which avoids unnecessary data transfer between page buffers and application programs. Methods defined by a query such as intern of DEPARTMENT is expanded into the outer query. To this end, the source codes and compiled codes for methods and demons are stored as program objects in databases. They are retrieved and compiled during query optimization. To store programs in databases makes the integration of query and programming facilities more elegant than to store them in ordinary program files. Polymorphic methods are translated into their corresponding implementation functions.

4. Object Buffering

Singleton access is also compiled into C programs, which are compiled and linked with the run-time support library. First, run-time support will be described. The first access of an object fetches the object from secondary memory to the page buffer. Then the object is cached in the active object table (AOT), a dedicated internal hash table for object buffering (see Fig. 13).

The primary role of AOT is to efficiently look up objects. When an instance is referenced through its OID for the first time, the instance is hashed by its OID as a hash key. A hash entry (an object descriptor) and an in-memory instance data structure is created. The hash entry points to the instance data structure. If the instance is referenced through its OID by other instances resident in AOT, the OID is mapped to the pointer to the hash entry through the AOT. The pointer can be cashed into the attribute of the referencing instance since

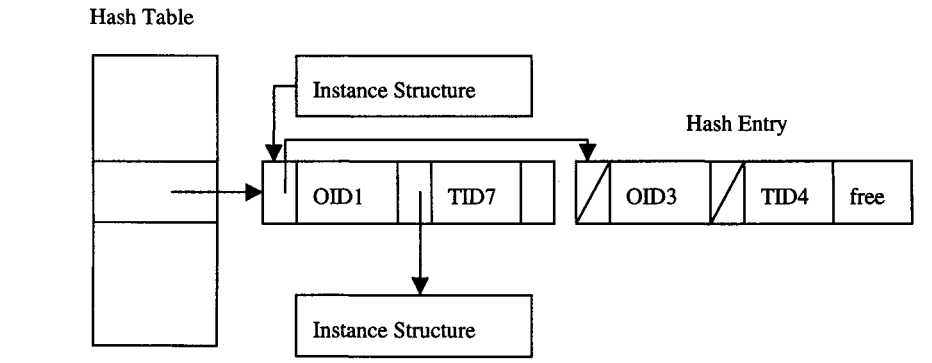


FIGURE 13 AOT structure.

an OID is longer than a physical pointer. Later, the instance can be directly accessed by the pointer without hashing.

Another important role is to maintain the status flags for update of objects. When an object is newly created or updated, the status flag in the hash entry for the object is set to create or update. When a transaction is committed, the object with the status create or update is modified or added into the page buffers. When an object is destroyed, the corresponding in-memory data structure is deallocated and the status flag is changed to destroy. Later, if the destroyed object is referenced, the validity of reference is checked and an exception handler is invoked. This can support referential integrity. When a transaction is committed, the object is destroyed in databases.

When objects fill up AOT, extraneous objects are swapped out. Such an object is flushed to the page buffers and the in-memory instance data structure is deallocated and the status flag in the hash entry is set to free. When the object is referenced again, the object is directly fetched from databases to AOT through its TID cashed in the hash entry.

The object management subsystem requires AOT, that is, object buffers in addition to page buffers of the data management subsystem for the following reason. In general, buffers are directly associated with patterns of access of objects. Page buffers have structures suitable for access of different instances of the same class. AOT have structures suitable for access of correlated instances of different classes. Advanced applications such as CAD have combinations of two such patterns. This necessitates a dual buffer scheme consisting of page buffers and object buffers, not a single-buffer scheme, which would contain unnecessary objects and decrease memory utilization.

The dual buffer approach, however, makes the same object appear in different formats in different buffers at the same time, so we must maintain internal consistency between two objects denoting the same entity. Currently, we first write back updated or newly created instances from AOT to page buffers in query evaluation. Then we evaluate a query against page buffers. An alternative is to devise different search mechanisms for different buffers and evaluate the same query on different buffers and integrate the results, which would make the system less compact.

Basically there are two methods for query evaluation using object buffers and page buffers as follows.

Single-buffer evaluation method: (1) The instances newly created or updated associated with the classes specified by the query are searched in the object buffers. (2) They are flushed from the object buffers to the page buffers. (3) The query is evaluated against the page buffers.

Dual-buffer evaluation method: (1) The query is evaluated against the object buffers. (2) The same query is evaluated against the page buffers. (3) The two results are merged into one.

Jasmine adopts the single-buffer evaluation method while ORION adopts a more sophisticated version of the dual-buffer evaluation method. The single-buffer evaluation method needs to transfer objects from the object buffers to the page buffers. However, the single-buffer evaluation method eliminates the

need for dual evaluation programs and makes the system small and processing simple in contrast to the dual-buffer evaluation method. Anyway, the combinational use of object buffers and page buffers can support the integration of programming and query facilities at an architecture level.

IV. APPLICATIONS

A. Introduction

New multimedia applications emerging on top of information infrastructures include digital libraries [34] and document warehousing for document management and analysis, which are supposed to be most promising as such networked multimedia applications. We need next-generation database systems which enable users to efficiently and flexibly develop and execute networked multimedia applications.

First, we analyze the characteristics of these networked multimedia applications and discuss the requirements for a multimedia information system consisting of database, operating system (OS), and network layers and the issues for a multimedia database system. Then we describe our approach based on OODB and extended with agents, focusing on a general architecture of a multimedia database system and its implementation.

1. Requirements for a Multimedia Information System

First, we discuss the requirements for a multimedia information system in general.

1. Multimedia applications must be interactive. So each layer of the system must allow control of *quality of service* (QOS) parameters, such as latency, bit and frame rates, to interactively process multimedia data in real-time.

2. A huge amount of data in forms such as text, video, and images are required for multimedia services which every layer of the system must efficiently process. In the database layer, database techniques for efficiently storing and accessing a large volume of data, which include access methods and clustering, are required. In the OS layer, techniques such as hierarchical storage systems, and thread mechanisms are required. In the network layer, network protocols suitable for multimedia along with the ability to efficiently process such protocols are required.

3. There is heterogeneity in media data such as text and video, and temporal and spatial dependency between them. Users must be able to uniformly manipulate heterogeneous media data. Users must also be able to structure heterogeneous media data explicitly by defining links among them, that is, *hypermedia links*. Users must also be able to define temporal and spatial relationships among various media data. Such functionality must be provided by the database layer as *multimedia data models*. Heterogeneous physical media, such as magnetic optical disks, and CD-ROMs, must also be uniformly accessed. Stream media data, such as audio and video, must be temporally synchronized and processed in real-time essentially by the OS layer. The network

layer must efficiently process multimedia network protocols based on ATM, to accommodate various types of media traffic.

4. Advanced multimedia applications must allow for distributed processing. In particular, site autonomy must be guaranteed for developing distributed multimedia applications. Users should be able to use distributed, object-oriented computing techniques, known as *active objects* or *agents*. Heterogeneity in data models, database systems, and database schemas must also be resolved, so the database layer must provide multidatabase management [21, 29]. The OS layer must perform distributed resource management.

2. Issues for a Multimedia Database System

Now we address the issues for a multimedia database system for networked multimedia applications, which are not comprehensive but mandatory.

1. We must distinguish between logical media data and physical media data. For example, in networked multimedia applications, multimedia contents are updated or deleted, or even moved from one server to another. We must allow users to access such contents independently of physical details such as locations. We must also allow users to access contents in a uniform fashion independent of data formats such as MPEG and Motion JPEG. Thus, we must allow users to flexibly define multimedia views by specifying mappings between logical and physical data.

2. We must provide a query facility based on keywords, which is a prerequisite for database systems, needless to say. Browsing alone is insufficient because a large amount of media data take a long time to play.

3. We must also provide a content-based retrieval facility [7]. In networked multimedia applications, keywords are not always attached to a large amount of data in advance. Moreover, users should sometimes express a query over multimedia data by using features, such as colors and motion directions, which are different from conceptual keywords. So we need content-based retrieval, which allows an inexact match in contrast to an exact match facilitated by the keyword-based retrieval facility.

4. We must provide a navigational search facility in addition to the query facility. Of course, the navigational search can be done through user-specified hyperlinks among data [14]. However, in large-scale networked applications, explicit specification of links requires a considerable amount of work. So we must logically cluster multimedia data based on similarity of keywords and characteristic data, such as colors, for navigation.

5. We must allow users to select physical storage appropriate for applications and physical clustering if necessary.

6. We must provide parallel, distributed processing in networked multimedia applications. For example, several streams are sometimes required to be played in parallel. Distributing process burdens, such as special effects of video streams, among servers is required. Federating a query to several distributed servers is also required.

7. We must handle program components as first-class objects. Program components are used to control access to databases and to make database application development more efficient.

8. We must control QOS in networked multimedia applications. For example, when a single or multiple users require multiple streams to play in parallel, we cannot guarantee QOS required by users unless we manage resources such as CPU and network.

B. System Architecture

1. Our Data Model

a. Multimedia

We think that multimedia data are not just static data, but rather compositions of several media data and operations on them. So we provide structural, temporal, spatial, and control operations as media composition operators, as described later. In other words, our model has multiple facets and subsumes existing models, such as object, temporal, spatial, and agent models. Individual operations are orthogonal to one another. Our model is integrated seamlessly with existing technologies.

Multimedia systems consist of multimedia databases and applications. Multimedia databases consist of a set of media data. Media types include texts, graphics, images, and streams. Stream types include audio, video, and streamed texts, graphics, and images as well. Multimedia applications consist of a set of scripts. Basically, a script has an identifier (ID) and temporal and spatial operations on a set of streams with QOS options. A stream has an ID and temporal and spatial operations on a set of frames. A frame has an ID and temporal and spatial operations on frame data.

QOS options are parameters given to a QOS controller. QOS types include latency, jitter, various bit rates and frame rates, resolution, colors, and fonts. QOS is controlled either by executing specified QOS functions or by retrieving stored QOS data. The QOS function takes a stream and a time and gives frame IDs. The QOS data consisting of time and a frame ID are stored in advance by obtaining them from rehearsal.

In order to concretely explain the features of our data model, we consider the following multimedia application or script, called Script1, assuming that there exist multimedia databases containing multiple video streams that have filmed the same object.

Script1:

(a) Retrieves all video streams which filmed the Prime Minister on January 17th, 1995.
(b) selects only parts of the retrieved video streams temporally overlapping each other.
(c) Arranges the selected parts on the same presentation space (i.e., window).
(d) Plays the parts in temporal synchronization.

b. Structural Operations

Jasmine models structures of multimedia objects and provides structural operations on them. For example, media objects such as streams and frames are defined (see Fig. 14).

```
STREAM
    Super      MEDIA
Property
    TIME       Internal Time
    TIME       Real Time
    SPACE      Internal Space
    SPACE      Real Space
    Set FRAME  Frame
    STRING     Topic

MPEG
    Super      STREAM
Property
    Set MPEG-Frame Frame
Method
    MPEG   AddFrame ()
    MPEG   RemoveFrame()
    VOID   Forward()
    VOID   Backward()

FRAME
    Super MEDIA-Data
Property
    TIME         Time
    FRAME-Data   Data
```

FIGURE 14 Definition of media objects.

For example, the following query retrieves streams that filmed the Prime Minister on January 17th, 1995, which realizes Script1 (a):

STREAM.Frame from STREAM where STREAM.RealTime = "01171995" and STREAM.Topic = "the Prime Minister".

c. *Temporal and Spatial Operations*

Temporal and spatial data are viewed as universal keys common to any stream media data. Such temporal and spatial relationships structure multimedia data implicitly in contrast to explicit links. We define set-oriented temporal and spatial operators specifying such relationships, which are analogous to relational algebra [3].

Although time is one-dimensional and space is three-dimensional, they have similar characteristics. Real-time is elapsed time taken for recording streams in the real world. Internal time is time required for a usual play of streams. External time is time taken for real play of streams by scripts. Usually, real time is equal to internal time. In the case of high-speed video, real time is shorter than internal time. External time is specified by providing a magnification level of internal time. Default magnification is one; that is, external time is equal to internal time by default. In the case of slow play of streams, external time is longer than internal time; in the case of fast play, external time is shorter than internal time. Assuming that S1 and S2 are streams and that P is a predicate on frames of a stream, temporal composition of streams is achieved by temporal operators as follows:

(a) Tintersection (S1, S2) returns parts of S1 and S2 which temporally intersect.
(b) Tdifference (S1, S2) returns a part of S1 that does not temporally intersect with S2.
(c) Tunion (S1, S2) returns S1 and S2 ordered in time with possible overlaps.
(d) Tselect(S1, P) returns a part of S1 which satisfies P.
(e) Tjoin (S1, S2, P) = Tselect(Tunion(S1, S2), P).
(f) Tproject(S1, Func) returns the result of Func on S1,

where Func is an operation on frames that may include the spatial operators described below.

Note that internal time of a composite stream is the union of external time of its component streams. Real time of a composite stream is the union of real time of its component streams. For example, we assume that the query result of

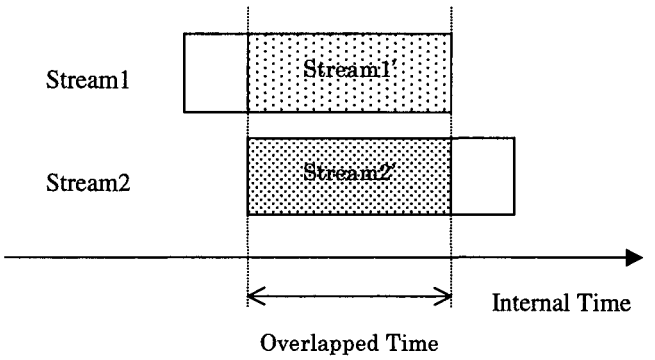


FIGURE 15 Schematic explanation of an expression Tintersection (Stream1, Stream2).

Script1(a) is scanned and is individually set to streams Stream1 and Stream2. To select only parts of Stream1 and Stream2 which temporally overlap one another, which realizes Script1 (b), we have only to execute an expression Tintersection (Stream1, Stream2) based on the internal time. Here we name the selected parts Stream1' and Stream2' for Stream1 and Stream2, respectively. The schematic explanation of the effect of the expression is presented in Fig. 15.

Similarly, space is divided into real space, internal space, and external space. Real space is space occupied by streams in the real world. Internal space is space typically occupied by streams in presentation. External space is space occupied by streams during the actual presentation of scripts, is specified by providing a magnification of internal space, and is equal to internal space by default. Assuming that F1 and F2 are frames and that P is a predicate on pixels of a frame, spatial composition of streams is accomplished by spatial operators as follows:

(a) Sintersection (F1, F2) returns parts of F1 and F2 that intersect in space.
(b) Sdifference (F1, F2) returns a part of F1 that does not intersect in space with F2.
(c) Sunion (F1, F2) returns F1 and F2 merged in space.
(d) Sselect(F1, p) returns a part of F1 that satisfies P.
(e) Sjoin (F1, F2, P) = Sselect (Sunion (F1, F2), P).
(f) Sproject(F1, Func) returns the result of Func on F1,

where Func is an operation on pixels.

Note that the internal space of a composite stream is the union of the external space of its component streams. The real space of a composite stream is the union of the real space of its component streams. For example, to arrange two frames Frame1 of Stream1' and Frame2 of Stream2' on the same window, which realizes Script1 (c), we evaluate an expression Sunion (Frame1, Frame2) based on the external space, whose effect is schematically explained in Fig. 16.

d. Control Operators

Processes, called agents, represent control structures. Processes consist of events, conditions, and actions. Event specification allows for serial, parallel,

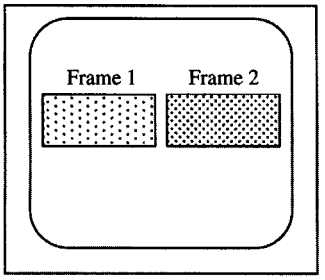


FIGURE 16 Schematic explanation of an expression Sunion (Frame1, Frame2).

and alternative occurrences of component events. Time is specifiable in events with keywords such as "before," "after," "between," "at," and "periodically." Events are invoked by actions within other processes. Conditions can monitor database states, process states, and QOS states. Actions specify control of processes, such as parallel, serial, and alternative, and other model operators, such as structural, temporal, and spatial. Concurrency is represented as annotation together with QOS options, which reduces to other model operators in the case of serial compilers. Serial specification is nothing but object-oriented programming and query languages. The merging of processes is specified by the conjunction of events. QOS is given as parameters to the process construct.

For example, two streams Stream1' and Stream2' executing in parallel, taking two QOS parameters latency and bit rate into consideration, is specified as follows based on the external time, which realizes Script1 (d):

```
QOS (Latency, Bit Rate);
Set1 = (Stream1', Stream2');
Set1.parallel.play;
```

The effect is schematically shown in Fig. 17. We describe how to satisfy QOS parameters later.

2. Overall Architecture

Now we describe the overall architecture of a multimedia database system. As shown in Fig. 18, the system architecture consists of agent management,

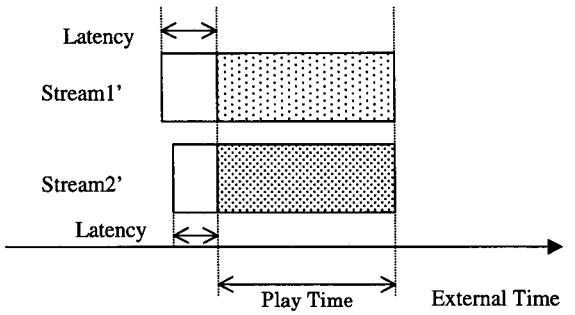


FIGURE 17 Schematic explanation of QOS-constrained concurrent playback of two streams.

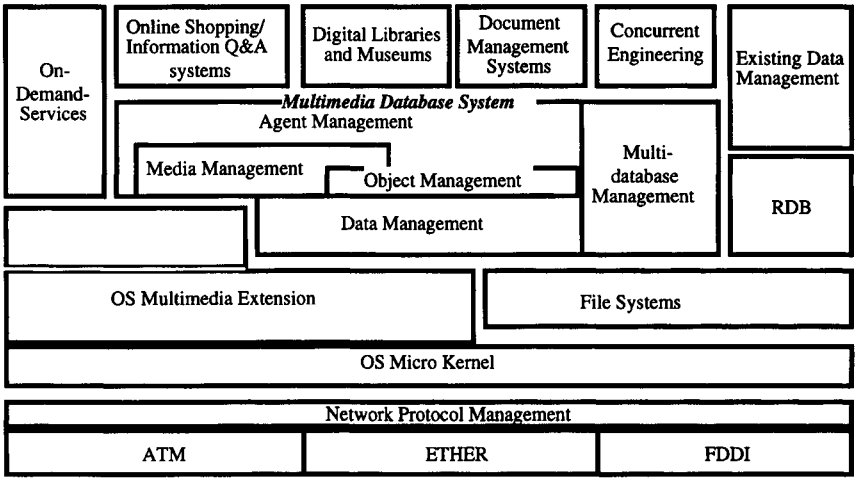


FIGURE 18 System architecture.

media management, object management, data management, and multidatabase management layers on top of OS and network protocol management layers. The agent management layer enables users to flexibly describe multimedia applications, as described in detail later. The media management layer provides interfaces to individual media data. The object and data management layers which provide basic database management facilities are based on Jasmine OODB. The multidatabase management layer provides integrated access to traditional media, such as numbers and character strings, in RDB and new media, such as audio and video streams, in OODB.

C. Our Approach to Implementation

1. Agents and Related Work

Before we describe our extended object-oriented approach based on agents to the implementation of a multimedia database system, we define our agents by extending object definitions described in Section II. In this paper, agents are defined as follows:

(1) Agents are basically objects in the context of OODB.
(2) Agents have at least one of the following characteristics:
  (2.1) Agents can be executed in a parallel or distributed fashion like processes. Conventionally, methods of objects are restricted to serial execution like subroutine calls.
  (2.2) Agents are mobile. Thus, agents move from one server to another like migrated processes. Agents may come back to the originating server or move further to a third server.
  (2.3) Agents can respond to events. Agents change behaviors in response to events incurred by the system, applications, or users.

Note that definitions of agents differ, depending on researchers. To clarify our approach, we strictly confine our definition of agents to the above

definition. In the following subsections, we describe agent-based approaches to the database issues discussed previously, focusing on multimedia data, that is, videos, texts, and program components, which are related to one another by explicit links or implicit links, such as keyword associations, similarity of contents, and temporal and spatial synchronization.

To solve all the database issues, we focus on the development of a database system that enables flexible and efficient acquisition, storage, access and retrieval, and distribution and presentation of large amounts of heterogeneous media data [22]. We take a realistic approach based on an OODB Jasmine [18, 20, 23], which is more suitable for the description of media structures and operations than a traditional RDB.

We propose a basic multimedia data model as an integration of structural, temporal, spatial, and control functionality. Our extended data model based on agents provides uniform interfaces to heterogeneous media in addition to defining structures and operations specific to such media. The model allows the representation of temporal and spatial relationships and of temporal synchronization and QOS control, through extending a *scripting language* suitable for multimedia application development. That is, we take an object-oriented database approach suitable for description of media structures and operations and extend the object-oriented approach by providing temporal and spatial operators, and control of distributed computing based on agents and QOS (quality of service). In particular, we adopt and *Event–Condition–Action* paradigm of active databases [19] to allow users to flexibly specify agent processes.

There are models that support temporal descriptions such as those of [8, 10]. Almost all of them focus only on temporal functionality and pay little attention to the other functionalities, such as spatial functionality. Our model provides set-oriented operators for associative (or partial) access to an ordered set of internal frames constituting streams, which are analogous to relational algebra [3] for associative access to an unordered set of records constituting tables. In general, control structures that support features such as concurrency are needed for the development of distributed multimedia applications. In our model, control is described as annotations to other functionality, i.e., structural, temporal, and spatial. In addition to concurrency control, QOS parameters such as latency and bit rates are specifiable in concurrency annotations for allowing real-time execution of stream media. We extend the annotated approach to concurrency to allow for QOS control options. QOS is controlled either by executing methods or by retrieving stored data. An Event–Condition–Action (ECA) paradigm of active databases [19] is applicable. However, it needs to be extended for real-time use. We use techniques such as prefetching and caching. In general, event specification facilitates relative invocation of synchronization or serialization of media objects. In particular, time events enable absolute invocation of media objects at specified times. Temporal relationships such as "before" and "after" are directly described by structural operators of the OODB. In summary, our multimedia data model is unique in that it allows concurrent object-oriented computing (i.e., agents) and QOS control for the development of distributed and real-time multimedia applications in addition to set-oriented temporal and spatial operators for associative access to media data unlike other data models.

2. Approach to Videos

Logical video contents need a semantic description of contents, that is, what are recorded. Moreover, logical contents are recorded in several ways, that is, in CODEC such as MPEG and Motion JPEG, and in quality such as frame sizes and rates. Long-duration play of whole video streams is not always required. Users should rather have partial access to video streams to jump to only their necessary portions. We allow users to access video streams with uniform interfaces independent of CODEC by using polymorphism of objects.

To facilitate interactive retrieval of multimedia, we enable users to flexibly and efficiently access partial data such as substreams of videos by temporal information (e.g., temporal intervals), keywords, and other related information. This technique of subsetting a large amount of media data is analogous to RDB views. Note that efficient processing of partial accesses is fully facilitated by combining software techniques such as access methods, clustering, and taking advantage of available hardware such as parallel servers. For example, Fig. 19 illustrates relationships among views, logical contents, and physical streams. A stream view selects a subset of logical contents by specifying a time interval. Keywords are attached to views for keyword-based retrieval. Characteristic data, such as figures, colors, and motion directions, are attached to frames of streams corresponding to views for content-based retrieval. Logical contents have several physical streams of different quality, which are chosen for appropriate QOS control in playback.

Unlike other approaches, we use a lightweight technique to segment scenes and recognize moving objects for content-based retrieval of stream data. First, the system detects scene cuts by using differences in successive frames, such as motion vectors of macro blocks of MPEG and colors. In MPEG coding, we abstract motion vectors of macro blocks by taking advantage of similarity between successive frames and make motion compensation by using such motion vectors. Motion compensation, however, becomes difficult at the point between successive different scenes. At that point, macro blocks with no motion compensation become dominant. So we detect cuts by checking such macro blocks. To enhance the precision of cut detection, we use the difference of colors between successive frames.

Then the user can define views of streams (i.e., substreams) by attaching keywords to such cut scenes. Keywords of stream views enable association between video data and other media data such as texts. Further, the user can

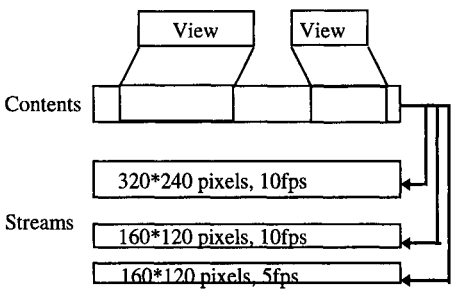


FIGURE 19 Views, contents, and streams.

define new views recursively by combining existing stream views. The system also chooses representative frames within a scene and abstracts characteristic data and stores them into databases. Please note here that matching with representative frames can reduce the recall ratio of a content-based query since characteristic data, such as colors and layouts, change even within a single scene.

The system also detects moving objects by using motion vectors of MPEG. The system decreases the number of colors to more accurately recognize moving objects. The system stores motion directions in addition to figures and colors associated with moving objects. Of course, the user can retrieve substreams corresponding to views with specified keywords. The user can further retrieve substreams containing samples of user-specified colors, figures, and motion directions. The system allows the users to retrieve video substreams containing user-specified moving objects without any interference from the background information because the system distinguishes between the moving objects and the backgrounds unlike other approaches such as QBIC [7]. Content-based retrieval is used by both end users and content providers.

Now we illustrate a scenario for content-based query by using scripts. Scripts allow specification of playback control such as parallel or serial and of layout of played streams. For example, a script for content-based retrieval is specified as follows:

```
Script2:
Set1 = VIEW from VIEW where VIEW.like (Sample1);
On Event Selection by User;
Set2 = Set1 from Set1 selected by User;
Set2.parallel.play;
```

Here the user specifies a sample, such as Sample1, through a GUI as shown in Fig. 20. A sample figure consists of several parts like a human body. The system abstracts characteristics data from the user-specified sample. The system uses the largest part, such as a human trunk, as a search key to a

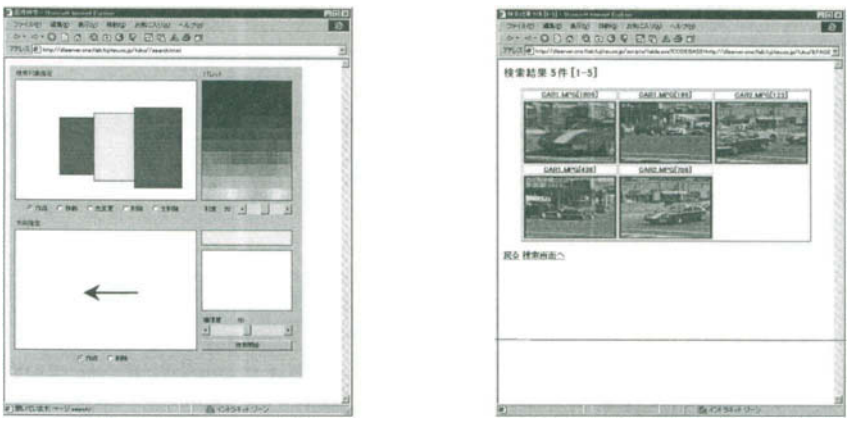


FIGURE 20 A GUI for content-based retrieval.

multidimensional index associated with characteristic data, such as R-tree and its variations [7]. In reality, we use eight indices for eight motion directions (up, down, left, right, and their middles). Each index has three dimensions corresponding to the three primary colors of objects moving in the same direction. The system selects an appropriate index for the user-specified direction and searches the primary key part against the index. The system evaluates the other parts, such as a head and legs, as additional conditions of a query by using the selected index. Of course, the user may explicitly specify favorite key parts other than the largest one. A content-based query evaluates to a set of views, such as Set1. Then the user chooses several views, such as Set2, among the resultant set for simultaneous playback.

We provide means for flexible distribution and presentation of retrieved multimedia data over distributed networks by executing QOS control and scripts. To this end, we use techniques, such as prefetching, caching, synchronization, and distributed processing.

In particular, our script scheduler [25] detects overlaps of intervals of view streams appearing in users' scripts and selects appropriate physical streams by using views to enforce QOS control, unlike other approaches. For example, if the user chooses three streams for parallel playback in Script1, the total playback time is divided into three intervals as shown in Fig. 21. The scheduler chooses physical streams of appropriate quality that can be played with available CPU resources within each interval. In other words, physical streams of different quality may be chosen for the same view such as a stream for View3. We first choose larger frame sizes and then larger frame rates. The system stores load factors obtained by rehearsal play of physical streams and GOP (group Of pictures) numbers and addresses for partial play.

Streams, such as videos, sounds, "streamfield" texts, and combinations of all, can be temporally synchronized with each other. Thus, all streams are guaranteed to be played without losing any frames if there no change occurs in the available CPU resources. Otherwise, the system skips frames which have missed deadlines.

We summarize an agent-based implementation of video data. We represent video streams as agents. As a result, video streams can be executed in parallel. As it takes much CPU load to transform video streams for special effects in general,

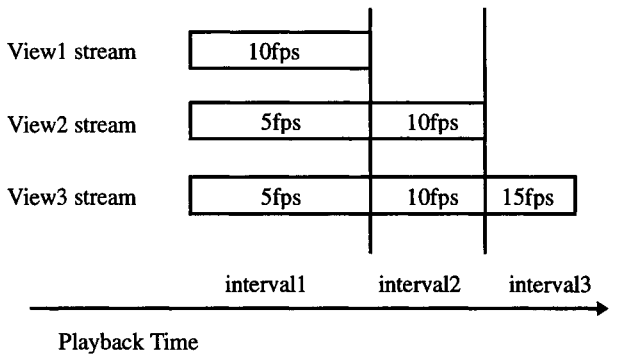


FIGURE 21 A script schedule example.

a stream to transform can be migrated to a different server for processing and only a result stream can be fed back the the originating client. Playback of streams is invoked by events such as the termination of other streams and user interactions.

We logically cluster video streams by using a multidimensional index on characteristic data. So we can allow users to do a similarity-based navigational search through the logically clustered video data. We allow users to store video data either in files, databases, or dedicated servers. We use databases to store logical data, such as keywords, time intervals, and physical data such as load factors, quality, GOP numbers. Bulk data are stored by either files or dedicated stream processors like VOD servers. We sequentially cluster streams in files. However, we decluster streams in dedicated stream servers for striping to provide wider IO bandwidth. We consider networked video applications such as digital libraries, VOD, NOD (news On-demand), and content management by providers.

3. Approach to Texts

Our database system allows users to acquire newly produced media data via distributed networks including ATM LANs and the Internet. Moreover, multidatabase functionality is provided to manage metadata (e.g., a directory) of existing data files or databases and to establish interoperable access to such files and databases. Our technology includes schema translation between OODB and RDB [7, 21, 29], uniform WWW gateways to databases, directory management by databases, and HTML (hypertext markup language) page management by databases. We consider digital libraries and on-line publishing as promising networked text applications.

Now we discuss an approach to text data management, focusing on HTML page management. First, mapping between text contents and formats such as HTML, SGML, and ODA is necessary. We resolve such heterogeneity by using polymorphism, too. Moreover, we need to allow users to acquire texts and reorganize them for further distribution. To this end, we provide HTML page management by databases including storage and retrieval. The system abstracts keywords from texts of collected HTML pages automatically and stores keywords and URL associated with texts into databases. Either HTML texts themselves, or only their file names and URL are stored in databases. In-line images as components of HTML texts are also stored by databases. The system adds URL, file names, titles, anchor character strings (i.e., links), and data types (e.g., GIF, JPEG) as default keywords. The user can delete or add favorite keywords. We prefer the recall ratio to the precision ration of keyword-based retrieval. Relatively addressed links (i.e., URL) are transformed to absolutely addressed links. The users can retrieve pages or components by a wide variety of keywords and drag and drop retrieved pages and components into work pages to create new home pages is a WYSIWYG fashion (see Fig. 22). Content-based retrieval of texts is facilitated by using a full text search engine.

Ease of data acquisition through WWW, however, makes the size of collected data unmanageable for the user. Keyword-based retrieval alone is not sufficient. So we logically cluster texts for similarity-based navigation. The system automatically abstracts keywords from collected HTML or SGML

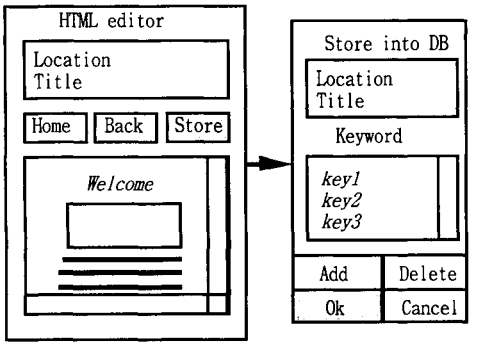
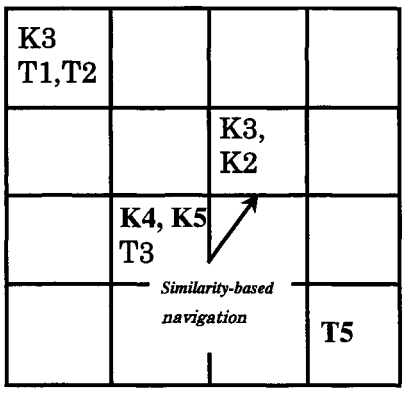


FIGURE 22 An HTML editor session.

texts. Then the system chooses the most frequent 100 keywords contained by a set of texts and places each text in the information space of 100 axes ranging from having the corresponding keyword to not having it. The system uses a *Self-Organizing Map* (SOM) [28] technique to logically cluster a set of collected texts into the given number of groups in the retrieval space. The system displays the structured map by using 2-D or 3-D graphics such as VRML. The user can retrieve texts by navigating a 2-D or 3-D user interface. T*i* and K*i* in Fig. 23 denote texts and keywords, respectively. The point is that the users cluster collected texts for their own use. Of course, content providers can use this technique when they cluster their own texts in advance.

We briefly describe how we have applied the SOM technique to logical text clustering. Input patterns to the information space, that is, texts have normalized characteristic vectors $Vi\,[vi1, \ldots, viM]\ i = 1, \ldots, N$. We choose 100 for M, that is, the most 100 frequent keywords. N denotes the total number of texts and is 100 for the moment. If a value of vij is 1, an input text i has a keyword



Ki: keyword
Ti: text

FIGURE 23 An SOM retrieval map.

key j; if the value is 0, then the text has no such keyword. On the other hand, grid patterns in the two-dimensional retrieval space Pij are assigned appropriate characteristic vectors as initial values $Vpij$ in the information space. For example, we use 10-by-10 grid patterns with a torus nature. Here we define similarity SIM between characteristic vectors Vi and Vj, distance DIS between them, length of a vector LEN, and vector operations PLUS, MINUS, DIV as follows:

SIM (Vi, Vj): sum (vik * vjk) $k = 1, \ldots, M$.
DIS (Vi, Vj): square-root (sum (square($vik - vjk$)) $k = 1, \ldots, M$).
LEN (Vi): square-root (sum (square(vik)) $k = 1, \ldots, M$).
PLUS (Vi, Vj): [($vik + vjk$)] $k = 1, \ldots, M$.
MINUS (Vi, Vj): [($vik - vjk$)] $k = 1, \ldots, M$.
DIV (Vi, C): [(vjk/C)] $k = 1, \ldots, M$.

Procedure (1). First, we select an input pattern Vi which has the maximum similarity with a grid pattern Pij and we move Pij, that is, its characteristic vector $Vpij$, closer to Vi in the information space. New $V'pij$ is calculated as

V' = PLUS ($Vpij$, DIV (MINUS (Vi, $Vpij$), exp (A*R*R)))
$V'pij$ = DIV (V', LEN (V')); normalization,

where $A < 1$ and R = DIS (Vi, $Vpij$).

Next we move grid patterns in the neighborhood of Pij (i.e., $Pij+1$, $Pij-1$, $Pi+1j$, $Pi-1j$, $Pi+1j+1$, $Pi+1j-1$, $Pi-1j+1$, $Pi-1j-1$) closer to Vi at that time.

We repeat PROCEDURE (1) until the maximum similarity between grid patterns and input patterns exceeds a limit. For the moment, we choose A such that repetition times is less than 10. After the termination of PROCEDURE (1), each input pattern is mapped to its nearest grid patterns in the retrieval space. We also apply PROCEDURE (1) to keywords, which are represented as characteristic vectors having only one nonzero element. Thus, we can map input patterns and keywords to the retrieval space with holding their topological relationships in the information space.

Once the clustering by SOM is completed, we map new input patterns to grid patterns whose vectors are most similar to those of new input patterns unless the total number of input patterns exceeds a limit. If the total number exceeds the limit, we recluster all the input patterns. We can avoid too frequent clustering by implementing the clustering as an agent. The clustering agent responds to the event that the total number of input patterns exceeds a limit (N *n, $n = 1, 2, \ldots$).

We can apply the SOM technique to cluster videos and components programs, too. If we apply the SOM to a mixture of videos, texts, and program components, we get hypermedia links among heterogeneous media data, based on similarity of contents.

Generally, we provide media data management mechanisms which enable efficient storage and access of large amounts of media data. They enable users to customize media-specific storage in aspects such as indexing, clustering, and buffering. We take an object-oriented approach to resolving heterogeneity in

data formats, CODEC, and physical media, used for implementation of logical media.

Here we describe our physical clustering facility by taking structured texts or compound documents consisting of simple texts, graphics, images, and videos. Structured texts such as SGML texts are often accessed according to component links. The system clusters relevant texts in the same or neighborhood pages to allow efficient retrieval of them. We assume that the user chooses to cluster texts. Thus, the user specifies how data are clustered. Then the system actually clusters data according to the user's specification. In future, we plan to provide a facility to monitor hot spots of access patterns. Either the system or the user clusters data based on the result of monitored accesses. We allow the user to recluster data after heavy updates. In addition to heterogeneous clustering, we allow homogeneous clustering such as all instances of the same class. We allow subtrees of a whole component tree to be flexibly clustered according to the user's specification by combining homogeneous and heterogeneous clustering.

To implement physical clustering, we introduce two types of pages. We usually use a single page to store instances of the same class, that is, for homogeneous clustering. We devise a multiple page to store instances of heterogeneous classes, that is, heterogeneous clustering. A multiple page consists of single pages. Each of them has its own size and contains instances of its own class. Multiple pages as a whole are mapped to contiguous space.

4. Approach to Program Components

To allow for access control of digital libraries, we must maintain integrity of programs for database access. Further, we must maintain program components for software reuse such as digital library application development. We must divide processing between clients and servers. The retrieved result is rather large, so we must adopt protocols sufficient for transfer of bulk data different from HTTP.

To this end, we allow users to execute or download programs on-line such as database access and to download program components for application development. So we manage component programs such as Java Applets and ActiveX Controls by maintaining integrity of programs, that is, consistent combinations of component programs. Programs can be executed on-line and visually.

Program components are managed by database servers. The system maintains information such as consistent combination or configuration of programs. When the user requires programs, the system compares the user's configuration and the server's current configuration and then sends only differences by looking up the configuration version and the user level stored by databases.

Programs are updated and further moved from one server to another. Programs have physical aspects such as program names, program files (or OID), and versions. Programs have logical aspects such as interfaces (i.e., signatures). So we provide program views as mapping between physical and logical aspects of programs. We can conceal changes of physical aspects from users by using such program views. Integrity maintenance is done either by eager propagation of updates using an ECA paradigm or by deferred propagation of updates using a time stamp.

The system processes programs by distributing processes between clients and servers specified by scripts. For example, client processes take care of a graphical user interface and server processes take care of database access. The query result is sent back to the client by the database system's protocols.

In addition to direct access of program components, a query over components by keywords is allowed. If a user's query for program components is not satisfied by one server, the query is federated to other servers. We take an agent-based approach to such a federation. The search agent also uses the SOM technique to construct a map for navigating the query through relevant servers, which is a new approach. The query retrieval is sent to the mail box specified by the client, from which the client obtains program components.

The system automatically abstracts keywords for program components from their signatures and explanatory texts and logically clusters program components based on such keywords using the SOM technique. As a result, we obtain an *intraserver map* for similarity-based search, which is more useful in a program query than the simple keyword-based query facility.

A map for guide in the navigation of servers is constructed from the results of application of the SOM technique to a set of component servers. Thus, keywords selected for an intraserver map when the SOM is applied to cluster program components within local servers are considered as characteristic data of local servers. Then we cluster servers by again using the SOM to get a *interserver map*. The interserver map associates keywords and servers, instead of texts. The system can find most relevant servers using the interserver map. Please note that the interserver map by the SOM can be constructed for videos and texts, too. Moreover, physical clustering applied to structured texts can also be applied to component programs through caller–callee relationships between program components like parent–child relationships between structured texts.

We summarize an agent-based implementation of program component management. Selected programs can be executed *in situ* using distributed object management such as CORBA, although they can be downloaded to client sites like Java Applets. So component programs are agents executable in a distributed manner. The search process moves from server to server via the interserver map for federating a query. Thus, the search process has agent characteristics. Integration maintenance by ECA responds to the event of updates as an agent process.

D. Conclusion

In this section, we have proposed a multimedia database system for networked multimedia applications such as digital libraries and document warehousing, based on an OODB model extended with concepts of agents. We have implemented an early prototype multimedia database system to verify the proposed approach. This prototype supports multimedia scripting, keyword-based and content-based multimedia view retrieval with QOS control, SOM-based logical data clustering, heterogeneous physical data clustering, and WWW/database integration. It has been applied to in-house digital libraries and has proved to be effective. We plan to enhance the functionality and performance of our system in order to make the system applicable to industrial applications.

V. CONCLUSION

First, in this paper we described a prototype object-oriented DBMS called Jasmine, focusing on the implementation of its object-oriented features. Jasmine shares a lot of functionality with other object-oriented database systems. However, Jasmine has the following features that differentiate it from other systems. Jasmine provides a powerful query language that allows users to specify complex objects, class hierarchies, and methods in queries. Jasmine optimizes such object-oriented queries by using hash joins, B-tree and hash indexes, and semantic information. Individual object access is evaluated on object buffers. Jasmine extends relational database technology. Jasmine provides nested relations to efficiently manage complex objects and provides user-defined functions evaluated on page buffers to efficiently process method invocation in queries. Jasmine provides a view facility for schema integration and a constraint management facility including integrity constraints, triggers, and rules. We compare Jasmine with current commercial object-oriented database systems and research prototypes as follows.

GemStone [32] originates from the attempt to make Smalltalk-80 programs databases. The GemStone data model is based on Smalltalk-80 and supports only single inheritance while Jasmine supports multiple inheritance. In addition to C, C++, and Smalltalk-80 interfaces, GemStone provides a programming interface called OPAL. GemStone distinguishes between a class and a collection of objects. A query expressed by OPAL is formulated against a single collection of objects. A Jasmine query is formulated against classes, allowing explicit joins.

ORION [26] supports a variety of functions, such as multiple inheritance, composite objects, versions, queries, and schema evolution. ORION is built in Lisp on a secondary storage system that provides facilities for segment and page management. ORION provides a programming interface to an object-oriented extension of Lisp. A query returns a collection of instances of a single class while a Jasmine query can generate instances combining more than one class. Mapping object identifiers to pointers is done by extensible hashing. A query with attributes of nonleaf classes is processed by use of a class-hierarchy index unlike Jasmine. ORION evaluates a query against the object and page buffers and merges the results while Jasmine uses the single-evaluation scheme. ORION uses sort-merge joins while Jasmine uses hash joins.

In O2 [5], an object contains a value, a list, a set, and a tuple as an attribute value. O2 is used through an object-oriented extension of C called CO2. The query language is defined rather formally. The query retrieves and composes a list, a set, and a tuple. O2 is implemented on top of WiSS (Wisconsin Storage System) in C. WiSS provides persistency, disk management, and concurrency control for flat records. Unlike Jasmine, O2 uses physical identifiers of WiSS records as object identifiers. Like ORION, O2 adopts a dual buffer management scheme. Like Jasmine, O2 uses a hash table to manage in-memory objects, but unlike Jasmine, O2 uses a class-hierarchy index to process queries against nonleaf classes.

In IRIS [31], based on the DAPLEX functional model, properties or methods defined by a class are represented as functions on the class. Functions are

stored or derived from other functions. IRIS supports multiple inheritance, versions, schema evolution, and queries. Query optimization is done by rule bases. Unlike Jasmine, IRIS is implemented on a relational storage system that supports only flat relations. IRIS has C and Lisp interfaces, but supports no integration with object-oriented programming languages while Jasmine does.

Next we discussed an object-oriented database approach to engineering as an advanced application. Then we described schema translation by view as an extension to Jasmine. OODBs have just been developed, so there are very few reports of real applications. We would like to apply our OODB Jasmine to various real-world problems, not only to verify the validity of our approach but to also give feedback to Jasmine from the experiences.

Our future plans include research on technical issues associated with exploratory aspects of advanced applications such as design: The incorporation of version management, constraint management, and view management in a heterogeneous environment. Version management is mandatory for exploratory applications, but concepts of versions differ from application to application. It is important to propose generic concepts of versions from which specific versions can be derived and to include both instance and class versioning. To explore design alternatives and propagate updates, we must incorporate generalized constraint management including constraint satisfaction rules and composite events [19]. To support cooperative exploration in a heterogeneous environment consisting of relational and object-oriented systems, we must provide more advanced view support that allows the user to look at schemas defined by other users in other systems (e.g., relational systems) differently from the original ones as if they were object-oriented.

REFERENCES

1. Batory, D. S., Leung, T. Y., and Wise, T. E. Implementation concept for an extensible data model and data language, *ACM Trans. Database Syst.* **13**(3): 231–262, 1988.
2. Carey, M. J., Dewitt, D. J., and Vandenberg, S. L. A data model and query language for EXODUS. In *Proc. of the 1988 ACM SIGMOD Conference*, Chicago, IL, June 1988, pp. 413–423. ACM, New York, 1988.
3. Date, C. J. *An Introduction to Database Systems*, Vol. 1. Addison-Wesley, Reading, MA, 1990.
4. Debloch, S. *et al.* KRISYS: KBMS support for better CAD systems. In *Proc. of the 2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, Gaithersburg, MD, Oct. 1989, pp. 172–182. IEEE, Los Alamitos, CA, 1989.
5. Duex, O. *et al.* The story of O2. *IEEE Trans. Knowledge Data Eng.* **2**(1): 91–108, 1990.
6. Fishman, D. H. *et al.* IRIS: An object-oriented database management system. *ACM Trans. Office Inform. Systems* 5(1): 48–69, 1987.
7. Flickner, M. *et al.* Query by image and video content: The QBIC system. *IEEE Computer* **28**(9): 23–32, 1995.
8. Gibbs, S., Breiteneder, C., and Tsichritzis, D. Data modeling of time-based media. In *Proc. of ACM Sigmod Conference*, May 1994, pp. 91–101.
9. Goldberg, A., and Robson, D. *Smalltalk-80: The Language and Its Implementation.* Addison-Wesley, Reading, MA, 1983.
10. Hamakawa, R., and Rekimoto, J. Object composition and playback models for handling multimedia data. *ACM Multimedia Systems* **2**: 26–35, 1994.
11. Ishikawa, H., Izumida, Y., Yoshino, T., Hoshiai, T., and Makinouchi, A. A knowledge-based

approach to design a portable natural language interface to database systems. In *Proc. of the IEEE Data Engineering Conference*, pp. 134–143. IEEE, Los Alamitos, CA, 1986.
12. Ishikawa, H., Izumida, Y., Yoshino, T., Hoshiai, T., and Makinouchi, A. KID: Designing a knowledge-based natural language interface. *IEEE Expert* 2(2): 57–71, 1987.
13. Ishikawa, H., Suzuki, F., and Makinouchi, A. Object-oriented multimedia knowledge base management system: Design and implementation. In *Proc. of the 2nd International Symposium on Interoperable Information Systems,* Tokyo, Japan, Nov. 1988, pp. 195–202. INTAP, Japan, 1988.
14. Ishikawa, H. An object-oriented knowledge base approach to a next generation of hypermedia system. In *Proc. of the 35th IEEE COMPCON Conference*, San Francisco, CA, pp. 520–527, IEEE, Los Alamitos, CA, 1990.
15. Ishikawa, H., Izumida, Y., and Kawato, N. An Object-oriented database: System and applications. In *Proc. of the IEEE Pacific Rim Conf. Communications, Computers, and Signal Processing,* Victoria, B.C., Canada, pp. 288–291. IEEE, Los Alamitos, CA, 1991.
16. Ishikawa, H. *The Design and Implementation of an Object-Oriented Database System for Advanced Applications.* Ph.D. Thesis., University of Tokyo, 1992.
17. Ishikawa, H. *et al.* An object-oriented database system and its view mechanism for schema integration. In *Proc. of the Second Far-East Workshop on Future Database Systems*, Kyoto, Japan, April 1992, pp. 194–200.
18. Ishikawa, H. *et al.* The design and implementation of an object-oriented multimedia knowledge base management system. *ACM Trans. Database Systems* 18(1): 1–50, 1993.
19. Ishikawa, H., and Kubota, K. An active object-oriented database: A multi-paradigm approach to constraint management. In *Proc. of the 19th VLDB Conference (Dublin, Ireland),* Aug. 1993, pp. 467–478. VLDB endowment.
20. Ishikawa, H. *Object-Oriented Database System.* Springer-Verlag, Berlin, 1993.
21. Ishikawa, H. *et al.* A script-based approach to relational and object-oriented database interoperability. In *Proc. of Intl. Symposium on Advanced Database Technologies and Their Integration,* Oct. 1994.
22. Ishikawa, H. *et al.* A next-generation industry multimedia database system. In *Proc. of IEEE 12th Intl. Conference on Data Engineering,* pp. 364–371, 1996.
23. Ishikawa, H. *et al.* An object-oriented database system Jasmine: Implementation, application, and extension. *IEEE Trans. Knowledge Data Eng.* 8(2): 285–304, 1996.
24. Ishikawa, H. *et al.* An extended object-oriented database approach to networked multimedia applications. In *Proc. of IEEE 14th Intl. Conference on Data Engineering,* pp. 259–266, 1998.
25. Kato, K., Kondo, A., and Ishikawa, H. Multimedia database infoServer—Script and video playback. In *Proc. of the 7th Data engineering Workshop*, pp. 109–114, 1996. [In Japanese.]
26. Kim, W. *et al.* Architecture of the ORION next-generation database system. *IEEE Trans. Knowledge Data Eng.* 2(1): 109–124, 1990.
27. Kitagawa, H., and Kunii, T. L. *The Unnormalized Relational Data Model for Office Form Processor Design.* Springer-Verlag, Tokyo, 1989.
28. Kohonen, T. *Self-Organizing Maps.* Springer-Verlag, Berlin, 1995.
29. Kubota, K., and Ishikawa, H. Structural schema translation in multidatabase system: Jasmine/M. In *Proc. of IPSJ Advanced Database Symposium,* 1994. [In Japanese.]
30. Larson, P.-A. Linear hashing with partial expansions. In *Proc. of the 6th VLDB Conference,* Montreal, Canada, 1980, pp. 224–232. ACM, New York, 1980.
31. Lyngbaek, P., and Vianu, V. Mapping a semantic database model to the relational model. In *Proc. of the 1987 ACM SIGMOD Conference*, San Francisco, CA, 1987, pp. 132–142. ACM, New York, 1987.
32. Maier, D. *et al.* Development of an object-oriented DBMS. In *Proc. of the 1st OOPSLA Conference,* Portland, OR, 1986, pp. 472–482. ACM, New York, 1986.
33. Morgenstern, M. Active databases as a paradigm for enhanced computing environments. In *Proc. of the 9th VLDB Conference*, Florence, Italy, Oct. 1983, pp. 34–42. VLDB endowment, 1983.
34. Special Issue: Digital Libraries, *CACM* 38(4): 1995.
35. Stefik, M., and Bobrow, D. G. Object-oriented programming: Themes and variations. *AI Magazine* 6(4): pp. 40–62, 1986.

36. Yamane, Y. A hash join technique for relational database systems. In *Proc. of the Foundation of Data Organization Conference,* Kyoto, Japan, May 1985, pp. 388–398.
37. Yamane, Y. *et al.* Design and evaluation of a high-speed extended relational database engine, XRDB. In *Proc. of International Symposium on Database Systems for Advanced Applications,* Seoul, Korea, April 1989, pp. 52–60.
38. Zloof, M. Security and integrity within the query-by-example data base management language. IBM Research Report RC6982, Feb. 1978.

4

QUERY OPTIMIZATION CONCEPTS AND METHODOLOGIES IN MULTIDATABASE SYSTEMS[1]

CHIANG LEE

Institute of Information Engineering, National Cheng-Kung University, Tainan, Taiwan, Republic of China

In a multidatabase system (MDBS), the participating databases are autonomous. The schemas of these databases may be different in various manner even though the same information is represented. The execution of a global query in this environment involves levels of translations. Query optimization can be partially achieved in each of these levels. We first discuss in this chapter the past research at these optimization levels. A problem with translation is that it lacks a convenient representation of the integrated

[1]This work was supported by the National Science Council under Grant NSC85-2213-E-006-018.

schema at the system level and a sound mathematical basis for data manipulation in a multidatabase system. To resolve this problem, we present the concept of hyperrelation and use it as a powerful and succinct model for the global level representation of heterogeneous database schemas. A hyperrelation has the structure of a relation, but its contents are the schemas of the semantically equivalent local relations in the databases. With this representation, the metadata of the global database and local databases and the data of these databases are all representable by using the structure of a relation. The impact of such a representation is that all the elegant features of relational systems can be easily extended to multidatabase systems. A hyperrelational algebra is designed accordingly. This algebra is performed at the MDBS level such that query transformation and optimization is supported on a sound mathematical basis.

Another most critical level of optimization is at the execution strategy level. Difficulties of optimization at this level is that each participating database system does not own the information (data) and the mechanism (software) required for converting data of one database to another to resolve the data-type conflict problems. More importantly, this confines the processing of an intersite operation (such as a join over two relations of different databases) to within the MDBS only. The participating database systems are not able to share the workload of the MDBS in this environment. Hence, how to minimize the consumption of MDBS resources is an urgent problem. In the second part of this chapter, we present three scheduling algorithms that are used in an MDBS to reduce the processing cost of a multidatabase query. A major difference between our strategies and the past methods is that ours does not require the regeneration of the cost models of the participating databases. Hence, it also minimizes the indeterminacy existing in multidatabase query optimizations.

I. INTRODUCTION

A multidatabase system is a system that manages databases as a repository resource and allows application programs to access this resource in a heterogeneous distributed environment [1, 3, 13, 39, 40, 69, 74, 79]. In such an environment, query processing is very time-consuming as multiple levels of conversions (of data as well as the query) among heterogeneous systems are required. Query optimization becomes an especially important task in order to reduce the query processing time. This issue, however, has attracted far less attention from researchers than it deserves. Looking into those approaches proposed, we see that the multidatabase query optimization issues were studied from different levels of a multidatabase system. They include:

Schema level. This is the highest level of optimization and the least-studied level in multidatabase systems. A query may be expressible in completely different forms in different databases, as schemas of different databases are often represented in different manners. As a result, selecting a proper site will facilitate the execution of a query. This allows us to achieve query optimization at the multidatabase schema level. Studying the effect of different schemas (i.e., representations of data) on the cost of query execution is the focus of research at this level. References [46, 49] are the only works found in the literature proposing solutions for issues in this category. More research in this area is needed.

Semantic query optimization level. It intends to reduce query execution cost through the elimination/reduction of operations or reduce query search space by utilizing the semantic knowledge of databases (e.g., the sum of a person's salaries in all databases is the person's total salary). Reference [64] gives an example of research in this category. However, only aggregate operations (such as *sum, avg, max, min*) have been discussed so far in current research.

Algebra level. There are also approaches utilizing algebraic transformation for query optimization. Similar to centralized databases, a query is expressed in an algebraic expression. By using transformation rules, the expression is representable as a set of equivalent expressions. Optimization is achieved by choosing one of the expressions that incurs the least processing cost. The algebra, however, is an extension of the relational algebra. These researches include a multirelational algebra [33] for loosely coupled multidatabase systems (those without an integrated schema) and a hyperrelational algebra [48] for tightly coupled multidatabase systems (those with an integrated schema).

Execution strategy level. The goal of this level of optimization is to estimate the processing time for relational operations (mainly join) by figuring out the *cost model* of a given DBMS [24, 61, 83, 86]. Whether the join is performed by a hash, a sort-merge, or a nested loop algorithm, and whether a relation is indexed/clustered are the key factors considered in their research. Based on the estimated cost models of the participating databases in a multidatabases environment, the multidatabase management system (MDBS) is able to determine the most suitable database(s) to process the query. Du *et al.* [25] later proposed strategies for reducing the response time of query processing in multidatabase systems. Although an autonomy of systems is considered in these approaches, representation discrepency of data in different systems are not taken into account. Reference [50] relaxes this assumption and proposes solutions for query optimization considering data heterogeneity.

Operation level. Outerjoin and outerunion are frequently used operations in multidatabase systems. Reference [11] studies the optimization of an outerjoin operation. An outerjoin of R and S can be expressed as the union of three components: (1) the join of R and S, (2) the dangling tuples of R padded with null values, and (3) the dangling tuples of S padded with null values. Reference [11] identifies the situations when an outerjoin can be reduced to a one-sided (i.e., left or right) outerjoin or even a regular join operation when there is no data inconsistency. Reference [63] extends the result to situations when data inconsistency exists.

In this chapter, we discuss the optimization issue at the two most important levels: the algebra level and the execution strategy level. They are the two levels easiest to implement and most influential in terms of improving the performance. The algebra level optimization is important because by applying algebraic transformation to a query, we can derive the equivalent expressions of the query. They all lead to the same query result but incur different exeuction costs. Hence, optimization at the algebra level is a key step in every database management system. Optimization at the execution strategy level is crucial due to a very similar reason: it is a necessary component in a query optimizer. By

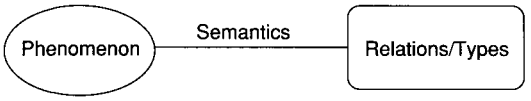


FIGURE 1 Semantics.

generating and evaluating various execution strategies, a system is able to find an optimal plan for a query execution. The content of this presentation is based on our previous work published in [48] and [50].

The rest of this chapter is organized as follows. In Section II, we depicts the concepts of semantic discrepency and the various types of schema conflicts. In Section III, we present the optimization at the algebra level. A hyperrelational algebra is presented accordingly. The algebraic transformation rules are also discussed in detail, and a comparison of this algebra with another multirelational algebra is given in this section. In Section IV, we discuss the optimization at the execution strategy level. A few strategies are presented and their cost models and performance analysis are discussed. Finally, the conclusion of this chapter is given in Section V.

II. SEMANTIC DISCREPANCY AND SCHEMA CONFLICTS

A. Semantic Discrepancy

Most of the information systems provide a set of constructs to model data of the real world. Each of these constructs is considered as a type, which may be a complex type (usually defined by the user) or a primitive type (normally provided by a system). The connection between the type and the data represented by that type is called the semantics. According to *The Oxford English Dictionary*[2] semantic (semasiology) is "the relationships between linguistic symbols and their meanings, or the interpretation of signs in general." In addition, the *Webster's Third New International Dictionary*[3] says that semantics is "the study dealing with the relations between signs and what they refer to, or the study of the relations of a sign to its referent and to other signs within a system."

In relational database systems, relations are the only user-defined types used to model the real world phenomena, and the relationships between the types and the referents decide the semantics of the relations, as shown in Fig. 1.

Two relations are said to be semantically equivalent if the relationships between the types and their corresponding referents are the same. For example, Fig. 2 shows two relations *Student* and *Stud* used in distinct databases to model students of two universities. They are considered semantically equivalent because they both model the information about students (i.e., the types and their corresponding referents are the same), no matter whether the sets of students are the same.

On the other hand, even if the sets of referents of two relations are the same, it does not necessarily imply that the two relations are semantically equivalent. For example, in Fig. 3, two databases model the same set of persons from

[2]Second Edition, Volume XIV, Clarendon Press, Oxford.

[3]Merriam-Webster, Incorporated, 1976.

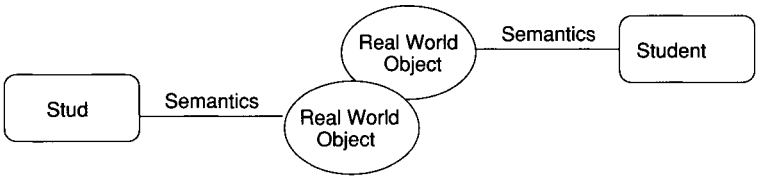


FIGURE 2 Two semantically equivalent relations.

different viewpoints. One models them as *Student*, while the other models them as *Employee*. They are not semantically equivalent because the relationships between the types and their referents (*Student* and *Employee*) are different.

Semantically equivalent relations need not have the same set of attributes. For example, in Fig. 4 the two relations *Student* and *Stud* are semantically equivalent, but their sets of attributes are not equivalent. The address information in *Student* is represented as *street, city, state*, but represented as *address* in *Stud*. Also, *Student* has *birthday*, whereas *Stud* has *gender*.

B. Schema Conflicts

Various types of conflicts can exist between schemas of relations in different databases. Since *value, attribute,* and *table* are the three major components of a database, the conflicts can be roughly classified into six types: *value-versus-table, value-versus-attribute, value-versus-value, attribute-versus-table, attribute-versus-attribute*, and *table-versus-table* conflicts [49]. An *X*-versus-*Y* conflict means that an *X* in one database is represented as a *Y* in another database. A multidatabase UNIV-DATABASE will be used as an example for illustration. The database consists of component databases of four universities and colleges as shown in Fig. 5.

1. Value-versus-Value Conflict

The value-versus-value conflict arises when different types of values are used in the same type of attributes. This type of conflict includes *expression conflicts, data unit conflicts*, and *precision conflicts*. Suppose that one database use {`Excellent, Good, Fair, Poor, Bad`} to represent the score of a student, while {`1, 2, 3, 4, 5`} is used in another database. We say that there exist expression conflicts between these two databases. A unit conflict exists if in one database the height of a student is measured in centimeters and in another it is in feet. A precision conflict exists if the scores of students fall in the range {`0--100`} in one database and {`0.0--100.0`} in another database.

In the literature, these problems are also referred to as the domain mismatch problems. Many works have been proposed to resolve these problems [20, 45, 55, 56, 75, 78].

FIGURE 3 Two semantically different relations.

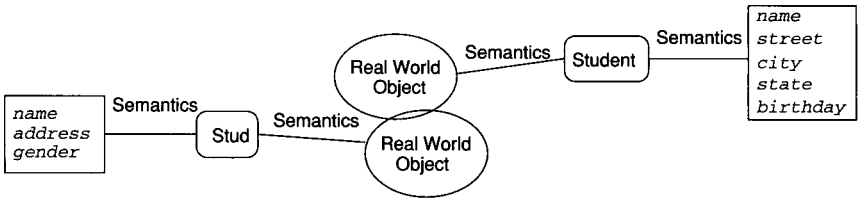


FIGURE 4 Information provided by databases.

2. Attribute-versus-Attribute Conflict

This type of conflict occurs when the relations in component databases use different numbers of attributes to represent the same information. For example, in CDB1 the addresses of the students are stored in one attribute Addr, while in CDB2 and CDB3 the same information is stored in three attributes (No, Str, State) and (Str, City, State), respectively. This type of conflict can be further classified into the following subtypes: *one-to-zero*, *one-to-one*, *one-to-many*, and *many-to-many* conflicts. A one-to-zero conflict is a *missing attribute* conflict, meaning that some attributes in a table do not appear in any form in the corresponding semantically equivalent table(s) of another database. For example, both the tables Stud of CDB1 and CS_Stud of CDB2 store data about students. However, the attribute Gender is missing from the table CS_Stud. A many-to-many attribute conflict means that two different sets of attributes are

University Database	*Relation Name*	*Attributes*
CDB1 (Uni. A)	Stud	(S#, Name, Addr, Tel#, Gender, Class)
	Course	(C#, Name, Credit, Type)
	Take	(S#, C#, Grade)
	Faculty	(SSN, Name, Addr, SpouseName)
CDB2 (College. B)	CS_Stud	(SN, FN, LN, No, Str, State, Tel#, Grade)
	Math_Stud	(SN, FN, LN, No, Str, State, Tel#, Grade)
	EE_Stud	(SN, FN, LN, No, Str, State, Tel#, Grade)
	Fndmntl_Crs	(CN, Name, Credits)
	Advncd_Crs	(CN, Name, Credits, Prereq-C#)
	Participate	(SN, CN, Score)
	Faculty	(F-SSN, Name, Addr, S-SSN)
	Spouse	(S-SSN, Name)
CDB3 (Uni. C)	Studs	(SSN, SNo, Name, Str, City, State, Gender, Class)
	Lecture	(CNo, Name, Credit)
	Tutorial	(CNo, Name, Credit)
	Seminar	(CNo, Name)
	Indvl_stdy	(CNo, Name)
	Takes	(SSN, CNo, Grade)
CDB4 (College. D)	ST1	(S#, Name, Addr, Tel#, Sex)
	ST2	(S#, Name, Addr, Tel#, Sex)
	ST3	(S#, Name, Addr, Tel#, Sex)
	ST4	(S#, Name, Addr, Tel#, Sex)
	Courses	(C#, Name, Credit, Hours, Fndmntl, Advncd)
	Enroll	(S#, C#, Score)

FIGURE 5 Component database schemas of four universities.

used to model the same information in two databases. Actually, the one-to-one and the one-to-many attribute conflicts are only special cases of the many-to-many conflict.

3. Attribute-versus-Table Conflict

This type of conflict occurs when an attribute in a relation of one database is represented as a table in another database. For example, in Fig. 5 the attribute SpouseName in relation Faculty of CDB1 stores the names of the spouses of the faculties, while in CDB2 the same information is modeled as a relation Spouse. As relevant information is stored in separate relations (e.g., Faculty and Spouse of CDB2), a foreign key is used in the Faculty of CDB2 to link to the associated Spouse information.

4. Value-versus-Attribute Conflict

In the multidatabase example, Course in CDB1 and Courses in CDB4 are two tables storing the data about the courses provided in the universities. They include the course number, the course name, the credits, and the type of courses. Suppose that the domain of Type of the courses in CDB1 is {fundamental, advanced}, and in CDB4 fundamental courses are further separated into lectures and tutorials, and advanced courses are also divided into seminars and individual studies. Then the value of attribute Type is expressed as attributes Fndmntl and Advncd in CDB4. This conflict is called the value-versus-attribute conflict, since the values in one database are treated as attributes in another database.

5. Value-versus-Table Conflict

In CDB1, student information is kept in one relation Stud, while in CDB2 they are distributed into three relations CS_Stud, Math_Stud, and EE_Stud, and in CDB4 the data are distributed into four relations ST1, ST2, ST3, and ST4. The three relations CS_Stud, Math_Stud, and EE_Stud store data about students majoring in "Computer Science," "Mathematics," and "Electronic Engineering," respectively. Also the four relations ST1, ST2, ST3, and ST4 store data about freshman, sophomore, junior, and senior students, respectively. These conflicts are considered as value-versus-table conflicts, as the values of an attribute in one database (e.g., Stud.Class of CDB1) are represented as the names of tables in another database, or equivalently, the data in one table are the metadata of another table. This type of conflict together with the value-versus-attribute conflict are often referred to as the types of conflicts between data and metadata.

6. Table-versus-Table Conflict

The table-versus-table conflict can be explored via two dimensions—*difference in number of tables* and *difference in the set of attributes*. The first dimension indicates that different numbers of tables are employed to model the same real world semantics in the databases. As the semantic information is the same, the difference must be caused by either the value-versus-table conflicts or the attribute-versus-table conflicts. (Other types of conflicts will not result in different numbers of tables when the same information is modeled.) Note that both of these two types are covered already. As for issues on the

second dimension in this type of conflict, i.e., the sets of attributes are different (but the numbers of involved tables in the two databases are the same), they are exactly what the attribute-versus-attribute type of conflict is talking about. Hence, all scenarios of table-versus-table conflicts are not new; they have been fully covered in the previous cases.

III. OPTIMIZATION AT THE ALGEBRA LEVEL

Schema conformation primarily emphasizes the task of homogenizing the *semantically equivalent*, but *structurally different* (i.e., schema conflicting) classes of data [43, 49]. For instance, height is an attribute of student relations of two databases. It may be measured in feet in one database, while in meters in another database. However, the semantics of "height" in the two databases are the same. For another example, the address information of employees in one database is described by one attribute, while in another database it is modeled as a relation separate from the employee relation. An explicit mapping for these schemas to a global schema needs to be defined and the mapping between the schemas also must be maintained in such multidatabase systems for future query translation.

It is the second (the schema conformation) issue on which this section is focused. In the past, although there have been various proposals for MDBS data modeling and even database languages [5, 7, 17, 20, 56, 58], these proposals are based on fairly high-level and intuitive understanding rather than formal and systematic mapping mechanisms. The hyperrelation concepts presented in the following provides a uniform representation of the global schema and the local schemas (both in relational form) such that the mapping between these two levels of schemas can be clearly defined. In our design, the global schema of a multidatabase system is composed of a set of *hyperrelations*. A hyperrelation is itself a relation containing a set of tuples and each tuple represents the schema of a relation of a component database. The semantics of a hyperrelation is the *least upper bound* (**lub**) of the semantics of the relations mapped onto the hyperrelations (to be detailed). We will demonstrate that by using such a structure; all types of schema-conflicting relations can be mapped to hyperrelations.

Because of this uniform representation of global and local schemas, the relational algebra can be easily extended to an algebra at the multidatabase level. This algebra, termed a hyperrelational algebra, is designed for query transformation and query optimization at the global level.

The major issues answered in this section include:

- How to determine the schema of a hyperrelation such that it properly conveys the semantics of local relations?
- How to map all the local relations onto hyperrelations? (This mapping is important for translating a global query to local subqueries.)
- How to define an algebra based on the designed hyperrelation structure so as to facilitate query transformation and optimization?

For the purposes of this work it is assumed that all other types of heterogeneities such as hardware, operating systems, networks, as well as semantic

heterogeneity (as discussed) and value inconsistencies (caused by different units, different precisions, etc.) have been resolved via a homogenizing veneer on each individual database.

A. Fundamentals and the Concepts of Lub

1. Ordering on Information Capacity

In relational databases, attributes are used to describe real world objects. We say that the *capacity* of the information conveyed by a relation is confined by the set of attributes of the relation. For semantically equivalent relations, as their attributes describe objects of the same semantics, the relations' information capacities are comparable. We can envision that a relation with attributes conveying more detailed information should have a greater information capacity than a relation whose attributes express less information. In order to compare the information capacities of two relations, we have the following definition.

DEFINITION 1 (ORDERING ON INFORMATION CAPACITY). Let $A(a_1, a_2, \ldots, a_n)$ and $B(b_1, b_2, \ldots, b_m)$ be two semantically equivalent relations, where a_i $(i = 1, \ldots, n)$ and b_j $(j = 1, \ldots, m)$ are attributes. We say that A is greater in information capacity than B, denoted by $A \sqsupseteq B$, if $\forall\, b \in \{b_1, b_2, \ldots, b_m\}$, $\exists\, \alpha \subseteq \{a_1, a_2, \ldots, a_n\}$, such that α and b are corresponding attributes.

Consider the following example. Let Stud(Name, Address, Gender, Age) and Student(Name, Street, City, State, Country, Gender, Birthday) be two (semantically equivalent) relations. The information capacity of Student is greater than that of Stud, since the information that Stud can provide is also provided by Student (i.e., each attribute Stud has a corresponding attribute in Student), but not vice versa. For semantically inequivalent relations, their information capacities are uncomparable (as they describe data of different types).

Note that the statement "$\exists\, \alpha \subseteq \{a_1, a_2, \ldots, a_n\}$" shows that α can be a subset of attributes in $\{a_1, a_2, \ldots, a_n\}$. Its corresponding attribute b, however, is only an attribute in $\{b_1, b_2, \ldots, b_m\}$. According to this definition, the information capacity of an attribute, such as {Address}, is not equivalent to that of {Street, City, State}, but {Address} $\sqsubseteq$ {Street, City, State} (assuming that all information in Address, such as country name, is also contained in the representation {Street, City, State}). This is the information capacity between *one-to-many* type corresponding attributes. As for *many-to-many* type corresponding attributes, their ordering of information capacity depends on the relationship between attributes. Suppose that $\{r_1, r_2\}$ and $\{s_1, s_2, s_3\}$ are corresponding attributes.

Case 1. The correspondence between attributes of these two sets can be decomposed to one-to-one and one-to-many types of correspondence. For example, r_1 corresponds to s_1 and s_2, and r_2 corresponds to s_3. According to our discussion on corresponding attributes of one-to-many type, we know that $\{r_1, r_2\} \sqsubseteq \{s_1, s_2, s_3\}$.

Case 2. There do not exist one-to-one/one-to-many correspondences between attributes. For instance, r_1 and r_2 represent a semantics that can be further

divided into (t,u) and (v,w), respectively, and s_1, s_2, and s_3 are divisible into (t), (u,v), and (w), respectively. In this case, the information capacities of these two sets of attributes are uncomparable. This leads to the definition of the concept of *least upper bound* to be given shortly.

The task of integrating a set of semantically equivalent relations can be viewed as defining another relation such that (1) it is semantically equivalent to the set of relations to be integrated, (2) its information capacity is greater than the set of relations, and (3) it should not contain any information not provided by the set of relations. In other words, the integrated relation schema should be the *least upper bound* of the schemas in component databases. Based on these concepts, we define the least upper bound of relations.

DEFINITION 2 (LEAST UPPER BOUND). Let $\mathcal{U}$ be the universal set of relations, and **A**, **B**, and **C** $\in \mathcal{U}$. **A** and **B** are semantically equivalent relations. We say that **C** is the least upper bound of **A** and **B**, denoted by **C**=**lub**(**A**, **B**), if

1. $\mathbf{C} \sqsupseteq \mathbf{A}$ and $\mathbf{C} \sqsupseteq \mathbf{B}$, and
2. $\forall \mathbf{W} \in \mathcal{U}$, if $\mathbf{W} \sqsupseteq \mathbf{A}$ and $\mathbf{W} \sqsupseteq \mathbf{B}$, then $\mathbf{W} \sqsupseteq \mathbf{C}$.

From the definition, we find some obvious properties as follows.

PROPERTY 1 (MINIMAL INCLUSION). *Let $\mathcal{U}$ be the universal set of relations, and* **A**, **B**, *and* **C** $\in \mathcal{U}$. **A** *and* **B** *are semantically equivalent relations. If* **C**= **lub**(**A**, **B**), *then we have*

1. *Covering:* **C** *is semantically equivalent to* **A** *and to* **B**.
2. *Fineness: The information capacity of* **C** *is greater than those of* **A** *and* **B**.
3. *Minimality:* **C** *does not arbitrarily contain the information not provided either by* **A** *or by* **B**.

PROPERTY 2 (IDEMPOTENCE). *For any relation* **A**, *the* **lub** *of* **A** *is* **A** *itself.*

PROPERTY 3 (UNIQUENESS). *For each set of semantically equivalent relations, their* **lub** *is unique.*

Proof. Let **C** and **C**′ be two **lubs** of a given set of relations {**A**, **B**}. According to the definition of **lub**, we have $\mathbf{C} \sqsupseteq \mathbf{A}$ and $\mathbf{C} \sqsupseteq \mathbf{B}$, and $\mathbf{C}' \sqsupseteq \mathbf{A}$ and $\mathbf{C}' \sqsupseteq \mathbf{B}$. As **C** is the **lub** of **A** and **B**, any relation (such as **C**′) satisfying $\mathbf{C}' \sqsupseteq \mathbf{A}$ and $\mathbf{C}' \sqsupseteq \mathbf{B}$ must also satisfy $\mathbf{C}' \sqsupseteq \mathbf{C}$. Similarly, we can derive that $\mathbf{C} \sqsupseteq \mathbf{C}'$ must be true. Therefore, **C** and **C**′ are equivalent in information capacity, meaning that the **lub** of **A** and **B** is unique. ■

The concepts of **lub** and the properties *covering*, *fineness*, and *minimality* can be used as a general guideline in the schema integration process in determining the attributes of an entity (relationship) type. In our approach, the local relations are divided into groups of relations. Within each group the relations are semantically equivalent. One hyperrelation that is in semantics the **lub** of the local relations is used to represent the schemas of the local relations. In this way, the representation of the schemas of the local relations is in a uniform

manner—the schemas of relations are still relations. Also, because a hyperrelation is the **lub** of its underlying relations, there will not be any unnecessary information added as attributes to the hyperrelation. Hence, a global query issued against the hyperrelations can always be translated into (local) queries specified on the underlying (local) relations. In the next section, we define the structure of a hyperrelation.

B. The Structure of a Hyperrelation (R^H)

In relational databases, a relation represents a class of tuples of the same type, and each tuple describes the characteristics (type) of a real world entity as shown in Fig. 6a. We extend this concept to define the structure of a hyperrelation. We consider that a hyperrelation represents a class of relations having the same semantics. Each entity in a hyperrelation is the schema of an existing relation in a component database. This concept is exemplified in Fig. 6b.

Formally, we define the structure of a hyperrelation as follows.

DEFINITION 3 (THE HYPERRELATION). *A* hyperrelation R^H is composed of a set of tuples, each having the schema of a relation in a component database as its data. The relations corresponding to the tuples in a R^H are all semantically equivalent relations, and R^H has the schema R^H (Database, Relation, $A_1, \ldots, A_n$), where R^H ($A_1 \ldots, A_n$) = **lub**($t_1, \ldots, t_m$), $t_1, \ldots, t_m$ are the tuples of R^H, and $A_1, \ldots, A_n$ are the attributes mapped from $t_1, \ldots, t_m$. Database and Relation are two system-defined attributes. The domain of Relation is the set of names of the local relations mapping onto R^H, and the domain of Database is the set of names of the databases to which these local relations belong.

According to the definition, the structure of a hyperrelation is similar to that of an ordinary relation. Each hyperrelation also has a *hyperrelation name* and consists of a number of attributes, each having an *attribute name*. However,

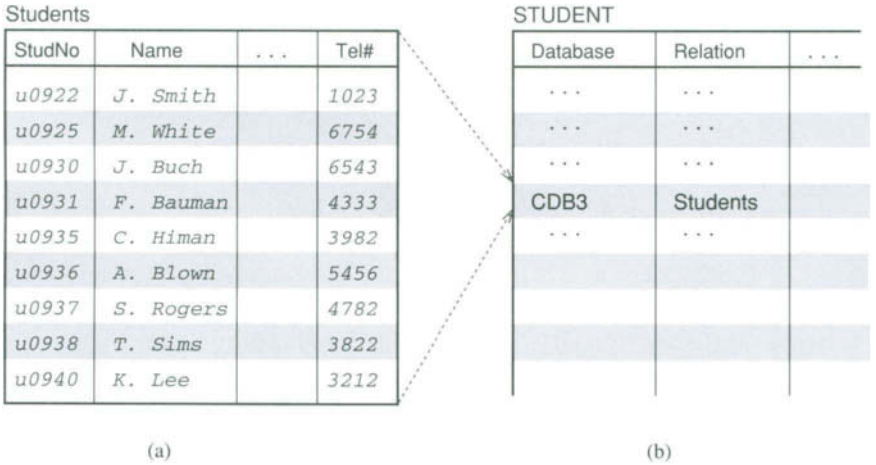


FIGURE 6 A hyperrelation structure analogous to a relation structure.

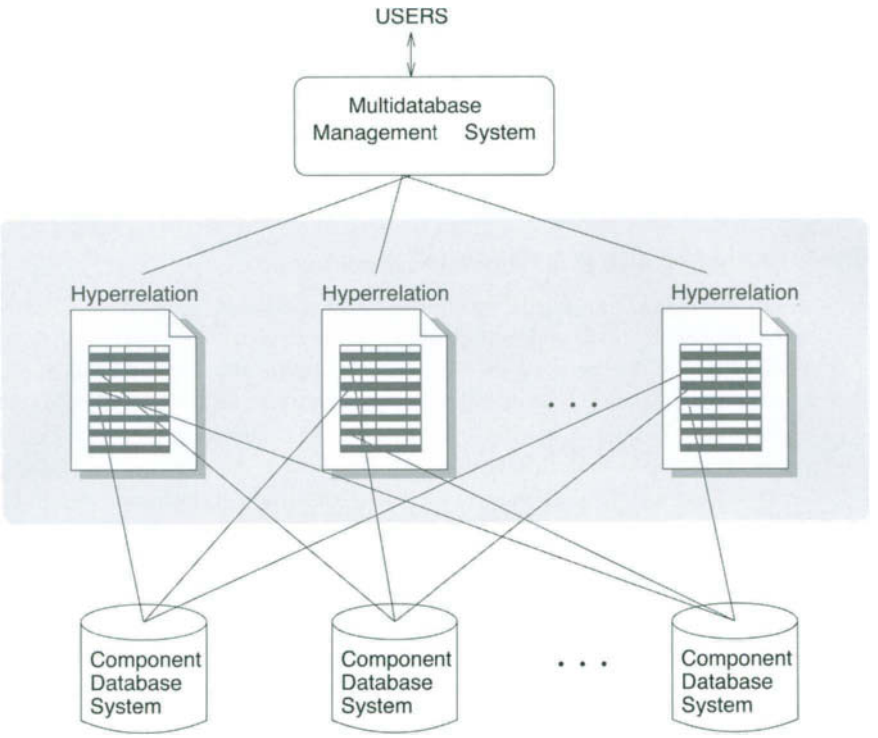


FIGURE 7 The environment of multidatabase systems.

instead of corresponding to a real-world entity, each *tuple* in a hyperrelation corresponds to a local relation that is semantically equivalent to the hyperrelation. The design of such a structure is for a representation of the local relations at the global level so as to facilitate query transformation and processing based on an algebra (to be presented). The user of a MDBS is allowed to inquire the hyperrelation schemas in order to issue a global query. The environment of such a multidatabase system is illustrated in Fig. 7, in which the hyperrelations in the middle represent the global schema of the component databases. Each global query is issued against the schemas of the hyperrelations.

Note that the hyperrelation is not just a directory or catalog as those in homogeneous distributed databases that keeps track of the attribute information of local relations. A hyperrelation needs also to reflect the conflicts between relations in a uniform manner, and more importantly, it allows a global query that accesses data in different expressions to be translated and executed locally in component databases. We discuss these issues in the following section.

C. Schema Conformation: R^H Schema and the Mapping

We discuss here how to determine the hyperrelation schema from (semantically equivalent) relations of conflicting schemas and how to map them onto a hyperrelation. A combination of more than one type of conflict can be decomposed into the above basic types and mapped to hyperrelations by following their corresponding mapping mechanisms. We start our description from simpler (more

intuitive) types of conflict to allow the reader to easier comprehend the whole mapping scheme.

1. Value-versus-Value Conflict

As data conflicts can normally be resolved by using conversion functions to convert the data, schema conformation techniques are not needed for this type of conflict.

2. Attribute-versus-Attribute Conflict

For ease of understanding, we first describe the conformation of schemas with missing attribute conflicts. Then we discuss the one-to-many attribute conflict before we discuss the most general many-to-many conflict. The one-to-one conflict case is covered in the discussion of the other two cases.

Missing Attribute Conflict

Let $R_1(a_{11}, \ldots, a_{1r_1})$, $R_2(a_{21}, \ldots, a_{2r_2}), \ldots, R_m(a_{m1}, \ldots, a_{mr_m})$ be a set of relations of different databases and among them there exists a missing attribute conflict. According to the requirements of **lub**, the set of attributes of their corresponding hyperrelation must be the union of all attributes of the relations, i.e., $\{a_{11}, \ldots, a_{1r_1}\} \cup \{a_{21}, \ldots, a_{2r_2}\}, \cup \ldots \cup \{a_{m1}, \ldots, a_{mr_m}\}$. In the union, $\{a_p\} \cup \{a_q\}$ = either $\{a_p\}$ or $\{a_q\}$, but not both, if they are the corresponding attributes. The mapping of R_i to this hyperrelation is straightforward: $R_i.a_{ij}$ is the value of the corresponding attribute in R^H (for $1 \leq i \leq m, 1 \leq j \leq r_i$) and is a NULL if R_i does not have such an attribute.

One-to-Many Conflict

Let $R(r_1, \ldots, r_m)$ $(\in DB_1)$ and $S(s_1, \ldots, s_n)$ $(\in DB_2)$ be two semantically equivalent relations, where $m < n$ and r_i is a same attribute as s_i, for $1 \leq i \leq m-1$, and r_m corresponds to $s_m, \ldots, s_n$ (i.e., a one-to-many attribute conflict). As a hyperrelation should be in the finest structure to convey the most detailed information (according to the definition of the **lub**), the structure $(s_1, \ldots, s_m, \ldots, s_n)$ should be chosen as the schema of R^H (in addition to the attributes Database and Relation). The attribute names $s_1, \ldots, s_{m-1}$ of R^H can be $r_1, \ldots, r_{m-1}$ too because there is no mismatch between them. To map the relation onto R^H, R is mapped to a tuple as

$$\langle DB_1, R, r_1, \ldots, r_{m-1}, \overbrace{r_m, \ldots, r_m}^{n-m+1} \rangle.$$

The values of $R.s_i$ in R^H is r_i, for $1 \leq i \leq m-1$, and the values of $R.s_i$, for $m \leq i \leq n$, are defined as r_m. S is mapped to

$$\langle DB_2, S, s_1, \ldots, s_m, \ldots, s_n \rangle$$

in R^H. Both of the mappings strictly follow the attribute correspondence rule mentioned in Section II.

For an example, the student relations of CDB1, CDB2, and CDB3 are mapped to STUDENT as shown in Fig. 8. The Stud of CDB1 in STUDENT has Name as the values for FN and LN because of the one-to-many conflict with the corresponding attributes in CDB2. The address information of these

STUDENT

Database	Relation	StudNo	FN	LN	No.	Street	City	State	Tel#	Gender	Grade
CDB1	Stud	$S\#^K$	Name	Name	Addr	Addr	Addr	Addr	Tel#	Gender	Class
CDB2	CS-Stud	$S\#^K$	FN	LN	No	Str	Str	State	Tel#	NULL	Grade
CDB2	Math-Stud	$S\#^K$	FN	LN	No	Str	Str	State	Tel#	NULL	Grade
CDB2	EE-Stud	$S\#^K$	FN	LN	No	Str	Str	State	Tel#	NULL	Grade
CDB3	Studs	$S\#^K$	Name	Name	Str	Str	City	State	NULL	Gender	Class

FIGURE 8 The schema of the hyperrelation STUDENT.

relations are mapped to STUDENT based on the same principle. This mapping is also important for query transformation based on algebra. We will get to that in a later section.

Many-to-Many Conflict

As for the many-to-many conflict, the basic idea is the same. We illustrate the mapping by using a conflict example involving two attributes in one relation and three attributes in another relation. A general case will simply be an extension of the case presented here. Let r_1 and r_2 be attributes of R and s_1, s_2, s_3 be attributes of S, and $\{r_1, r_2\}$ are the corresponding attributes of $\{s_1, s_2, s_3\}$. Assume that r_1 and r_2 represent a semantics that can be further divided into (a, b) and (c, d), respectively; i.e., a, b, c, and d are subsemantics of r_1 and r_2. In S, the same semantics (a), (b, c), and (d) are modeled in s_1, s_2, and s_3, respectively. Then the hyperrelation should choose a, b, c, and d as its attributes since this representation is semantically richer than the other two representations. In other words, R^H has the schema $R^H(\ldots, a, b, c, d, \ldots)$ (irrelevant attributes are denoted by dots). Based on the mapping principle presented above, r_1 and r_2 are mapped to $\langle \ldots, r_1, r_1, r_2, r_2, \ldots \rangle$ in R^H and s_1, s_2, and s_3 are mapped to $\langle \ldots, s_1, s_2, s_2, s_3, \ldots \rangle$ in R^H.

3. Attribute-versus-Table Conflict

Consider the Faculty relation of CDB1 and the Faculty and Spouse relations of CDB2. As the Spouse relation is semantically different from the Faculty relation, there should be two hyperrelations FACULTY and SPOUSE in the global database.

Formally, let us assume that $R(a_1, \ldots, a_p, a_{p+1})$ is a local relation of DB_1 that semantically corresponds to $R_1(a_1, \ldots, a_p, f)$ and $R_2(f, b_1, \ldots, b_q)$ of DB_2, where $R.a_i$ corresponds to (i.e., the corresponding attribute of) $R_1.a_i$, for $1 \leq i \leq p$, $R.a_{p+1}$ corresponds to $R_2.b_1$, and f is the foreign key (key) attribute of R_1 (R_2). R and R_1 are semantically equivalent relations. In this circumstance, the hyperrelations will have the schemas R_1^H (*Database, Relation,* $A_1, \ldots, A_p, F$) and R_2^H (*Database, Relation,* $F, B_1, \ldots, B_q$), where R_1^H is obtained from the schemas of R and R_1, and R_2^H from that of R_2. F is a key or a foreign key of these hyperrelations to express the referencing information. We define that R_1 and R_2 are mapped to the tuples

$$\langle DB_2, R_1, a_1, \ldots, a_p, f \rangle$$

and

$$\langle DB_2, R_2, f, b_1, \ldots, b_q \rangle$$

in hyperrelations R_1^H and R_2^H, respectively. The relation R is mapped to two tuples

$$\langle DB_1, R, a_1, \ldots, a_p, \rangle$$

and

$$\langle DB_1, R, \pounds, \overbrace{a_{p+1}, \text{Null}, \ldots, \text{Null}}^{q} \rangle$$

in R_1^H and R_2^H, respectively, because a_{p+1} corresponds to b_1 and R does not have the other (i.e., $B_2, \ldots, B_q$) attributes. $\pounds$ is a special character to indicate that relation R does not have such an attribute and can be grouped back to one relation schema through this link (i.e., $\pounds$). In this manner, a global query issued based on the hyperrelation schema can be correctly translated to local queries expressed in local schemas.

4. Value-versus-Attribute Conflict

Let $R_1(a_1, a_2, \ldots, a_p)$ and $R_2(a_{11}, a_{12}, \ldots, a_{1k}, a_2, \ldots, a_p)$ be the relations of databases DB_1 and DB_2, respectively, where a value-versus-attribute conflict exists between the values of $R_1.a_1$ and the attributes $a_{11}, a_{12}, \ldots, a_{1k}$ of R_2, and $R_1.a_i$ corresponds to $R_2.a_i$ (no schema conflict between them) for $2 \leq i \leq p$. As $\{a_{11}, a_{12}, \ldots, a_{1k}\}$ has a finer structure than a_1, the schema of their corresponding hyperrelation should be R^H(*Database, Relation,* $A_{11}, A_{12}, \ldots, A_{1k}, A_2, \ldots, A_p$), where a_{1j} corresponds to A_{1j} (for $1 \leq j \leq k$) and a_i to A_i (for $2 \leq i \leq p$). Mapped to R^H, the schema of R_1 becomes the tuple

$$\langle DB_1, R, \overbrace{a_1, a_1, \ldots, a_1}^{k}, a_2, \ldots, a_p \rangle.$$

Since R_2 has no conflict with R^H, the mapping result is simply R_2's schema.

For example, the attribute Type of Course in CDB1 is mapped onto the hyperrelation COURSE as shown in Fig. 9, in which the attributes Fundamental and Advanced of the CDB1 tuple have the value "Type."

5. Value-versus-Table Conflict

Let $R(r, a_1, \ldots, a_p)$ be a relation in the database DB_1 that has a value-versus-table conflict with the set of relations $R_1(a_1, \ldots, a_p), \ldots, R_m(a_1, \ldots, a_p)$ in database DB_2. Formally, we say that the domain of the attribute r of R is an index set $\{r_1, \ldots, r_m\}$ on R_i $(i = 1, \ldots, m)$, where r_i refers to the domain

COURSE

Database	Relation	CNo	CName	Credits	Hours	Fundamental	Advanced	Prereq
CDB1	*Course*	*C#*^{K}	*Name*	*Credit*	*NULL*	*Type*	*Type*	*NULL*

FIGURE 9 The schema of the hyperrelation COURSE.

STUDENT

Database	Relation	SSN	StudNo	FN	LN	No.	Street	City	State	Tel#	Gender	Grade
CDB4	ST1	Null	S# ^{k}	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST1.Grade*
CDB4	ST2	Null	S# ^{k}	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST2.Grade*
CDB4	ST3	Null	S# ^{k}	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST3.Grade*
CDB4	ST4	Null	S# ^{k}	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST4.Grade*

CDB4.ST1.Grade -> {freshman}, CDB4.ST2.Grade -> {sophomore}

CDB4.ST3.Grade -> {junior}, CDB4.ST4.Grade -> {senior}

FIGURE 10 The hyperrelation STUDENT.

of relation R_i.[4] It is the global database designer's responsibility to define the domains of relations (analogous to defining the domain of an attribute) in the database design process. As a hyperrelation, not counting the attributes Database and Relation, is the **lub** of the underlying relations, R^H have the schema R^H (*Database, Relation, r*, $A_1, \ldots, A_p$), where $A_1, \ldots, A_p$ correspond to $a_1, \ldots, a_p$, respectively, and $R^H.r$ to $R.r$. The relation R of DB_1 is mapped to a tuple in R^H as

$$\langle DB_1, R, r, a_1, \ldots, a_p \rangle.$$

As for R_i of DB_2, it is mapped to a tuple as

$$\langle DB_2, R_i, r_i^\star, a_1, \ldots, a_p \rangle$$

in which the value of the attribute r is not null but $r_i^\star$ (where r_i is an index to the domain of R_i). In the example discussed above, the relations ST1, ST2, ST3, and ST4 of CDB4 given in Fig. 5 are mapped to tuples of a hyperrelation STUDENT shown in Fig. 10. The values of the added attribute Grade are set to {CDB4.ST1.Grade*, CDB4.ST2.Grade*, CDB4.ST3.Grade*, CDB4.ST4.Grade*} for the relations. The $\star$-tagged values indicate (to the global user) that the local relation does not have the corresponding attributes, but each index ($r_i^\star$) refers to the domain of a local relation, as indicated in Fig. 10. The k-tagged values in the hyperrelation stand for the key attributes of the local relations.

Note that in a previous attribute-versus-table conflict the relations $R_1(a_1, \ldots, a_p, f)$ and $R_2(f, b_1, \ldots, b_q)$ of DB_2 are mapped to different hyperrelations, while the $R_1(a_1, \ldots, a_p), \ldots, R_m(a_1, \ldots, a_p)$ of this value-versus-table conflict are mapped onto the same hyperrelation. The reason is that the relations $R_1(a_1, \ldots, a_p), \ldots, R_m(a_1, \ldots, a_p)$ in the latter type of conflict are semantically equivalent, whereas the semantics of R_1 and R_2 in the former case are distinct. The solutions of these two cases are therefore different.

6. Table-versus-Table Conflict

Thus far, we have classified the schema conflicts and discussed the mappings of each type of conflicts to hyperrelations. Based on the principles discussed

[4]For simplicity, we have assumed here that there exists a one-to-one and onto mapping from the values in the domain of the attribute r, i.e., $\{r_1, \ldots, r_m\}$, to the domains of the set of relations $\{R_1, \ldots, R_m\}$ in DB_2. We have found that this is indeed the case most of the time.

STUDENT

Database	Relation	SSN	StudNo	FN	LN	No.	Street	City	State	Tel#	Gender	Grade
CDB1	Stud	Null	$S\#^K$	Name	Name	Addr	Addr	Addr	Addr	Tel#	Gender	Class
CDB2	CS_Stud	Null	SN^K	FN	LN	No	Str	Str	State	Tel#	NULL	Grade
CDB2	Math_Stud	Null	SN^K	FN	LN	No	Str	Str	State	Tel#	NULL	Grade
CDB2	EE_Stud	Null	SN^K	FN	LN	No	Str	Str	State	Tel#	NULL	Grade
CDB3	Studs	SSN^K	SNo	Name	Name	Str	Str	City	State	NULL	Gender	Class
CDB4	ST1	Null	$S\#^K$	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST1.Grade*
CDB4	ST2	Null	$S\#^K$	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST2.Grade*
CDB4	ST3	Null	$S\#^K$	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST3.Grade*
CDB4	ST4	Null	$S\#^K$	Name	Name	Addr	Addr	Addr	Addr	Tel#	Sex	CDB4.ST4.Grade*

CDB4.ST1.Grade -> {freshman}, CDB4.ST2.Grade -> {sophomore}
CDB4.ST3.Grade -> {junior}, CDB4.ST4.Grade -> {senior}

FIGURE 11 The hyperrelation STUDENT.

above, multidatabase relations can be mapped onto their corresponding hyperrelations. Some of the hyperrelations mapped from the relations given in Fig. 5 are shown in Figs. 11, 12, and 13.

D. The Hyperrelational Algebra

In this section, we introduce how the relational operations are extended to hyperrelational operations and how these operations are performed on a hyperrelation. Analogous to the relational algebra in the relational model, the hyperrelational algebra provides a mathematical foundation for expressing global queries and the transformation and optimization of the queries.

1. The Hyperrelational Operations

The hyperrelational operations are a direct extension of the relational operations. They are performed on hyperrelations. These operations include H-SELECTION (σ^H), H-PROJECTION (π^H), H-JOIN ($\bowtie^H$), H-INTERSECTION ($\cap^H$), H-UNION ($\cup^H$), H-DIFFERENCE ($-^H$), and

COURSE

Database	Relation	CNo	CName	Credits	Hours	Fundamental	Advanced	Prereq
CDB1	Course	$C\#^K$	Name	Credit	NULL	Type	Type	NULL
CDB2	Fndmntl_Crs	CN^K	Name	Credits	NULL	α	NULL	NULL
CDB2	Advncd_Crs	CN^K	Name	Credits	NULL	NULL	γ	Prereq-C#
CDB3	Lecture	CNo^K	Name	Credit	NULL	β	NULL	NULL
CDB3	Tutorial	CNo^K	Name	Credit	NULL	δ	NULL	NULL
CDB3	Seminar	CNo^K	Name	NULL	NULL	NULL	λ	NULL
CDB3	Indvl_Stdy	CNo^K	Name	NULL	NULL	NULL	ω	NULL
CDB4	Courses	$C\#^K$	Name	Credit	Hours	Fndmntl	Advncd	NULL

α = CDB2.Fndmntl_Crs.Fundamental* β = CDB3.Lecture.Fundamental*
δ = CDB3.Tutorial.Fundamental* γ = CDB2.Advncd_Crs.Advanced*
λ = CDB3.Seminar.Advanced* ω = CDB3.Indvl_stdy.Advanced*

FIGURE 12 The hyperrelation COURSE.

TAKE

Database	Relation	SSN	SNo	CNo	Score
CDB1	Take	Null	$S\#^K$	$C\#^K$	Grade
CDB2	Participate	Null	SN^K	CN^K	Score
CDB3	Takes	SSN^K	Null	CN^K	Grade
CDB4	ENROLL	Null	$S\#^K$	$C\#^K$	Score

FIGURE 13 The hyperrelation TAKE.

H-PRODUCT ($\times^H$). They are defined based on the relational operations, the relational algebra, and the set theory. The relational set operations, including union, intersection, and difference operations, are in their broader sense in the transformation; that is, the set operations are *outer-union*, *outer-intersection*, and "*generalized-difference*" operations (in which the generalized-difference operation is a new operation to be introduced). Each global query is expressible in a hyperrelational algebraic expression by using these operations. During execution, a hyperrelational operation is used to select qualifying tuples, which represent some local relations, and is then converted to corresponding relational operations. The result of the operation is obtained by performing the corresponding relational operations on the selected local relations. The result is still a relation.

Before giving definitions of the operations, we first introduce an operation that is used in all the hyperrelational operation acting as a conflict conformation operator. This operator is called the $\star$-*expansion* (read *star expansion*) operator. It is applied to the tuples of a hyperrelation when a hyperrelational operation is being performed.

DEFINITION 4 ($\star$-EXPANSION). Let $r(v_1, \ldots, v_m)$ be the schema of r and $\langle db, r, c_1^\star, \ldots, c_n^\star, v_1, \ldots, v_m\rangle$ be the tuple of r in R^H, where c_i are $\star$-tagged values. The $\star$-expansion of $c_1^\star, \ldots, c_n^\star$ on r, or simply the $\star$-expansion of r, is defined as

$$\begin{aligned}\star_{\{c_1,\ldots,c_n\}} db.r &= \star(\langle db, r, c_1^\star, \ldots c_n^\star, v_1, \ldots, v_m\rangle)\\ &= \{\langle c_1\rangle\} \times \ldots \times \{\langle c_n\rangle\} \times r(v_1, \ldots, v_m).\end{aligned}$$

This definition implies that the result of the $\star$-expansion operation is a relation that has the attributes $c_1, \ldots, c_n$ (i.e., $\star$-tagged values) in addition to the attributes $v_1, \ldots, v_m$ contained in $db.r$ (i.e., the relation r of database db). The values of these new attributes are set to be the same as their attribute names. In our multidatabase example, for instance, the relation ST1 in CDB4 stores the data about the freshmen in College D. The $\star$-expansion of ST1 is $\{\langle \text{freshman}\rangle\} \times$ ST1; that is, the original information about the grade of a student is added as an attribute to the relation and all its values of the tuples in the student relation are "freshman." The $\star$-expansion is performed on all $c_i^\star$ (if more than one), but not any subset of them. Since all $c_i^\star$ will be applied, the operation $\star_{\{c_1,\ldots,c_n\}} db.r$ can also be expressed as $\star db.r$ for an abbreviation. The purpose of this operation is to mediate the conflict between data and metadata. It is automatically applied whenever a hyperrelation operation is performed on the hyperrelation (to be shown shortly).

PROPERTY 4 (IDEMPOTENCE). $\star(\star \cdots \star(\langle c_i^\star, \ldots, c_n^\star, v_1, \ldots, v_m\rangle)\ldots) = \star(\langle c_1^\star, \ldots, c_n^\star, v_1, \ldots, v_m\rangle)$.

The proof of this property is quite straightforward and is therefore omitted. This property guarantees that multiple applications of the $\star$ operation on a hyperrelation do not incur an erroneous result.

The H-SELECTION Operation

The H-SELECTION operation is to perform selections on a hyperrelation and then to perform transformed selection operations on the selected local relations. Formally, the operation is defined as follows.

DEFINITION 5 (H-SELECTION). Given a hyperrelation R^H having the set of tuples $\{t_1, \ldots, t_n\}$, the H-SELECTION (σ^H) on R^H under the selection condition SC is defined as

$$\sigma_{SC}^H(R^H) = \sigma_{SC_1}(\star t_1) \cup \cdots \cup \sigma_{SC_n}(\star t_n),$$

where σ and $\cup$ are the relational operator SELECTION and the set operator UNION,[5] respectively. Assume that the SC is a minterm[6] and expressed as $SC = (a_{i_1}\theta_1\omega_1) \wedge (a_{i_2}\theta_2\omega_2) \ldots \wedge (a_{i_j}\theta_j\omega_j)$, where a_{ik} is an attribute of R^H, $\theta_k \in \{>, \geq, =, \neq, <, \leq\}$, and ω_k is a constant value, for all $k = 1, \ldots, j$. If the values of the attributes $a_{i_1}, \ldots, a_{i_j}$ of the tuple $\star t_p (p = 1, \ldots, n)$ are $v_{i_1}, \ldots, v_{i_j}$, respectively, and none of them is a NULL, then $SC_p = (v_{i_1}\theta_1\omega_1) \wedge (v_{i_2}\theta_2\omega_2) \ldots \wedge (v_{i_j}\theta_j\omega_j)$, If any of the attribute values is a NULL, then $\sigma_{SC_p}(\star t_p) = \phi$.

The H-SELECTION operation is composed of a number of relational SELECTIONs and UNIONs. A H-SELECTION on a hyperrelation is carried out by applying SELECTIONs on the tuples of the hyperrelation, i.e., the underlying relations from component databases, and UNIONs of the results from each component databases. For instance, given a hyperrelation STUDENT as shown in Fig. 11, the hyperrelational query

$$\sigma^H{}_{\text{Grade=freshman}}(\text{STUDENT})$$

is equal to the following query expressed in relational algebra:

$$\sigma_{\text{Class=freshman}}(\text{CDB1.Stud}) \cup \sigma_{\text{Grade=freshman}}(\text{CDB2.CS\_Stud}) \cup$$
$$\sigma_{\text{Grade=freshman}}(\text{CDB2.MATH\_Stud}) \cup$$
$$\sigma_{\text{Grade=freshman}}(\text{CDB2.EE\_Stud}) \cup \sigma_{\text{Class=freshman}}(\text{CDB3.Studs}) \cup$$
$$\sigma_{\text{freshman=freshman}}(\star_{\{\text{freshman}\}}\text{CDB4.ST1}) \cup$$
$$\sigma_{\text{sophomore=freshman}}(\star_{\{\text{sophomore}\}}\text{CDB4.ST2}) \cup$$
$$\sigma_{\text{junior=freshman}}(\star_{\{\text{junior}\}}\text{CDB4.ST3}) \cup$$
$$\sigma_{\text{senior=freshman}}(\star_{\{\text{senior}\}}\text{CDB4.ST4})$$

[5]It is actually an outer-union operator. We will have a detailed explanation in a moment.

[6]Two OR-ed minterms such as $(A \wedge B) \wedge (C \wedge D)$ can be viewed as two queries and treated separately.

For the selection conditions of the CDB4 relations, as the condition freshman=freshman is always true, all tuples of the local relation ST1 are selected. On the other hand, no tuples will be selected from ST2, ST3, and ST4 because their selection conditions are always false.

Also, if a tuple r of R^H has a NULL value for an attribute that is involved in the SC, we define that the $\sigma_{SC}(r) = \phi$: The reason for not processing such a relation is to avoid uncertain (maybe) results. Let R^H (DB, Rel, Studno, Name, Address, Gender) be the schema of a hyperrelation and ⟨db_1, r_1, S#, SName, Addr, NULL⟩ and ⟨db_2, r_2, S#, SName, Addr, Sex⟩ be two tuples of the hyperrelation. If a selection query is to find the information of students whose gender is female, then none of the data of relation r_1 should be considered, as the gender of those students is unknown. Only the relation corresponding to the second tuple ($db_2.r_2$) should be processed. However, if maybe values are welcome in a particular environment, the hyperrelational algebra will still work by changing the definition of H-selection to allow selection on uncertain information.

An interesting aspect of this algebra is that as some relations corresponding to a R^H may not have all the attributes defined in R^H, the issue of *union compatibility* [28] is involved in the UNION operation in σ^H (refer to the definition). There are two choices in dealing with this problem:

1. Strictly confine that all relations to be unioned must have the same set of attributes. If any relation does not meet this condition, it is removed from the σ^H expression.
2. Take the *outer-union* approach, which releases the union compatibility constraint by allowing relations of different sets of attributes to be unioned and the resulting relation contains all attributes from the unioned relations.

We take the second approach for the hyperrelational algebra, because it allows the user to obtain the largest set of information as the result. If the first choice is taken, then following the last example a relation in R^H (DB, Rel, Studno, Name, Address, Gender) that has the schema ⟨db_3, r_3, S#, SName, NULL, Sex⟩ will not be processed (because the relations are union incompatible), even though the sexuality of students is known. To the user, the resultant female students not including those in r_3 are an incomplete set of students. The answer of a multidatabase query is by default to inquire all existing qualifying tuples. Hence, the second choice above is considered. A formal definition of the outer-union operation can be found in [28]. The column of Address for r_3 tuples can simply be filled with NULLs to indicate nonavailability of the information.

There may be a concern that the result of an outer-union of two partially compatible relations having schemas such as Student(SSN, Name, Address, Class) and Teacher(SSN, Name, Address, Rank will have the schema R(SSN, Name, Address, Rank, Class) (i.e., all attributes of Student and Teacher are included). Hence, an outer-union of multiple relations without a guidance will likely make the semantics of the result relation too complex to be clear and precise. In our definition of the hyperrelational algebra, however, this situation will not occur because the union is confined to those relations (tuples) within the same R^H. These relations are semantically equivalent; the relations in the

above example (Student and Teacher) will never be outer-unioned in our case. Also, as the semantics of the relations mapped to a R^H are equivalent to that of the R^H, the result of the outer-union of any of the relations still has the same semantics as R^H. In the following operations, all the "$\cup$" symbols (not including $\cup^H$) denote an outer-union, unless specified otherwise.

The H-PROJECTION Operation

The H-PROJECTION operation is to perform projections on the tuples of a hyperrelation. The H-PROJECTION operation is defined as follows.

DEFINITION 6 (H-PROJECTION). Given a hyperrelation R^H having the set of tuples $\{t_1, \ldots, t_n\}$, the H-PROJECTION (π^H) of R^H on a set of attributes {PA} is defined as

$$\pi^H_{\{PA\}}(R^H) = \pi_{\{PA_1\}}(\star t_1) \cup \cdots \cup \pi_{\{PA_n\}}(\star t_n),$$

where π is the relational operator PROJECTION and $\{PA_i\}$ is the set of attributes of t_i corresponding to the attributes $\{PA\}$.

Let us take the hyperrelation STUDENT shown in Fig. 11 as an example. The H-PROJECTION operation on STUDENT

$$\pi^H_{\{\text{FN,LN,Tel\#}\}}(\text{STUDENT})$$

is equal to the operations

$$\pi_{\{\text{Name,Name,Tel\#}\}}(\text{CDB1.Stud}) \cup \pi_{\{\text{FN,LN,Tel\#}\}}(\text{CDB2.CS\_Stud}) \cup$$
$$\pi_{\{\text{FN,LN,Tel\#}\}}(\text{CDB2.MATH\_Stud}) \cup \pi_{\{\text{FN,LN,Tel\#}\}}(\text{CDB2.EE\_Stud}) \cup$$
$$\pi_{\{\text{Name,Name,NULL}\}}(\text{CDB3.Studs}) \cup \pi_{\{\text{Name,Name,Tel\#}\}}$$
$$(\star_{\{\text{freshman}\}}\text{CDB4.ST1}) \cup \pi_{\{\text{Name,Name,Tel\#}\}}(\star_{\{\text{sophomore}\}}\text{CDB4.ST2}) \cup$$
$$\pi_{\{\text{Name,Name,Tel\#}\}}(\star_{\{\text{junior}\}}\text{CDB4.ST3}) \cup$$
$$\pi_{\{\text{Name,Name,Tel\#}\}}(\star_{\{\text{senior}\}}\text{CDB4.ST4}).$$

The projection operation $\pi_{\{\text{Name,Name,Tel\#}\}}(\text{Stud})$ is equal to $\pi_{\{\text{Name, Tel\#}\}}(\text{Stud})$, as according to the set theory $\{A, A, B\}$ is equal to $\{A, B\}$. Note that although the relation CDB3.Studs does not have the Tel# attribute, its data are still included in the result. The consideration here about a NULL value is somewhat different from that in the H-SELECTION operation. In the H-SELECTION operation, if a local relation does not have the attribute specified in the condition of a global selection query, the entire local relation is excluded from the result. The intention is to avoid creating uncertain tuples in the final result. In the H-PROJECTION, however, tuples having a missing attribute are still included in the result. This is because projection on a missing attribute does not cause an uncertainty problem. All the tuples in such a relation are still true tuples (instead of maybe tuples as in the H-SELECTION case). It is similar to the case in relational databases in which an employee tuple in an

employee relation, for instance, with unknown Tel# is still considered a legal and valid tuple in the relation.

The H-JOIN Operation

The H-JOIN operation is used to select related tuples from two hyperrelations and then performs joins on each pair of the selected tuples. We defined the H-JOIN operation as follows.

DEFINITION 7 (H-JOIN). Given two hyperrelation R^H and S^H having the set of tuples $\{t_1, \ldots, t_n\}$ and $\{u_1, \ldots, u_m\}$, respectively. The H-JOIN ($\bowtie^H$) on R^H and S^H under the join condition JC is defined as

$$(R^H) \bowtie^H_{JC} (S^H) = ((\star t_1) \bowtie_{JC_{11}} (\star u_1)) \cup \cdots \cup ((\star t_1) \bowtie_{JC_{1m}} (\star u_m)) \cup \cdots$$
$$\cup((\star t_n)\bowtie_{JC_{nm}} (\star u_m)),$$

where $\bowtie$ is the relational JOIN operation. The join condition JC is a Boolean expression and is specified on attributes from the two hyperrelations R^H and S^H. JC_{ij} is the transformation of JC by substituting the attributes of the hyperrelations with the values of the attributes. $(\star t_i) \bowtie_{JC_{ij}} (\star u_j)(i = 1, \ldots, n$ and $j = 1, \ldots, m)$ is an empty set if any of the values of the attributes of t_i and u_j involved in JC_{ij} is a NULL.

For example, the following H-JOIN

$$\text{STUDENT} \bowtie^M_{\text{StuNo=SNo}} \text{TAKE}$$

is equal to

$$\left(\text{Stud} \bowtie_{\text{S\#=S\#}} \text{Take}\right) \cup \left(\text{Stud} \bowtie_{\text{S\#=SN}} \text{Participate}\right) \cup \left(\text{Stud} \bowtie_{\text{S\#=Null}} \text{Takes}\right) \cup$$
$$\left(\text{Stud} \bowtie_{\text{S\#=S\#}} \text{Enroll}\right) \cup \left(\text{CS\_Stud} \bowtie_{\text{SN=S\#}} \text{Take}\right) \cup \left(\text{CS\_Stud} \bowtie_{\text{SN=SN}} \text{Participate}\right) \cup$$
$$\left(\text{CS\_Stud} \bowtie_{\text{SN=Null}} \text{Takes}\right) \cup \left(\text{CS\_Stud} \bowtie_{\text{SN=S\#}} \text{Enroll}\right) \cup \cdots \cup \left(\text{Studs} \bowtie_{\text{SNo=S\#}} \text{Take}\right) \cup$$
$$\left(\text{Studs} \bowtie_{\text{SNo=SN}} \text{Participate}\right) \cup \left(\text{Studs} \bowtie_{\text{SNo=Null}} \text{Takes}\right) \cup \left(\text{Studs} \bowtie_{\text{SNo=S\#}} \text{Enroll}\right) \cup$$
$$\left(\text{ST1} \bowtie_{\text{S\#=S\#}} \text{Take}\right) \cup \left(\text{ST1} \bowtie_{\text{S\#=SN}} \text{Participate}\right) \cup \left(\text{ST1} \bowtie_{\text{S\#=Null}} \text{Takes}\right) \cup$$
$$\left(\text{ST1} \bowtie_{\text{S\#=S\#}} \text{Enroll}\right) \cup \cdots \cup \left(\text{ST4} \bowtie_{\text{S\#=S\#}} \text{Take}\right) \cup \cdots \cup \left(\text{ST4} \bowtie_{\text{S\#=S\#}} \text{Enroll}\right).$$

The database names in this case need not be given since all local relations happen to have distinct names. Note that the H-JOIN performs a relational join on *every pair* of the relations from the two operand hyperrelations. This definition gives the general formula for a hyperrelational join operation. Note that two students in different universities (databases) having the same student number do not necessarily mean the same person (and most likely they are not). To identify whether two records in different databases refer to the same real-world entity is termed the *entity identification* problem in the literature [11, 16, 20, 81]. As this problem is beyond the focus of this paper, we simply assume that the entity identification process is done before or after the algebraic

transformation. In an environment in which entity identification is not supported, the joins between relations of different databases become unnecessary and a hyperrelational join such as

$$\text{STUDENT} \bowtie^{H}_{\text{StuNo=SNo}} \text{TAKE}$$

is reduced to

$$(\text{Stud} \bowtie_{\text{S\#=S\#}} \text{Take}) \cup (\text{CS\_Stud} \bowtie_{\text{SN=SN}} \text{Participate}) \cup$$
$$(\text{Math\_Stud} \bowtie_{\text{SN=SN}} \text{Participate}) \cup (\text{EE\_Stud} \bowtie_{\text{SN=SN}} \text{Participate}) \cup$$
$$(\text{Studs} \bowtie_{\text{SNo=Null}} \text{Takes}) \cup (\text{ST1} \bowtie_{\text{S\#=S\#}} \text{Enroll}) \cup \cdots \cup (\text{ST4} \bowtie_{\text{S\#=S\#}} \text{Enroll}).$$

The next group of hyperrelational data manipulation operations are the mathematical set operations. These operations include H-UNION, H-DIFFERENCE, H-INTERSECTION, and H-PRODUCT. Note that although they are all based on the relational outer-union operations, they require that *the operand hyperrelations be union-compatible,* analogous to the requirements of their counterpart in relational algebra. Details are given as follows.

The H-UNION Operation

The H-UNION operation ($\cup^H$) is used to merge tuples by first applying unions to two sets of relations and then applying a union again to the unioned relations.

DEFINITION 8 (H-UNION). Given two union-compatible hyperrelations R^H and S^H having the sets of tuples $\{t_1, \ldots, t_n\}$ and $\{u_1, \ldots, u_m\}$, respectively, the H-UNION ($\cup^H$) on R^H and S^H is defined as

$$(R^H) \cup^H (S^H) = ((\star t_1) \cup \cdots \cup (\star t_n)) \cup ((\star u_1) \cup \cdots \cup (\star u_m)).$$

The condition union-compatible in this defintion requires that the schemas of R^H and S^H be the same. This condition is enforced in hyperrelational algebra to avoid producing semantically uninterpretable hyperrelations during query processing. The results are from multiple databases the tuples owned by the relations that are mapped to R^H and S^H.

The H-DIFFERENCE Operation

The H-DIFFERENCE operation is used to select those tuples that are in one hyperrelation but not in another.

DEFINITION 9 (H-DIFFERENCE). Given two union-compatible hyperrelations R^M and S^M having the sets of tuples $\{t_1, \ldots, t_n\}$ and $\{u_1, \ldots, u_m\}$, respectively, the H-DIFFERENCE ($-^H$) on R^H and S^H is defined as

$$(R^H) -^H (S^H) = ((\star t_1) \cup \cdots \cup (\star t_n)) - ((\star u_1) \cup \cdots \cup (\star u_m)),$$
$$= ((\star t_1 - \star u_1) \cup \cdots \cup (\star t_n - \star u_1)) \cap \cdots \cap ((\star t_1 - \star u_m) \cup \cdots \cup (\star t_n - \star u_m)),$$

where $-$ is the relational DIFFERENCE operation.

Similar to the H-UNION operation, the H-DIFFERENCE operation can be performed only on union-compatible hyperrelations. Also similar to $\cup$, $\cap$

is an OUTER-INTERSECTION operation [28]. That is, $R \cap S$ (R and S are relations) will have all the attributes from R and S. Tuples of the relations having the same key values will be retained. For the same purpose, $R - S$ does not require that R and S be union compatible. We generalize the concept of the relational difference operation and define a new operation named G-difference (Generalized-difference, $-^G$) operation as follows.

DEFINITION 10 (G-DIFFERENCE). Given two semantically equivalent relations R and S having the schemas $R(r_1, r_2, \ldots, r_n)$ and $S(s_1, s_2, \ldots, s_m)$, respectively, the *G-DIFFERENCE* on R and S is defined as

$$R -^G S = \{v \mid v \in R \text{ and } v \notin S, \text{and } \forall v \text{ has the same schema as } R(r_1, r_2, \ldots, r_n)\}.$$

As we can see, this operation relaxes the union compatibility restriction of the relational difference operations and requires only that two relations be semantically equivalent. Hence, their schemas need not be exactly the same. The result of $R - S$ will have the same schema as relation R. Beyond this distinction, the G-difference is the same as the relational difference operation. Note that the operation "$-$" in Definition 3.9 as well as in the rest part of the paper is actually the difference operation in the generalized sense. By generalizing the $\cap$, $\cup$, and $-$ operations to OUTER-UNION, OUTER-INTERSECTION, and G-DIFFERENCE operations, the hyperrelational algebra can provide a multidatabase system with the greatest flexibility on processing local relations to obtain the maximal set of data without sacrificing the semantic integrity of the data.

The H-INTERSECTION Operation

DEFINITION 11 (H-INTERSECTION). Given two union-compatible hyperrelations R^H and S^H having the sets of tuples $\{t_1, \ldots, t_n\}$ and $\{u_1, \ldots, u_m\}$, the H-INTERSECTION ($\cap^H$) on R^H and S^H is defined as

$$(R^H) \cap^H (S^H) = ((\star t_1) \cup \cdots \cup (\star t_n)) \cap ((\star u_1) \cup \cdots \cup (\star u_m))$$
$$= (\star t_1 \cap \star u_1) \cup \cdots \cup (\star t_1 \cap \star u_m) \cup (\star t_2 \cap \star u_1) \cup \cdots \cup (\star t_n \cap \star u_m).$$

The results of this operation are those tuples from multiple databases that belong to every pair of relations mapped to R^H and S^H.

The H-PRODUCT Operation

The H-PRODUCT ($\times^H$) operation is to perform CARTESIAN PRODUCTs ($\times$) on every pair of relations mapped to the two operand hyperrelations, and then to perform outer-unions on the results of the CARTESIAN PRODUCT operations to form the final result.

DEFINITION 12 (H-PRODUCT). Given two hyperrelations R^{H} and S^{H} having the sets of tuples $\{t_1, \ldots, t_n\}$ and $\{u_1, \ldots, u_m\}$, the H-PRODUCT ($\times^H$) of R^H and S^{H} is defined as

$$(R^H) \times^H (S^H) = ((\star t_1) \times (\star u_1)) \cup \cdots \cup ((\star t_1) \times (\star u_m)) \cup \cdots \cup ((\star t_n) \times (\star u_m)),$$

where $\times$ is the relational operator CARTESIAN PRODUCT.

In the following, we will explore the properties of operations in the hyperrelational algebra. As we have stated in the Introduction, the properties will serve as a mathematical basis for global query optimization at the algebraic level.

2. Transformation of Hyperrelational Operations

Based on the above definitions, the hyperrelational operations can be transformed from one form to another. Interestingly, even though the hyperrelations stores the schemas of local relations, all the transformation rules in relational algebra are still applicable to the hyperrelational operations. In the following, we give the main theorems on the transformation rules for hyperrelational operations. The proofs of these properties are a direct extension of those given in [28], and hence are omitted here. Note that in the following $attr(X)$ denotes the set of attributes involved in the Boolean expression X, if X is a selection or join condition, and it can also mean the set of attributes of X, if X is a hyperrelation.

PROPERTY 5 (IDEMPOTANCE OF σ^H, π^H).

$$\sigma_C^H R^H \Rightarrow \sigma_{C_1}^H(\sigma_{C_2}^H R^H), \qquad \text{if } C = C_1 \wedge C_2$$
$$\pi_A^H R^H \Rightarrow \pi_A^H(\pi_{A_1}^H R^H), \qquad \text{if } A \subseteq A_1.$$

PROPERTY 6 (COMMUTATIVITY OF σ^H, π^H).

$$\sigma_{C_1}^H \sigma_{C_2}^H R^H \Rightarrow \sigma_{C_2}^H \sigma_{C_1}^H R^H$$
$$\sigma_{C_1}^H \pi_{A_2}^H R^H \Rightarrow \pi_{A_2}^H \sigma_{C_1}^H R^H$$
$$\pi_{A_1}^H \sigma_{C_2}^H R^H \Rightarrow \sigma_{C_2}^H \pi_{A_1}^H R^H, \qquad \text{if } attr(C_2) \subseteq A_1$$
$$\pi_{A_1}^H \pi_{A_2}^H R^H \Rightarrow \pi_{A_1}^H R^H, \qquad \text{if } A_1 \subseteq A_2.$$

PROPERTY 7 (COMMUTATIVITY AND ASSOCIATIVITY OF $\bowtie^H, \times^H, \cup^H, \cap^H$).

$$R^H \theta S^H \Rightarrow S^H \theta R^H, \qquad \theta \in \{\bowtie^H, \times^H, \cup^H, \cap^H\}$$
$$R^H \theta S^H \theta T^T \Rightarrow (R^H \theta S^H)\theta T^H, \qquad \theta \in \{\bowtie^H, \times^H, \cup^H, \cap^H\}$$
$$\Rightarrow R^H \theta (S^H \theta T^H).$$

PROPERTY 8 (DISTRIBUTIVITY OF σ^H AND π^H OVER $\bowtie^H, \times^H, \cup^H, \cap^H$).

$$\sigma_C^H(R^H \bowtie^H S^H) \qquad \text{if } C = C_{R^H} \wedge C_{S^H}$$
$$\Rightarrow \sigma_{C_{R^H}}^H(R^H) \bowtie^H \sigma_{C_{S^H}}^H(S^H)$$
$$\sigma_C^H(R^H \times^H S^H) \qquad \text{if } C = C_{R^H} \wedge C_{S^H}$$
$$\Rightarrow \sigma_{C_{R^H}}^H(R^H) \times^H \sigma_{C_{S^H}}^H(S^H)$$
$$\pi_A^H(R^H \bowtie_C^H S^H) \qquad attr(C) \subseteq A$$
$$\Rightarrow \pi_{A_{R^H}}^H(R^H) \bowtie_C^H \pi_{A_{S^H}}^H(S^H),$$

$$\pi_A^H(R^H \times^H S^H)$$
$$\Rightarrow \pi_{A_{R^H}}^H(R^H) \times^H \pi_{A_{S^H}}^H(S^H)$$
$$\sigma_C^H(R^H \theta S^H) \qquad \theta \in \{\cup^H, \cap^H\}$$
$$\Rightarrow \sigma_{C_{R^H}}^H(R^H) \theta \sigma_{C_{S^H}}^H(S^H),$$
$$\pi_A^H(R^H \theta S^H) \qquad \theta \in \{\cup^H, \cap^H\}$$
$$\Rightarrow \pi_{A_{R^H}}^H(R^H) \theta \pi_{A_{S^H}}^H(S^H),$$

where in this property C_{R^H} and C_{S^H} are the selection conditions involving only R^H and S^H, respectively, and A_{R^H} and A_{S^H} are the projected attributes belonging to only R^H and S^H, respectively.

PROPERTY 9 (TRANSFORMATION RULE BETWEEN $\cup^H$, $-^H$, AND $\cap^H$).

$$X \cap^H Y \equiv (X \cup^H Y) -^H ((X -^H Y) \cup^H (Y -^H X)),$$

where X and Y are two hyperrelations.

In the relational algebra the set of operations $\{\sigma, \pi, \cup, -, \times\}$ is a *complete* set, i.e., any other relational operations can be expressed as a composition of operations in this set. Similarly, the complete set of the hyperrelational algebraic operations is $\{\sigma^H, \pi^H, \cup^H, -^H, \times^H\}$. The proof is also similar to that of the relational case.

3. Examples

We give a few examples to show how hyperrelational algebra works for expressing global queries in a multidatabase environment. The databases used in these examples are those given in Fig. 5, and the hyperrelations are shown in Fig. 11 through Fig. 13. SQL will be the query language used in the examples.

Query 1. Find the names of the students who take the database course. The SQL of this query is as follows:

```
Select   FN, LN
From     STUDENT, COURSE, TAKE
Where    STUDENT.StudNo=TAKE.SNo
and      COURSE.CNo=TAKE.CNo
and      COURSE.CName='Database'
```

In this query, the relations STUDENT, COURSE, and TAKE are the hyperrelations. This global query can also be expressed in hyperrelational algebra as

$$\pi_{FN,LN}^H(\sigma_{COURSE.CName='Database'}[(STUDENT \bowtie_{StudNo=SNo}^H TAKE) \bowtie_{CNo=CNo}^H COURSE]).$$

Query 2. Find the names of the courses taken by only female students or by only male students.

The SQL of this global query can be written as follows:

```
Select  CName
From    COURSE C
Where   Not Exists
        ( Select *
          From  TAKE T, STUDENT S
          Where C.CNo=T.CNo
          and   T.SNo= S.StudNo
          and   S.Gender='Female')
Union   Not Exists
        ( Select *
          From  TAKE T, STUDENT S
          Where C.CNo=T.CNo
          and   T.SNo= S.StudNo
          and   S.Gender='Male')
```

The algebraic expression of this global query is

$$\{[\pi_{CNo}\ COURSE - \pi_{CNo}(\sigma_{Gender='Female'}\ STUDENT \bowtie_{StudNo=SNo}\ TAKE)] \cap \pi_{CNo} TAKE\}$$

$$\cup\{[\pi_{CNo}\ COURSE - \pi_{CNo}(\sigma_{Gender='Male'}\ STUDENT \bowtie_{StudNo=SNo}\ TAKE)] \cap \pi_{CNo}\ TAKE\}.$$

Using the hyperrelational algebra, a global query can be expressed precisely based on the hyperrelations. The expansion of the hyperrelational operations (to relational operations), not shown in the examples as the details have been given in the previous subsections, will allow the local databases to process these queries and return the correct results. Based on the transformation rules given above, a global query can be equivalently converted to different forms. Optimization of a global query can be realized at the multidatabase systems level by choosing from the equivalent algebraic expressions the one that incurs the least execution cost.

E. A Comparison with Related Works

Not much work has been proposed in this category. As we mentioned earlier, although schema integration in the context of semantic conformation has been studied for a long time, it has a different goal than what we are trying to achieve in multiple autonomous databases environments. Recently, Agarwal *et al.* [2] proposed a *flexible relation* approach to the integration of multiple relational databases. This model extended the classical relational model by providing support for inconsistent data. Inconsistent data in their paper refers to those attribute values of an entity, such as a person's age, that in different databases are not the same. For example, a database says that a person is 19 years old, while another says that the same person is 20 years old. Flexible mechanisms were provided to resolve the data inconsistency problems. An algebra was also proposed for this flexible relation model. Their focus, however, using our terminology, was only on the "value-versus-value conflict" issues discussed in this paper. Reference [20] is an earlier example trying to resolve the same problems

(terned *domain mismatch* problems in the latter). Although an algebra was provided in both works, they cannot be applied to resolving the schema integration issues, nor can it resolve the global query optimization issues at the schema level, because the scope of their algebra is limited to value inconsistency problems. Schema conflict issues, the key part of our work, were not dealt with in their papers.

Zhao *et al.* [85] proposed a universal relation approach to manage the schemas of federated databases. They extended the notion of *universal relation* to a heterogeneous multiple database environment by keeping all federated attributes in a single table. The main purpose was to facilitate the translation of a global query to local queries. The resolution of structural conflicts of local schemas, however, was not a concern of the paper. Also, no algebra based on their approach was proposed. Hence, optimization of a global query at the multidatabase system level could not be supported easily in this approach.

We consider the *multirelational approach* proposed by Grant *et al.* in [33], a work most similar and comparable to ours. Their approach, however, was designed for multidatabase systems without the need of an integrated global schema. All kinds of representational conflicts were specified in a query issued by the global user. Hence, the user must be knowledgeable enough to define clearly the differences between schemas of data. The major advantage of their work was that the schema integration issues were avoided. An algebra, named the *multirelational algebra*, associated with their model was proposed. This algebra played a role very similar to that of hyperrelational algebra. (For more details of the work, the reader is directed to [55, 56, 58].) As this work is very similar to the hyperrelational algebra (except that the application domains are different), an in-depth comparison of the two works is given in the following.

- Both algebras are based on the relational algebra and both are an extension of the relational algebra.
- Operations in both algebras are convertible to relational operations.
- All transformation properties in relational algebra remain valid in both algebras.
- Both algebras function as a mathematical basis for query transformation and optimization.

The differences:

- Supporting environments are different. The multirelational algebra is designed to support an environment that does not need a global schema (i.e., the multidatabase language approach as mentioned earlier). The hyperrelational algebra, however, is dedicated to those tightly integrated systems in which a global schema exists. It is based on this schema that a user query is issued.
- Definitions of "multirelation" and "hyperrelation" are different. In the multirelational algebra, a multirelation is a set of local relations that are involved in a user query. In other words, it is dependent on the query and, hence, there is not a fixed structure for a multirelation. In the hyperrelational algebra, however, both the structure and the contents of a hyperrelation are explicitly defined. They are completely determined by the integrated global schema. Each hyperrelation has a clearly defined semantics. Structurally, a hyperrelation is the

same as a relation; contentwise, it has the schemas of all semantically equivalent relations in the multidatabase system.

- Outputs of an algebraic expression in two cases are different. The result of a multirelational algebraic expression is still a set of relations, dissimilar to the relational algebra. The schemas of the relations in the result may not even be the same. For instance, $M\ Join(\{R_{11}, R_{12}, \ldots, R_{1n}\}, \{R_{21}, R_{22}, \ldots, R_{2m}\} : C)$ is a multirelational join operation on two sets of relations based on the join condition C. The result of this operation is $\{Join(R_{11}, R_{21} : C), \ldots, Join(R_{11}, R_{2m} : C), \ldots, Join(R_{1n}, R_{2m} : C)\}$. The schemas of these relations may not be the same. Hence, the result relations may have very different schemas. The user needs to interpret the semantics of the result, just like being responsible for specifying all necessary schema/data conflicts in a query. In the hyperrelational algebra, the result of an algebraic expression is still a relation. Hence, the semantics of the result relation is very clear. The user need not refer to any local relations to understand the semantics of the result.
- Requirements on union compatibility are different. In multirelational algebra, the operand multirelations need not be union compatible in executing a set operation. For example, the result of executing MUnion($\{R\}$, $\{S\}$) (where MUnion is a multirelational union operator) is $\{R, S\}$ if the local relations R and S are not union compatible. The union compatibility is enforced, however, when a multirelational operation is converted to relational operations. In contrast to their work, the operand hyperrelations must be union compatible if a set operation is executed. This difference is caused by the fact that the hyperrelational operations are performed on hyperrelations, each of them having a well-defined schema. This in a sense is the same as the relational algebra in which each relation has a well-defined schema. Union compatibility is enforced in these cases, as mentioned in the discussion of the H-SELECTION operation, to avoid performing set operations on semantically completely different relations. When a hyperrelation is converted to relational operations, the union compatibility requirement is relaxed because all operand relations are semantically equivalent.

IV. OPTIMIZATION AT THE EXECUTION STRATEGY LEVEL

In this section, we discuss multidatabase query optimization at another key level, the execution strategy level. The optimization at the algebra level focuses on the optimization that can possibly be achieved during the transformation of a global query to local queries. The execution strategy level optimization follows the task at the algebra level by finding an optimal execution strategy for the global query. Several important considerations at this level will be discussed in this section. A few optimization strategies and a performance study of them will also be presented.

In an MDBS, autonomous and heterogeneous database systems are integrated to serve for users who want to retrieve data from the component databases, i.e., the participating databases. Execution of a global query often involves the systems of these participating databases. For convenience in presentation, we will refer to these systems as the participating database systems

(PDBS). As each PDBS is autonomous, the MDBS has no control over the internal optimization decisions within each PDBS. This makes query optimization at the execution strategy level a difficult task. Despite the importance of the work, it has so far received little attention. As the query optimizer of each PDBS is a "black-box" to the MDBS, the main focus of the past research was on how to regenerate in the MDBS the cost model of the query optimizer of each PDBS [24]. Du *et al.* [24] proposed a calibrating database that is synthetically created so as to reasonably deduce the coefficients of cost formulas in a cost model. Works in [27, 29, 61, 68, 86] are other examples of having a similar approach. The effectiveness of all these schemes, however, heavily relies on the precision of the regenerated cost models of the PDBSs. Our study reveals that many factors affect the execution cost and the processing time of a query in a PDBS, which makes query optimization using this (regenerating cost models) approach unrealistic. For instance, one of the factors is that some commercial DBMSs automatically cache a large portion of a recently accessed relation in memory for some time and others only cache a very small portion or not cache at all for such data. For those DBMSs that do, a subsequent query, if it happens to involve the cached relation(s), is processed much faster than the previous query. The two dramatically different processing times for the same operation on the same relation could make the regenerated cost model of a PDBS (which is usually based on those times) significantly deviate from its real execution time. A global query optimizer using these inaccurate cost models could lead to making incorrect decisions. An even worse situation is that there is no way to predict how long such a relation will stay in the cache (since it depends on many other factors), which lets the regenerated cost model in the MDBS simply be untrustful.

Another factor that worsens the situation is that in many occasions multiple joins are performed in a PDBS. However, there is no way to predict the execution order of these join operations. A different execution order of the joins could lead to a dramatically different execution time. None of the previously proposed methods covers this problem.

Recently, Du *et al.* [25] proposed a method for reducing query response time by using tree balancing. This method is based on an assumption that all PDBSs have the same processing rate for database operations. As all PDBSs are equal in terms of processing speed, it becomes unnecessary to estimate the cost models of the PDBSs in their method. Therefore, the issue of regenerating cost models in that paper is ignored. In reality, however, having to have cost models for optimization is still a major problem for their method to be realistic.

In this section, we will propose an optimization strategy at a level lower than that of our previous work but still without the need of regenerating the cost models of the PDBSs. We propose three query processing strategies for a truly autonomous multidatabase environment. The first two strategies are based on traditional optimization concepts and skills and mainly designed for the purpose of comparing with the third strategy. We assume that all global query results are eventually collected by the MDBS and sent from the MDBS to the global users. The first strategy is a first-come-first-serve (FCFS) strategy. The MDBS performs joins on relations that are received first. In other words, an intersite join in the MDBS is performed as long as both of its operand relations have

been received. As it is normally hard to predict the traffic of a network as for when a hot spot will occur, it is difficult to arrange the join sequence in advance to achieve a low response time. The FCFS strategy actually accommodates well to an unpredictable traffic environment and is likely to give a low response time in that circumstance. However, the total execution cost incurred by using this strategy can be high because join order is not arranged at all to minimize resource consumption. Our second strategy is a greedy strategy. In this strategy, we select at each stage the join that will produce relatively the smallest result. By "relatively" we mean the ratio of the result size of a join to its input operand size is a minimum. (We will state the reason later when the algorithm is presented.) Intuitively, this second strategy incurs a lower processing cost since a better option is chosen for each join.

In both of these two strategies, each intersite join operation is entirely performed in the MDBS. The third strategy is designed to further reduce the MDBS workload by leaving part of a join task to a PDBS. Its key idea is based on the sort–merge join concept. The traditional sort–merge join algorithm contains two major steps: *sorting* relations and *merging* them to get the join result. In our design, however, the sort part of an intersite join is accomplished in the PDBSs, that is, all subquery results output from the PDBSs are sorted, and the sort is on the attribute over which the intersite join (in the MDBS) is performed. Then the sorted relations are sent to the MDBS to perform the merge part of an intersite join. Asking a PDBS to perform a sort is achievable without violating its autonomy by sending an SQL query with an ORDER BY command from MDBS to the PDBS (to be detailed in a later section). Using this method, the sort of a join is performed in parallel in PDBSs. As the sort part of a sort–merge join algorithm dominates the join cost, leaving it to a PDBS instantly reduces the workload of the MDBS. In addition, performing the sort processes in parallel in different PDBSs also speeds up the join process. For operations involving only local relations of a PDBS, they are executed locally in the PDBS. In the next section, we will discuss that it is unwise to assign such local operations to any other PDBS.

Our focus of this strategy is on the assignment of the maximum number of pairs of relations that can be merge joined in the MDBS. The merge part is performed in the MDBS because an intersite operation unavoidably involves resolution of data discrepancy [21, 43, 51]. As merge is a simple process with a linear complexity, a MDBS's workload is minimized by employing these merge joins. However, a difficulty of this strategy arises as follows. If we consider that the *intersite* join operations of a query are organized into a query tree, the input data to a bottom level node of the tree are in fact directly from the output of a PDBS. The sorted results from the PDBSs could only facilitate the joins at the bottom level of the query tree. How to make the best use of the sorted results from the PDBSs for the higher level intersite joins is crucial for minimizing resource consumption in the MDBS. A good design is required to take full advantage of the sorted results of PDBSs for these higher level joins. We will discuss these issues in detail. The advantages of our methods are (1) unlike the previous approaches, our strategies can proceed without the need of regenerating local cost models while optimizing a multidatabase query, (2) the MDBS workload is minimized, and (3) our method is much simpler

and easier to implement than those complex cost model regeneration methods proposed in the past.

Finally, we note that the reason for not considering the other two well-known join algorithms, i.e., the nested-loop algorithm and the hash-based algorithm, in our method is because they are either impossible or inefficient to be implemented in a MDBS environment. We will clarify this point further also in a later section.

A. Assumptions

From the information exchange point of view, we can classify the participating database systems into three categories [24]:

- *Proprietary DBMSs.* The PDBSs can provide all the relevant information on cost functions and database statistics to the MDBS.
- *Conforming DBMSs.* The PDBSs provide database statistics but are incapable of divulging the cost functions.
- *Nonconforming DBMSs.* The PDBSs are incapable of divulging either the database statistics or the cost functions.

Our discussion in the following is dedicated to the *nonconforming DBMSs* environment. The reasons are, first, it is the least studied environment up to now. All past query optimization methods have not been designed for this environment [24, 25, 27, 29, 61, 68, 86]. Second, it is the most realistic environment considering the fact that almost all DBMSs today either have had or will be supplied with an accessing interface to the World Wide Web. There are also vendors utilizing *data warehousing* and *digital library* techniques to provide users with various data sources. All these sources are basically heterogeneous, same as in the nonconforming DBMSs environment. Both of the other two environments discussed in [24] require a major modification/enhancement to the core modules of the existing DBMSs.

B. Where Should an Intersite Operation Be Performed?

One difficulty of processing a multidatabase query is that data conflicts may exist. While performing an intersite operation (such as join, union), the conflicts of data must first be resolved. This is normally achieved by converting local data to a certain globally recognized format, such as that specified by the global schema. The mapping functions (or mapping tables) of heterogeneous data and the mapping mechanisms (software) all reside in the MDBS. Can the mapping tables for resolving data discrepancy can be sent to a PDBS to let the PDBS perform the tasks that are otherwise performed in the MDBS? The answer is no because not only the mapping tables but also the mechanisms (software) are needed to execute those conversions. Without the mechanisms, a PDBS is still unable to convert data of different types. All past research mentioned earlier ignored this issue in the design of an intersite join optimization strategy.

However, is it feasible to let the MDBS convert the data and the PDBS perform the intersite joins after data discrepancy is resolved? This issue should be examined in more detail. Basically, only two query execution schemes are

feasible: (1) PDBSs perform the operations, and the MDBS converts the data, and (2) the MDBS performs the operations and data conversion. Let us use an example to illustrate these two schemes. Assume that $R \bowtie_1 S \bowtie_2 T$ is a multidatabase query, where R, S, and T belong to distinct databases.

1. *PDBSs perform the operations.* Assume that the databases containing R, S, and T are PDBS_R, PDBS_S, and PDBS_T, respectively. A strategy of this category contains the following steps:

   (a) PDBS_R sends R to MDBS for data conversion.
   (b) MDBS converts R to a format understandable to PDBS_S.
   (c) MDBS sends R to PDBS_S.
   (d) PDBS_S performs $R \bowtie_1 S$.
   (e) PDBS_S sends the result, letting it be RS, to MDBS for another conversion.
   (f) MDBS converts RS to a format understandable to PDBS_T.
   (g) MDBS sends RS to PDBS_T.
   (h) PDBS_T performs $RS \bowtie_2 T$.
   (i) PDBS_T sends the result, letting it be RST, to MDBS for conversion.
   (j) MDBS converts the result and sends it back to the user.

Certainly, we can also send S to PDBS_R to perform the join, or even perform $\bowtie_2$ before $\bowtie_1$. However, as long as joins are performed in PDBSs, the required communications between PDBSs and the MDBS as well as the data conversion to be performed in the MDBS are basically the same.

2. *MDBS performs the operations.* The same query will be executed in the following steps:

   (a) PDBS_R, PDBS_S, and PDBS_T send R, S, and T, respectively, to MDBS simultaneously.
   (b) MDBS performs conversion on these relations.
   (c) MDBS performs $\bowtie_1$ and $\bowtie_2$.
   (d) MDBS sends the result to the user.

Obviously, the first scheme involves too much communication between each PDBS and the MDBS, causing serious overhead on query processing. It can be aggravated in today's applications that are accessed through the World Wide Web (WWW), a key feature desired by the users and implemented by most DBMS vendors in their current/next generation DBMSs. On the other hand, the second scheme does not require as much communication. In addition, many joins performed in MDBS can be reduced to simply a merge operation if the relations (subquery results) sent out from PDBSs are sorted. Hence, the second scheme is a preferred option. However, the MDBS's workload is heavy in this strategy. How to reduce query cost becomes crucial to the performance of a MDBS.

As the MDBS is heavily involved in global query execution, a major concern of optimization is to minimize the consumption of MDBS system resources so that it will not be easily overloaded during query processing. In other words, the optimization goal at this level should be to minimize the query execution cost so that the MDBS can serve a maximum number of concurrent users.

Other assumptions made are, first, the autonomy of a PDBS should not be violated or compromised for any reasons. The MDBS can only interact with a PDBS and utilize a PDBS's processing power under this restriction. Through this assumption, a running database system is ensured to be able to participate in a multidatabase without the need of modifying any part of its system code for the query optimization purpose. The second assumption is that all local operations (i.e., involving relations of a single database) of a query must be performed locally. A multidatabase (global) query after compilation is decomposed into subqueries, each being processible in a PDBS. For instance, a selection operation on a local relation should be performed before the local relation is sent to the MDBS (to participate in an operation involving relations of multiple databases). The processed result is then transmitted to the MDBS. Certainly, it is possible that the result size of a local operation (such as a join) becomes larger than its input size such that it seems to be better to send its input relations, instead of its result, to the MDBS and let the MDBS perform the (join) operation so as to reduce communication overhead. We do not consider this approach because our major concern here is that the MDBS could easily be overloaded if all locally processible tasks are sent to the MDBS. Since our optimization goal is to minimize the query execution cost in the MDBS, processing local operations locally becomes a natural choice.

C. Different from the Issues in Traditional Database Systems

As each PDBS is like a processor in a parallel as well as in a distributed database system, it is natural to ask whether our problem is the same as query optimization in parallel database systems, or in homogeneous distributed database systems? Also as all received relations are processed solely in the MDBS (as the second scheme in the previous section), how this part is different from a query optimization in a centralized database system is another question that needs explanations. We answer these questions in the following.

- *Different from query optimization in parallel database systems.* In a parallel database system, each task (such as sort and join) can be assigned to multiple processors [12, 37, 62]. There is apparent parallelism between consecutive tasks or even within a single task. In the MDBS environment under discussion, local tasks are processed locally and intersite operations are all processed in the MDBS. Therefore, a two-level data processing mechanism, one in PDBSs and the other in the MDBS, is employed in our design. Compared to a parallel database system where tasks can be assigned freely to any processors, the environment under study is much more restrictive on the collaboration of PDBSs for task assignment.
- *Different from query optimization in distributed database systems.* Similar to a parallel database system, a local system of a homogeneous distributed database system can process data sent from any other database freely, except that communication cost due to data transmission through communication network needs to be taken into account. No restrictions (due to heterogeneity) on task assignment to processors need to be considered. Hence, it is very different from the issue discussed in our environment.

• *Different from query optimization in centralized databases.* A key difference is that in a centralized database hash-based join methods are normally considered more efficient than sort–merge join methods [12, 22]. This is because the sorting process has a complexity of $O(n \log n)$ for n records, which is much higher than the linear hash process when n is large. Even if sort–merge join is sometimes used when the distribution of data is highly skewed where hash join becomes less efficient, we do not sort all the base relations before joining them. In a multidatabase environment under study, however, since sorting the subquery results is performed in each PDBS rather than in the MDBS, it is desired to have them all sorted (on an attribute on which the next join is performed) before they are sent to the MDBS to alleviate the burden of and facilitate the global joins in the MDBS. The focus becomes to design an optimization algorithm which is able to utilize as many sorted results from PDBSs as possible—a still different problem from that in traditional database systems.

From the above discussion, we see that the past query optimization methods cannot be directly applied to the new environment. New methods must be designed.

D. Two Scheduling Strategies Not Involving PDBS

Figure 14 is a "reduced" join graph of a multidatabase query in a MDBS. By reduced graph, we mean that operations and relations of a local query (subquery) are represented by simply one node, which is in fact the result of a local query. Hence, each node is a result relation from a PDBS. In this particular example, the number of PDBSs involved in this multidatabase query is 11. Each edge between two nodes stands for a join on the two corresponding relations. Each node is labeled by the name of the relation and each edge labeled by the join attribute. Formally, let us refer to such a graph as $G(V, E)$, where V and E are the sets of nodes and edges, respectively. Let the number of nodes (relations) and edges (joins) be $|V|$ and $|E|$, respectively.

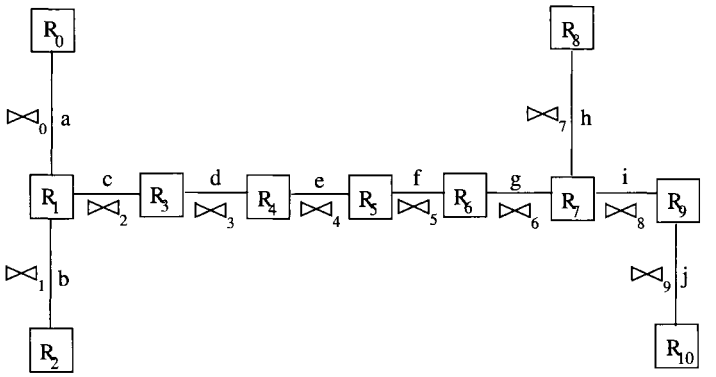


FIGURE 14 A join graph.

1. The First-Come-First-Serve (FCFS) Strategy

A straightforward method of joining the incoming relations in the MDBS is to use a first-come-first-serve (FCFS) strategy, meaning whichever relations that are received first by the MDBS are joined first in the MDBS. For instance, two joins $R_1 \bowtie R_2$ and $R_3 \bowtie R_4$ are involved in a query. If R_1 and R_2 are received in the MDBS earlier than R_3 and R_4, MDBS will perform $R_1 \bowtie R_2$ before $R_3 \bowtie R_4$. This strategy represents a case that we do not have any knowldge at all about the possible execution order of joins. MDBS will perform a join on whichever relations that have arrived at the MDBS. Intuitively, this is a good strategy because we normally have no control over the traffic of a network. A predetermined sequence of joins may not give a good performance if a relation that is scheduled to be joined early is blocked by the traffic of a network. We studied the performance of this strategy and the results will be presented in a later section. In Fig. 15 we give a formal presentation of the FCFS strategy. The complexity of this strategy is $O(|E|)$, where $|E|$ is the number of edges of a query graph $G(V, E)$, because each edge (i.e., join) need only be visited once in the algorithm.

2. The Greedy Scheduling (GRS) Strategy

The GRS strategy is an iterative algorithm that selects in each iteration the join that has the lowest join selectivity factor at the current stage. This algorithm is named so because it produces the least amount of data at every join stage. The details of this strategy is listed in Fig. 16.

A number of criteria could be used to select a join [12, 60] in a greedy multijoin algorithm. Three frequently seen criteria include:

1. **min**(JS): select a join with the minimum join selectivity factor (producing *relatively* the least amount of data, or $|R_i \bowtie R_j|/(|R_i| * |R_j|)$ being minimum).
2. **min**$(|R_i| + |R_j|)$: select a join whose total amount of input data is minimum.
3. **min**$(|R_i| * |R_j| * JS)$: select a join that produces the least amount of data, i.e., $|R_i \bowtie R_j|$ being minimum.

```
FCFS strategy :
  Input : a join graph G(V, E);
  Output : a join sequence Q;

  Repeat until |V| = 1 ;
     Begin
        Choose R_i ⋈ R_j such that R_iR_j ∈ E and R_i and R_j have been arrived MDBS
           (if more than one joins can be performed (their operands are ready),
           an arbitrary one is selected);
        Add R_i ⋈ R_j into Q ;
        Update graph G by merging nodes R_i and R_j into R_min(i,j) ;
     End Begin
  End Repeat
```

FIGURE 15 The First-Come-First-Serve (FCFS) strategy.

GRS strategy :

Input : a join graph $G(V, E)$;
Output : a join sequence Q;

Repeat until $|V| = 1$;
  Begin
    Choose $R_i \bowtie R_j$ such that $\overline{R_i R_j} \in E$ and it has the lowest join selectivity factor at the current stage;
      (if more than one such joins are available, an arbitrary one is selected;)
    Add $R_i \bowtie R_j$ into Q;
    Update graph G by merging nodes R_i and R_j into $R_{min(i,j)}$;
  End Begin
End Repeat

FIGURE 16 The Greedy Scheduling (GRS) strategy.

These criteria aim at minimizing the intermediate relation sizes so as to achieve a lower total execution cost. Each of them has its strength and weakness. The third criterion may look attractive at the first sight. However, the smallest join result (i.e., $|R_i \bowtie R_j|$) may simply be incurred by the join of two small relations. The relative gain $|R_i \bowtie R_j|/(|R_i| * |R_j|)$ may be much less than that of the first criterion. A performance study of these criteria has been conducted in [60] and the result shows that none of them outperforms the others in all circumstances. In this paper, we simply adopt the first criterion, min(JS), in the GRS strategy.

The complexity of the GRS strategy is $O(|E|^2)$, where $|E|$ is the number of edges, because it needs $|E|$ iterations and each iteration checks on at most $|E|$ edges.

E. PDBS Sharing Workload with the MDBS

1. Utilizing PDBS Processing Power?

The hash, the sort–merge, and the nested-loop join algorithms are the most frequently used algorithms in traditional database systems. Especially, the hybrid hash join algorithm has been considered the fastest join algorithm among all [22]. Now the question is in a nonconforming DBMS environment can these algorithms be similarly implemented and still be as efficient as in a centralized database environment? We briefly discuss this issue in the following.

Assume that R_A (A1, A2, A3) and R_B (B1, B2, B3) are relations of different databases, and a multidatabase query is to join these two relations on A1 and B1 (i.e., $R_A.A1 = R_B.B1$). To implement a hash join algorithm is this environment, both R_A and R_B should be hashed in their own database systems before the hashed data buckets are sent to the MDBS to proceed on the probing phase.[7] However, as each PDBS is an autonomous system, the MDBS has no control over the hash function used in the PDBSs. The two hash functions used in the two databases are likely different. If the MDBS simply lets each PDBS hash its data in its own way, there may not exist any matching data buckets between the two relations. Besides, there is not a database language for the MDBS to

[7] A hash join can be split into two phases: hash and probe.

instruct a PDBS to perform simply a hash operation. In order to utilize PDBSs' processing power, hash join must be implemented by sending a set of queries to each PDBS to partition tuples of a local relation into buckets of different value ranges. Then corresponding buckets of the relations can be formed and joined in the MDBS. In the above query, for instance, the MDBS sends the following set of queries to database A (the one containing R_A):

```
select *    From R_A   Where R_A.A1 ≤ 20
Select *    From R_A   Where 20 ≤ R_A.A1 ≤ 40
Select *    From R_A   Where 40 ≤ R_A.A1 ≤ 40
Select *    From R_A   Where 60 ≤ R_A.A1 ≤ 80
Select *    From R_A   Where 80 ≤ R_A.A1
```

In this example we have assumed that the data type of attribute A1 is an integer. The relation is hashed into five buckets containing tuples having values in the ranges $(-\infty,20]$, $(20,40]$, $(40,60]$, $(60,80]$, $(80,+\infty]$, respectively. Similarly, the MDBS sends the same set of queries to database B (the one containing R_B) except that R_A in the queries is changed to R_B and A1 is changed to B1. Assume that R_A^1, R_A^2, R_A^3, R_A^4, and R_A^5 are the results of the five SQL queries on R_A and R_B^1, R_B^2, R_B^3, R_B^4, R_B^5 are those on R_B. Then R_A^i and R_B^i (for $1 \leq i \leq 5$) are the corresponding hashed buckets of the two relations and the probing phase can be executed in the MDBS. This way of hashing, however, requires that each PDBS executes as many queries as the number of hashed buckets. When hundreds or even thousands of hashed buckets are required, it will lead to a significant overhead for compiling and processing this large set of queries. This shortcoming prohibits PDBSs from sharing the workload with the MDBS in implementing a hash join method.

To implement a sort–merge algorithm in the same environment, only one query is needed to send to each PDBS. The query requires that the resulting relation be sorted in order on the join attribute. In the above example, a query sent to database A will be like this:

```
Select     *
From       R_A
Order By   R_A.A1
```

The same query but asking for a sorted order on attribute B1 will be sent to database B. The results obtained from these two databases can then be easily merged by the MDBS to obtain the join result. This sort–merge approach is much simpler and more efficient than the hash join approach. As merge is a simple and linear process, the workload added onto the MDBS is extremely light.

The nested-loop join algorithm is the easiest to implement since it only requires the MDBS to send the following query to each PDBS to access the data:

```
Select  *
From    R_A
```

The PDBS need not process the relation in any manner. Since it does not fully utilize the PDBS's processing power, we do not consider it as a viable approach.

One may wonder whether it is possible for database A and database B to send one block of data at a time to the MDBS such that the MDBS simply join these two blocks of data in memory. In this manner, the MDBS need not save the entire relations R_A and R_B to disk in order to run the nested-loop join algorithm, hence helping to reduce the workload of the MDBS. This scheme however is infeasible because the required low-level handshaking mechanism violates the autonomy of a PDBS. Thus, it is not implementable in a nonconforming DBMSs environment.

The following table summarizes the features of the three join algorithms implemented in a nonconforming DBMS environment. It is apparent that the sort–merge join is the most suitable algorithm for joins in a multidatabase environment, and therefore it is adopted as the global join algorithm in this research.

Algorithms	Consumed PDBS processing power	Consumed MDBS processing power
Hash join	Very high	Medium
Sort-merge join	Medium	Low
Nested-loop join	Low	Very high

F. The Maximum Merge Scheduling (MMS) Strategy

In order to reduce the burden of a MDBS, the best we can do is to ask the PDBSs to sort their local results before they send the results to the MDBS. When the MDBS receives the sorted relations, it only needs to merge them rather than starting from the very beginning to hash or sort the received relations to perform the join. In this way, the workload of a MDBS is highly minimized. So the MDBS's task is to determine the maximum number of pairs of sorted relations that can be merged (i.e., join) and instruct the corresponding PDBSs to sort their results. If we take the join graph in Fig. 14 as an example, the maximum number of pairs would be five, as shown in Fig. 17. Certainly, there may be other choices on the pairs of relations, such as choosing R_1 and R_2 as a pair rather than the pair R_0 and R_1. Based on the matching, the MDBS asks the PDBSs to sort their results on the given join attributes. For instance, the MDBS will ask the PDBS having R_0 to sort it on attribute a and the same to the PDBS having R_1. In this way, there will be five joins simplified to a simple "merge" process in the MDBS, greatly reducing the burden of the MDBS. In our algorithm (to be presented shortly), we group those larger relations as a pair when we have multiple choices because by sorting larger relations, the PDBSs share more workload of the MDBS.

An existing algorithm, named the *maximum matching algorithm* [10], can be used to find the maximum number of matching pairs of a given graph. In this maximum matching algorithm, a graph G is called **1-regular** if every vertex of G has degree 1, where the **degree** of a vertex is the number of vertices adjacent to this vertex. Let adjacent edges be those having the same terminal vertex. A **matching** of a graph G is a 1-regular subgraph of G, that is, *a subgraph*

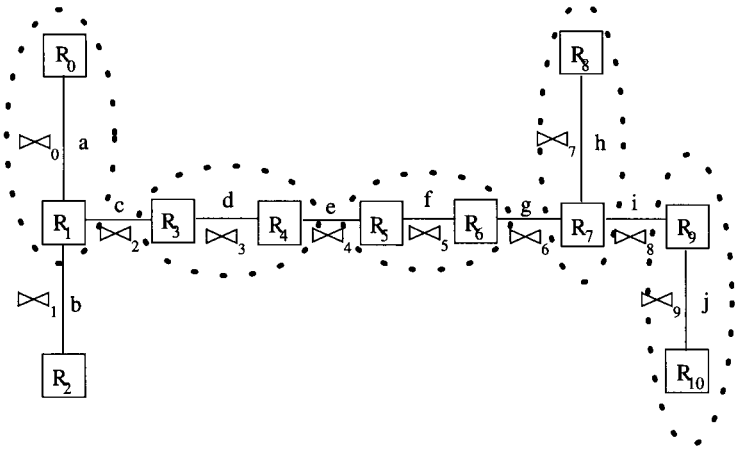


FIGURE 17 One of the maximum matchings found using the maximum matching algorithm.

containing a collection of nonadjacent edges. The **cardinality** of a matching represents the number of nonadjacent edges within the matching. A matching of the maximum cardinality of graph G is called a **maximum matching** of G. For an example, Fig. 17 shows a maximum matching of the join graph. The pairs of nodes that are circled by dotted ovals are the matching pairs. As this maximum matching algorithm is not difficult to infer based on the above description, we do not list here the details of the algorithm. Readers may refer to [10] for details.

This existing algorithm, however, does not work while there is more than one join on the same attributes. Figure 18 gives another example in which the relations are the same but the join attributes are different from those of Fig. 14. Obviously the maximum matching for our purpose should be the two groups as shown in the graph, because it allows seven joins to be implemented by a sort–merge join rather than five by using the maximum matching algorithm.

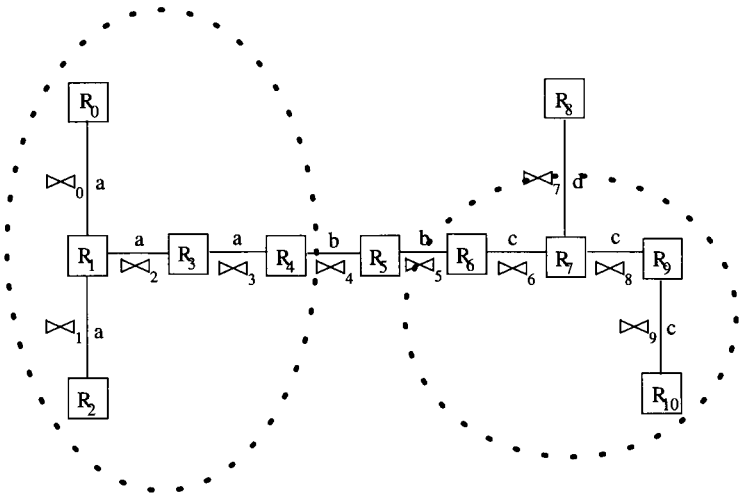


FIGURE 18 The same join graph with a different set of join attributes.

Our maximum merging strategy (MMS) is designed to work for such general query graphs.

We take a simple and yet effective approach to maximize the number of matching pairs in order to reduce the complexity of our algorithm. The basic idea is that we first use the existing maximum matching algorithm to find the maximum number of relation pairs, and let this graph be G. Then we identify the relations connected through the same join attribute, such as a dotted oval in Fig. 18. For each set, say C, of these relations, we check whether benefit (more matching pairs) is gained if the part of G corresponding to C is replaced by C. If it is beneficial, then they are replaced by C. If the number of matching pairs does not change by replacing them with C, then we check whether the total intermediate result size after the replacement will decrease. If it decreases in size, then the replacement is still adopted because it helps to reduce the workload of the MDBS. The same process continues until all such sets are examined. In order not to miss a large set (i.e., with more nodes), this process should start from the largest set and proceed in a descending order of the set size (number of nodes in a set) because selecting one set of matching relations might affect the selection of a set that is considered later.

Let us illustrate our idea by using the example in Fig. 18. Assume that the maximum matching pairs found by using the maximum matching algorithm is that shown in Fig. 17, but note that their join attributes should be the same as those in Fig. 18. There are three sets of relations, each set of relations being connected through the same attribute. In descending order of the size, they are the set $A = \{R_0, R_1, R_2, R_3, R_4\}$, the set $C = \{R_6, R_7, R_9, R_{10}\}$, and the set $B = \{R_4, R_5, R_6\}$ in which set B is not circled because eventually it is not selected in our result. For convenience, let us denote the set of relations $\{R_i, R_j, \ldots, R_k\}$ that are connected through the same attribute as $\overline{R_i R_j \ldots R_k}$. We start from set A, replace the matching pairs $\overline{R_0 R_1}$ and $\overline{R_3 R_4}$ by the set of matching relations $\overline{R_0 R_1 R_2 R_3 R_4}$, and check whether the number of matching relations in the entire graph increases. Since in this case the number is increased from five to seven, we adopt the new set. Next, we consider the set C. If the matching pairs $\overline{R_5 R_6}$, $\overline{R_7 R_8}$, and $\overline{R_9 R_{10}}$ are replaced by the set $\overline{R_6 R_7 R_9 R_{10}}$, the number of edges is still three. So whether they will be replaced by the new set is determined by the intermediate result size. That is, assuming that the result relation of $R_i \bowtie R_j \bowtie \ldots \bowtie R_k$ is denoted as $R_{ij\ldots k}$, and the size of realtion R is denoted as $|R|$, then the total size of the intermediate result relations (for only the part of involved relations R_5, R_6, R_7, R_8, R_9, R_{10}) before the replacement is $|R_{56}| + |R_{78}| + |R_{910}|$, and that after the replacement is $|R_5| + |R_{67910}| + |R_8|$. If the size become smaller, then the replacement is adopted. Otherwise, it is rejected. If C is adopted, then B need not be checked because only one node, R_5, is left for B. Otherwise, the set B is examined similarly. We list our algorithm details in the following.

Maximum Merging Strategy

Input: join graph $G(V, E)$;
Output: join strategy Q;
Let $|V|$ be the number of vertices of G, and $|E|$ be the number of edges of G;
We denote the join result of $R_1 \bowtie R_2 \cdots R_k$ as $R_{12\ldots k}$, and the size of R as $|R|$;

Use the maximum matching algorithm to find for G a set of matching pairs S_G with the number of matching pairs = M;
Assume that there are I sets of connected vertices. The edges of each set have the same label (i.e., join attribute) and the number of vertices in each set ≥ 3, where $0 \leq I \leq [|V|/3]$;
Each of these sets of vertices is a subgraph of G, and we denote such a subgraph as $G_i(V_i, E_i)$, where $0 \leq i \leq I$;
Let $V_i = \{R_{i1}, R_{i2}, \ldots, R_{i|V_i|}\}$;
Sort G_i's in descending order (i.e., G_1 is the largest set);
Let $Q = \phi$;
For $i = 1$ to I Do
    Begin
    $G'(V', E') = G(V, E) - G_i(V_i, E_i)$;
    Find maximum matching pairs for G' using the maximum matching algorithm; let the number of matching pairs be M';
    CASE
    • $M < M' + |E_i|$
      Begin
      $G(V, E) = G'(V', E')$;
      $M = M' + |E_i|$;
      $Q = Q + \{\overline{R_{i_1} R_{i_2} \cdots R_{i_{|V_i|}}}\}$; /* $R_{i_j} \in V_i$ */
      End
    • $M = M' + |E_i|$
      Begin
      If $\sum_{\forall h,k} |R_{hk}| + \sum_{\forall f,g} |R_{f,g}| > \sum_{\forall h} |R_h| + |R_{i_1 i_2 \cdots i_{|V_i|}}|$, *where* $\overline{R_h R_k}$, $\overline{R_f R_g} \in S_G$ and $R_h \in V'$, and $R_k, R_f, R_g \in V_i$
      /* $R_H R_k$ is a matching pair of G that is broken by the selection of G_{ij} $\overline{R_f R_g}$ is a matching pair of G residing in G_i. */
      then /* accept G_i */
        $G(V, E) = G'(V', E')$;
        $M = M' + |E_i|$;
        $Q = Q + \{\overline{R_{i1} R_{i2} \ldots R_{i|V_i|}}\}$;
      End
    • $M > M' + |E_i|$
      Begin
      Do nothing;
      End
    End-of-For
Find matching pairs of $G(V, E)$ using the maximum matching algorithm and let $S_G = \{\overline{R_{11} R_{12}}, \overline{R_{21} R_{22}}, \cdots, \overline{R_{m1} R_{m2}}\}$ be the set of matching paris;
/*$G(V, E)$ might have been update so that a recalculation is still needed. */
$Q = Q + S_G$;
/* End of algorithm */

At the end of this algorithm, the relations that will be merge joined are determined. The results of these merge joins, however, could be unsorted on the attribute to be joined next. In general, these later stages of joins could have either one or two of the input relations being unsorted on the join attribute.

We process these joins based on two simple rules:

1. if one of the input relations is sorted on the join attribute, then perform an *order-preserving hash sort* on the unsorted relation and then do the merge join. We term this join a *semi-merge join to* distinguish it from a merge join.
2. if both of the relations are unsorted, then perform a hybrid hash join.

The reason why we employ an order-preserving hash sort in rule 1 is as follows. In order to minimize the number of disk accesses in the MDBS, when one of the input relations is already sorted, the second relation is only sorted at the block (bucket) level. This means that after the order-preserving hash sort, if $B_0, B_1, \ldots, B_n$ are the produced blocks, then records $b_0, \in B_0, b_1 \in B_1, \ldots, b_n \in B_n$ are in order, but records within each block are not necessarily in order. Then, joining records of two corresponding blocks can be performed in memory. This is especially efficient for today's systems in which a large memory is usually provided. By sorting only at the block level, the sorting process is accomplished in one scan of the relation, which significantly reduces the number of disk accesses. Since one of the input relations needs to be scanned and order-preserving-hash-sorted, the cost of a semi-merge join is higher than that of a merge join, i.e., *cost(merge join)* < *cost(semi-merge join)*. A hash join on both unsorted relations, on the other hand, requires a scan and hash on both relations and then joins them. Hence, it is the most costly operation. Overall, the relationship of the costs of these three operations are *cost(merge join)* < *cost(semi-merge join)* < *cost(hash join)*.

As all results sent from PDBSs are sorted, there can be maximally $[N/2]$ joins reduced to a simple merge process (i.e., merge join) for a global query of N joins, if all join attributes are distinct attributes. If there are multiple joins on the same attribute, then this number can be even higher. Figure 19 shows an example of a query execution tree for Fig. 18, in which each circle is a join and each leaf node is a relation from a PDBS. The number in each circle is simply used to number the joins. In this query tree, seven of the joins (1 to 7) are reduced to a merge process. Join number 8 is a semi-merge join because R_8 can be sorted by its owner PDBS and the MDBS hash-sorts the result of join number 6. The remaining two joins (9 and 10) will be executed in the form of a hash join. Since a maximum number of merge joins is desired, a *bushy query tree* is in general better than a *left* or *right-deep tree* [12, 62]. In the performance study, we generate a set of queries to compare the performance of using different processing strategies. For the MMS, we always build for each query a bushy query tree such that a maximum number of merge joins can be employed.

G. Performance Study

1. Query Model

In order to compare the performance of the proposed strategies, we design a model that generates queries in a random manner. Each query in our model is a graph where a node is a relation and an edge is a join. Local operations are processed locally in the PDBSs. Hence, each node of a generated query graph represents a result relation output from a PDBS. The query can be either

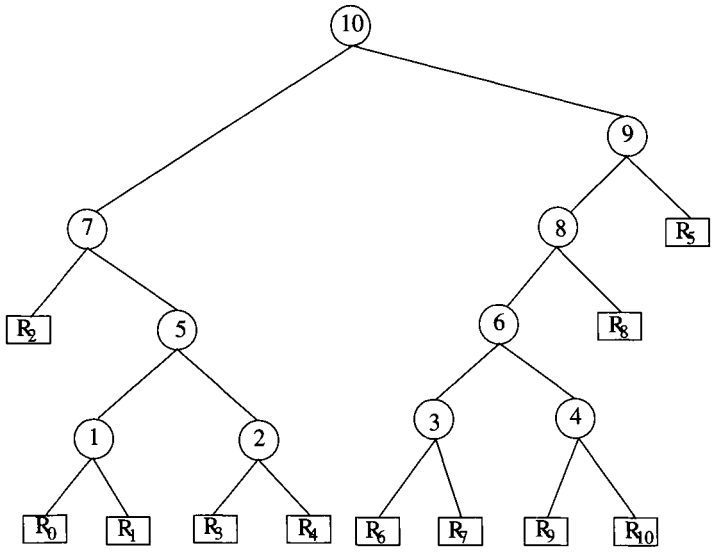


FIGURE 19 A query execution tree.

noncyclic or cyclic (i.e., loops may be formed in a query). A thousand queries will be generated for the experiments. There are two key tasks in generating such graphs. One is to model the joins on pairs of relations and another to model the join attributes. Let the number of relations involved in a query be N_r.

The Steps for Modeling Pairs of Relations

1. Create a node as the initial node.
2. Denote the initial node as R_i.
3. R_i may connect to $1 \sim 4$ new nodes according to the following probabilities.
   - 1 node: 1/2
   - 2 nodes: 1/4
   - 3 nodes: 1/8
   - 4 nodes: 1/8.

The nodes that R_i has been connected to are included in the generated number. That is, if p is the generated number (which may be 1, 2, 3, or 4 nodes) and if R_i has been connected to p' nodes before this step, then the actual number of new nodes added to connect to R_i is $p - p'$ if $p > p'$, or 0 if $p \leq p'$. However, the number of nodes of each query should not be more than N_r. When this step is done, mark R_i.

4. If the number of nodes of a query is equal to N_r, go to Step 5.

If all nodes of this join graph are marked but the number of nodes is smaller than N_r, select a new node from this join graph as the initial node and go to Step 2. Otherwise, select an arbitrary unmarked node and denote it as R_i. Go to Step 3.

5. For each pair of nonadjacent nodes in this join graph, an edge is generated to connect them based on a probability of 1/10.

From Step 1 to Step 4 this model creates tree queries. Step 5 is to generate cyclic queries based on a certain probability.

The Steps for Modeling Join Attributes

We say that two edges are adjacent if they have a same terminal node.

1. Assign different join attributes to all edges. Let all edges be unmarked.
2. Randomly select an edge as an initial edge and denote it as E_i.
3. The join attribute of an unmarked edge, say E_j, that is adjacent to E_i may be changed to the same join attribute as E_i based on a probability of 1/4. If $E_j's$ join attribute is changed to be the same as $E_i's$ join attribute, then mark E_j. After all unmarked edges adjacent to E_i have been gone through this process, marks E_i.
4. If all edges of a join graph are marked, Protocol stops. Otherwise, arbitrarily select an unmarked edge and let it be E_i. Go to Step 3.

Through these steps, we generate 1000 join queries (for a given N_r) to test the performance of the proposed strategies.

2. Parameter Setting

Parameters used in our study and their default values are summarized in Fig. 20. These values are standard values and have been used in many other works [37, 62]. Also similar to other research, we assume that the memory size is large enough to contain at least a pair of corresponding blocks of the two relations to be joined so as to simplify our simulation task. The execution cost is measured in the number of disk accesses. CPU cost is ignored since it is normally much smaller than disk I/O cost.

3. Results and Discussion

In our experiments, the size of each relation received by the MDBS is assumed to be 10^6. Hence, if JS is equal to 10^{-6}, the result of a join will have a same size as the source relations, i.e., $|R \bowtie S| = |R| = |S|$. In order to examine the effect of join selectivity factor on the performance of the strategies, we divide JS into three ranges.

1. 2×10^{-7}–8×10^{-7}: represents a **low** join selectivity factor. A join in this case will produce a result of only 20–80% of its input size.
2. $8 \times 10^{-7} - 1.2 \times 10^{-6}$: stands for a **medium** join selectivity factor. The result size of such a join falls between 80 and 120% of the input size.

Symbol	*Meaning*	*value*
N_r	The number of relations	$5 \sim 30$
$\|R\|$	The number of tuples of relation R	10^6
T_s	The size of each tuple	200 bytes
P_s	The size of each page	4 Kbytes
JS	The join selectivity factor	$2*10^{-7} \sim 2*10^{-6}$

FIGURE 20 Symbols and their meanings.

3. $1.2 \times 10^{-6} - 1.8 \times 10^{-6}$: stands for a **high** join selectivity factor. The result size of such a join is increased by more than 20% (i.e., 120%) of the input size up to 180%.

Low Join Selectivity

We first examine the result when JS is low. Figure 21 shows the performance ratio of all strategies under a low JS. The performance ratio is obtained by dividing the average execution cost of the generated queries of a proposed strategy by that of the FCFS strategy. "X" in the label of the vertical axis of the figure means one of the three strategies. Hence, the curve for FCFS is always 100%, and X's curve below FCFS's curve means that X is better than FCFS and above FCFS's curve means X is worse.

As we expected, FCFS is the worst because it does not utilize any knowledge about the join selectivity factors in scheduling the join operations. GRS is better than FCFS by only about 10%. MMS is the best and outperforms the other two consistently by about 50%. In other words, it consumes only about half of the computing power that is required by the other two strategies. This is mainly because of the use of reduction of a join to simply a merge process in that strategy. GRS is only slightly better than FCFS because the greedy scheduling scheme provides somewhat benefit. However, this advantage is overshadowed by the large cost for performing joins. The consistent benefit margin of MMS over the others indicates that it is independent of N_r, the number of relations.

Medium Join Selectivity

Figure 22 presents the result of medium join selectivity. In this experiment, we found that GRS performs much better than FCFS at large N_r. This is because the join selectivity factor is larger in this experiment. The greedy strategy becomes more effective when the number of relations to be joined increases. The MMS does not show such a phenomenon. This is because at least half of the joins of each query are simply a merge process. Some of the rest of the joins require only an (order-preserving) hash on one relation followed by a block-level merge of two relations. The number of "true" joins is not many in the

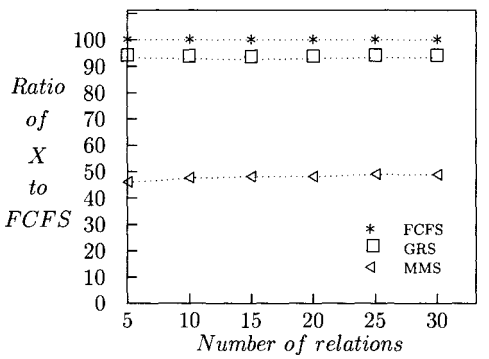


FIGURE 21 Comparison of strategies at low JS.

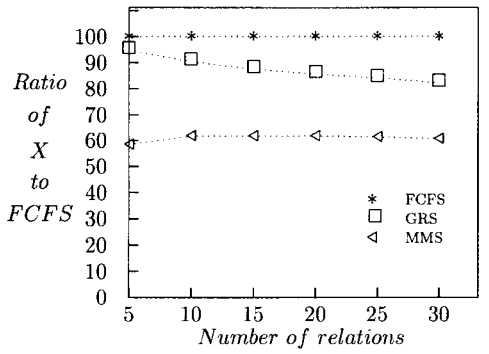


FIGURE 22 Comparison of strategies at medium JS.

range of the given number of relations in this experiment. Its saving will become more apparent when the number of relations (N_r) increases outside the range in this experiment.

High Join Selectivity

Figure 23 gives the result of high JS. In this case, both GRS and MMS outperform FCFS by a great margin, especially when N_r is large. This is because the JS is high such that the output size keeps increasing during the joins. Hence, if the number of joins increases, the benefit over FCFS increases too, enlarging the difference between FCFS and the other two strategies. Also we note that the performance of GRS approaches that of MMS at large N_r. This indicates that GRS is actually a good choice too under the condition that JS is high because this algorithm is simple, easy to implement, and does not require the PDBSs to sort their output relations. The only limitation is that it is confined to a high JS.

In summary, the MMS strategy is the best and outperforms the other two by a significant margin in all circumstances. GRS can be a second choice, in an environment where implementation complexity is a consideration and the join selectivity factor is known to be large. We have also conducted other sets of experiments by varying different parameters. Because of size limitation, we will present them in our future report.

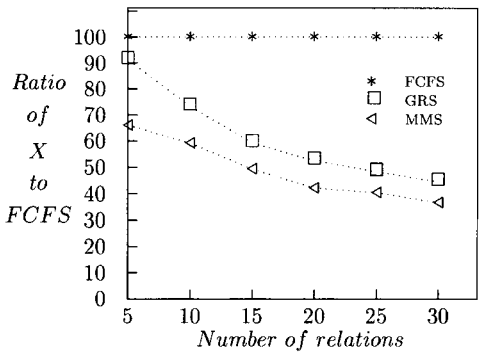


FIGURE 23 Comparison of strategies at high JS.

V. CONCLUSIONS

In a multidatabases environment, query processing is a time-consuming task as multiple levels of conversions (of data as well as the query) are involved. Query optimization in such an environment becomes especially involved. This issue, however, has not attracted much attention from researchers in the past.

In this chapter, we first classify the possible approaches to optimization from different levels. Among these approaching levels, our focus is solely on the algebra level and the execution strategy level. The reasons are that they are the key and necessary tasks of multidatabase query optimization, and optimization at these two levels is more practical and implementable.

For optimization at the algebra level, the task is to first find a uniform representation of conflicting relation schemas for multidatabase query translation. Although there has been plenty of research discussing the semantic conformation issues as cited earlier, there is still a lack of a thorough investigation on the representation and manipulation of data of conflicting schemas. The hyperrelational approach presented here provides support for dealing with these problems. Key concepts discussed for this algebra are as follows:

1. The notion of the least upper bound for a set of semantically equivalent relations is defined. Based on this definition, a hyperrelation can be found for each set of such relations. Instead of being a record keeping attribute values of an entity, a tuple in a hyperrelation keeps the schema information of a local relation. We presented in detail how schemas of each type of conflict are mapped to their corresponding hyperrelation.
2. Extending the notion of relational algebra, we developed a hyperrelational algebra that is performed on hyperrelations. Provided with algebraic transformation rules, the hyperrelational algebra serves as a sound basis for multidatabase query translation and optimization.

Having a clear and succinct tabular representation of conflicting local schemas, we believe that the SQL can be used as a multidatabase DDL/DML with minimum changes (to the SQL), because the tabular structure of a hyperrelation is not much different than that of a relation. This allows a global user to handle a multidatabase system easily, and it also reduces great effort in implementing a multidatabase system. View mechanism, a direct application of the query language, can therefore be easily dealt with in the hyperrelational approach.

For optimization at the execution strategy level, we discussed that conflict resolution on values of relations from different databases is often required in performing intersite database operations because of the heterogeneity of data between autonomous databases. Previous researches ignored this heterogeneity in their query optimization algorithms. In this chapter, we discussed this problem and argued that the MDBS, rather than an arbitrary PDBS, must perform intersite operations because only the MDBS has the required conflict resolution information and mechanisms. The transformed (and optimized) algebraic expression of a global query (obtained from the previous level of optimization) is used to find an optimal execution strategy. As MDBS is the site for executing intersite operations, it can easily become a bottleneck while serving for multidatabase users/applications. We presented algorithms that minimize the

consumption of system resources such that the MDBS bottleneck problem can be alleviated. The advantages of our methods are (1) the knowledge of cost models of the PDBSs is no needed such that there is not need to regenerate all PDBSs' cost models in the MDBS (this is however required in all previous works), (2) our methods reduce the load of the MDBS and even distribute part of the task (sort) to the PDBSs, and (3) our methods are much simpler and easier to implement than those complex cost model regeneration methods proposed before.

Overall, query optimization in a multidatabase system can be achieved from multiple levels. Optimization techniques at these levels can be combined into a complete query compilation/transformation/optimization framework and implemented as modules in a multidatabase query optimizer. This integration framework is especially crucial for new applications such as data warehouses, web-based data sources, and mobile information systems, because these applications involve a large number of information sources. We expect that the presented techniques can soon be realized in the next generation information systems.

REFERENCES

1. A Special Issue on Heterogeneous Database (March, S. Ed.), *ACM Comput. Surveys* **22**(3): 1990.
2. Agarwal, S., Keller, A. M., Wiederhold, G., and Saraswat, K. Flexible relation: An approach for integrating data from multiple, possibly inconsistent databases. In *International Conference on Data Engineering*, March 1995.
3. Ahmed, R., Smedt, P. D., Du, W., Kent, W., Ketbchi, M. A., Litwin, W. A., Rafii, A., and Shan, M. C. The Pegasus heterogeneous multidatabase system. *IEEE Computer* **24**(12): 19–27, 1991.
4. Batini, C., and Lenzerini, M. A methodology for data schema integration in the entity relationship model. *IEEE Trans. Software Engrg.* **10**(6): 650–664, 1984.
5. Batini, C., Lenzerini, M., and Navathe, S. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surveys* **18**(4): 1986.
6. Bregolin, M. Implementation of multidatabase SQL, technical report. University of Houston, May 1993.
7. Breitbart, R., Olson, P. L., and Thompson, G. R. Database integration in a distributed heterogeneous database system. In *Proc. IEEE Data Engineering Conference*, pp. 301–310, 1986.
8. Bright, M. W., Hurson, A. R., and Pakzad, S., Automated resolution of semantic heterogeneity in multidatabases. *ACM Trans. Database Systems* **19**(2):212–253, 1994.
9. Chatterjee, A., and Segev, A. Data manipulation in heterogeneous databases. *SIGMOD Record* **20**(4):1991.
10. Chartrand, G., and Oellermann, O. R. *Applied and Algorithmic Graph Theory.* McGraw-Hill, New York, 1993.
11. Chen, A. L. P. Outerjoin optimization in multidatabase systems. In *International Symposium on Databases in Parallel and Distributed Systems*, 1990.
12. Chen, M.-S., Yu, P. S., and Wu, K.-L. Optimization of parallel execution for multi-join queries. *IEEE Trans. Knowledge Data Engrg.* **8**(3):416–428, 1996.
13. A Special Issue on Heterogeneous Distributed Database Systems (Ram, S. Ed.), *IEEE Computer* **24**(12):1991.
14. Czejdo, B., Rusinkiewicz, M., and Embley, D. W. An approach to schema integration and query formulation in federated database systems. In *Proceedings of Third International Conference on Data Engineering*, 477–484, 1987.
16. Czejdo, B. and Taylor, M. Integration of database systems using an object-oriented approach. In *International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, 1991, pp. 30–37.

17. Dayal, U. Query processing in a multidatabase system. In *Query Processing in Database Systems* (Kim, W., Reiner, D. S., and Batory, D. S. Eds.), pp. 81–108. Springer-Verlag, Berlin, 1984.
18. Dayal, U., and Hwang, H. View definition and generalization for database integration in multidatabase: A system for heterogeneous distributed database. *IEEE Trans. Software Engrg.* 10(6):628–644, 1984.
19. DeMichiel, L. G. *Performing Operations over Mismatched Domain.* In *IEEE Proceedings of the Fifth International Conference on Data Engineering*, 1989.
20. DeMichiel, L. G. Resolving database incompatibility. *IEEE Trans. Knowledge Data Engrg.* 1(4):1989.
21. DeMichiel, L. G. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Trans. Knowledge Data Engrg.* 1(4):485–493, 1989.
22. DeWitt, D. J. *et al.* Implementation techniques for main memory database systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, June 1984, pp. 1–8.
23. DeWitt, D. J. *et al.* The gamma database machine project. *IEEE Trans. Knowledge Data Engrg.* 2(1):44–62, 1990.
24. Du, W., Krishnamurthy, R., and Shan, M. Query optimization in heterogeneous DBMS. In *Proc. of the 18th VLDB Conference*, pp. 277–291, 1992.
25. Du, W., Shan, M., and Dayal, U. Reducing multidatabase query response time by tree balancing. In *Proc. of the ACM SIGMOD Conference*, pp. 293–303, 1995.
26. Dwyer, P. A., and Larson, J. A. Some experiences with a distributed database tested system. *Proc. IEEE* 75(5):1987.
27. Egyhazy, C. J., Triantis, K. P., and Bhasker, B. A query processing algorithm for a system of heterogeneous distributed databases. *Distrib. Parallel Databases.* 4(1):49–79, 1996.
28. Elmasri, R., and Navathe, S. B. *Fundamentals of Database Systems.* Benjamin/Cummings, Reduced City, CA, 1994.
29. Evrendilek, C., Dogac, A., Nural, S., and Ozcan, F. Multidatabase query optimization. *Distrib. Parallel Databases.* 5(1):77–114, 1997.
30. Gangopadhyay, D., and Barsalou, T. On the semantic equivalence of heterogeneous representations in multimodel multidatabase systems. *SIGMOD Record* 20(4):1991.
31. Geller, J., Perl, Y., and Neuhold, E. J. Structural schema integration in heterogeneous multidatabase systems using the dual model. In *Proc. the 1st RIDE-IMS Workshop*, Kyoto, Japan, 1991.
32. Graefe, G. Query evaluation techniques for large databases. *ACM Comput. Surveys* 25(2):1993.
33. Grant, J., Litwin, W., Roussopoulos, N., and Sellis, T. Query languages for relational multidatabases. *VLDB Journal* 2(2):1993.
34. Heimbigner, D., and McLeod, D. A federated architecture for information management. *ACM Trans. Office Inform. Systems* 253–278, 1985.
35. Hsiao, D. K., and Kamel, M. N. Heterogeneous database: Proliferations, issues, and solutions. *IEEE Trans. Knowledge Data Engrg.* 1(1):1989.
36. Hua, K. A., Lee, C., and Young, H. Data partitioning for multicomputer database systems: A cell-based approach. *Inform. Systems* 18(5):329–342, 1993.
37. Hua, K. A., Lee, C., and Hua, C. M. Dynamic load balancing in multicomputer database system using partition tuning. *IEEE Trans. Knowledge Data Engrg.* 7(6):968–983, 1995.
38. Hwang, H. Y., Dayal, U., and Gouda, M. Using semiouterjoin to process queries in multidatabase systems. In *ACM SIGMOD Conference on Management of Data*, 1984.
39. Kambayashi, Y., Rusinkiewicz, M., and Sheth, A., (Eds.) *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, 1991.
40. Schek, H-J., Sheth, A., and Czejdo, B., (Eds.). *Proceedings of the International Workshop on Interoperability in Multidatabase Systems*, Vienna, Austria, 1993.
41. Kamel, M. N., and Kamel, N. N. Federated database management system: Requirements, issues and solutions. *Comput. Commun.* 15(4):1992.
42. Kent, W. Solving domain mismatch and schema mismatch problems with an object-oriented database programming language. In *Proceedings of the International Conference on Very Large Data Base*, 1991.
43. Kim, W., and Seo, J. Classify schematic and data heterogeneity in multidatabase systems. *IEEE Computer* 24(12):12–17, 1991.

44. Krishnamurthy, R., Litwin, W., and Kent, W. Language features for interoperability of databases with schematic discrepancies. In *Proc. of the SIGMOD Conference*, pp. 40–49, 1991.
45. Lee, C., and Tsai, C. L. Strategies for selection from heterogeneous databases. In *Proceedings of the Third International Symposium on Database Systems for Advanced Applications*, Taejon, Korea, April 1993.
46. Lee, C., Chen, C.-J., and Lu, H. An aspect of query optimization in multidatabase systems. *ACM SIGMOD Record*, 24(3):28–33, 1995.
47. Lee, C., and Chang, Z.-A. Utilizing page-level join index for optimization in parallel join execution. *IEEE Trans. Knowledge Data Engrg.* 7(6):1995.
48. Lee, C., and Wu, M.-C. A hyperrelational approach to integration and manipulation of data in multidatabase systems. *Int. J. Intell. Cooperative Inform. Systems* 5(4):395–429, 1996.
49. Lee, C., and Chen, J. R. Reduction of access sites for query optimization in multidatabase systems. *IEEE Trans. Knowledge Data Engrg.* 9(6):941–955, 1997.
50. Lee, C., Ke, C.-H., and Chen, Y.-H. Minimization of query execution in multidatabase systems *Int. J. Cooperative Inform. Systems*, 2002.
51. Lim, E.-P., Srivstav, J., Prabhakar, S., and Richardson, J. Entity identification in database integration. In *Proc. of the International Conference on Data Engineering*, pp. 294–301, 1993.
52. Litwin, W. MALPHA: A relational multidatabase manipulation language. In *Proceedings of the International Conference on Data Engineering*, April 1984.
53. Litwin, W., and Abdellatif, A. Multidatabase interoperability. *IEEE Computer* 1986.
54. Litwin, W., and Vigier, P. Dynamic attributes on the multidatabase system MRDSM. In *Proc. of the International Conference on Data Engineering*, 1986.
56. Litwin, W., and Abdellatif, A. An overview of the multi-database manipulation language mDSL. *Proc. IEEE* 75(5):1987.
57. Litwin, W., Mark, Leo., and Roussopoulos, N. Interoperability of multiple autonomous databases. *ACM Comput. Surveys*, 22(3):267–293, 1990.
58. Litwin, W. MSQL: A multidatabase language. *Inform. Sci.* 1990.
59. Litwin, W., Ketabchi, M., and Krishnamurthy, R. First order normal form for relational databases and multidatabases. *ACM SIGMOD Record* 20(4):1991.
60. Lu, H., Shan, M.-C., and Tan, K.-L. Optimization of multi-way join queries. In *Proceedings of the 17th International Conference on VLDB*, Barcelona, September pp. 549–560, 1991.
61. Lu, H., Ooi, B.-C., and Goh, C.-H. Multidatabase query optimization: Issues and solutions. In *Proc. of the Workshop on Interoperability in Multidatabase Systems*, pp. 137–143, 1993.
62. Lu, H., Ooi, B.-C., and Tan, K.-L. (Eds.). *Query Processing in Parallel Relational Database Systems*. IEEE Computer Society Press, Los Alamitos, CA, 1994.
63. Meng, W., Yu, C., Guh, K. C., and Dao, S. Processing multidatabase queries using the fragment and replicate strategy, Technical Report CS-TR-93-16. Department of Computer Science, SUNY at Binghamton, 1993.
64. Meng, W., and Yu, C. Query processing in multidatabase systems. In *Modern Database Systems: the Object Model, Interoperability, and Beyond* (Kim, W. Ed.), Chap. 27. Addison-Wesley, Reading, MA, 1995.
65. Missier, P., and Rusinkiewicz, M. Extending a multidatabase manipulation language to resolve schema and data conflicts, Technical Repost UH-CS-93-10. University of Houston, November 1993.
66. Motro, A. Superviews: Virtual integration of multiple databases. *IEEE Trans. Software Engrg.* 13(7):785–798, 1987.
67. Navathe, S., Elmasri, R., and Larson, J. Integrating user views in database design. *IEEE Computer* 1986.
68. Ngu, A. H. H., Yan, L. L., and Wong, L. S. Heterogeneous query optimization using maximal sub-queries. In *International Conference on Database Systems For Advanced Applications (DASFAA'93)*, Taejon, Korea, April pp. 413–420, 1993.
69. A Special Issue on Semantic Issues in Multidatabase Systems, *ACM SIGMOD Record* 20(4):1991.
70. Rusinkiewicz, M., Elmasri, R., Czejdo, B., Georgakopoulos, D., Karabatis, G., Jamoussi, A., Loa, K., and Li, Y. Query processing in OMNIBASE—A loosely coupled multi-database system, Technical Report UH-CS-88-05. University of Houston, February 1988.

71. Salza, S., Barone, G., and Morzy, T. Distributed query optimization in loosely coupled multidatabase systems. In *International Conference on Database Theory*, 1994.
72. Savasere, A., Sheth, A., Gala, S. K., Navathe, S. B., and Marcus, H. On applying classification to schema integration. In *Proc. First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 1991.
73. Sciore, E., Siegel, M., and Rosenthal, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Trans. Database Systems* **19**(2):254–290, 1994.
74. Sheth, A. P., and Larson, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surveys* **22**(3):1990.
75. Siegel, M., and Madnick, S. E. A metadata approach to resolving semantic conflicts. In *Proc. of the 17th International Conf. on VLDB*, September 1991.
76. Spaccapietra, S., and Parent, C. Conflicts and correspondence assertions in interoperable databases. *SIGMOD Record* **20**(4):1991.
77. Spaccapietra, S., Parent, C., and Dupont, Y. Model independent assertions for integration of heterogeneous schemas. *VLDB Journal* 1, 1992.
78. Suardi, L., Rusinkiewicz, M., and Litwin, W. Execution of extended multidatabase SQL. In *IEEE Proceedings of the International Conference on Data Engineering*, 1993.
79. Templeton, M., Brill, D., Dao, S. K., Lund, E., Ward, P., Chen, A. L. P., and Macgregor, R. Mermaid—A front-end to distributed heterogeneous databases. *Proc. IEEE* 75(5):1987.
80. Tresch, M., and Scholl, M. H. Schema transformation processors for federated object-bases. In *DASFAA* 1993.
81. Wang, Y. R., and Madnick, S. E. The inter-database instance identification problem in integrating autonomous systems. In *Proc. of the International Conference on Data Engineering*, 1989.
82. Whang, W. K., Chakravarthy, S., and Navathe, S. B. Heterogeneous database: Inferring relationships for merging component schemas, and a query language, Technical Report UF-CIS-TR-92-048. Department of Computer and Information Sciences, University of Florida, 1992.
83. Yang, J., and Papazoglou, M. P. Determining schema interdependencies in object-oriented multidatabase systems. In *International DASFAA Symposium*, Korea, 1993.
84. Yu, C., Sun, W., Dao, S., and Keirsey, D. Determining relationships among attributes for interoperability of multi-database systems. In *Proc. of the Workshop on Interoperability in Multidatabase Systems*, 1991.
85. Leon Zhao, J., Segev, A., and Chatterjee, A. A univerisal relation approach to federated database management. In *International Conference on Data Engineering*, March, 1995.
86. Zhu, Q., and Larson, P. A. A query sampling method for estimating local cost parameters in a multidatabase system. In *Proc. of the Int'l Conf. on Data Engineering*, pp. 144–153, 1994.

5

DEVELOPMENT OF MULTILEVEL SECURE DATABASE SYSTEMS

ELISA BERTINO

Dipartimento di Scienze dell'Informazione, Università di Milano, 20135 Milano, Italy

ELENA FERRARI

Dipartimento di Chimica, Fisica e Matematica, Università dell'Insubria-Como, Italy

I. INTRODUCTION

Data protection from unauthorized accesses is becoming more and more crucial as an increasing number of organizations entrust their data to database systems [10,33].

Database and Data Communication Network Systems, Vol. 1

An important functionality that every *database management system* (DBMS) must support is the ability to protect data and system resources from intrusions, modifications, theft, and unauthorized disclosures. Since data in a database are related by semantic relationships, a damage in a database environment does not only affect a single user or application, but the entire information system.

Security breaches are typically categorized into the following categories: *unauthorized data observation, incorrect data modification,* and *data unavailability.* Unauthorized data observation results in the disclosure of information to subjects not entitled to gain access to such information. All organizations we may think of, ranging from commercial organizations to social or military organizations, may suffer heavy losses from both financial and human point of views upon unauthorized data observation. Incorrect modifications of data, either intentional or unintentional, result in an inconsistent database state. As a result, the database is no longer correct. Any use of incorrect data may again result in heavy losses for the organization. When data are unavailable, information that is crucial for the proper functioning of the organization may not be readily accessible when needed. Therefore, a complete solution to the data security problem entails addressing three main issues: *secrecy* or *confidentiality, integrity,* and *availability.*

Ensuring secrecy means preventing improper disclosure of information. Ensuring integrity means protecting data from unauthorized or improper modifications or deletions. Finally, ensuring availability means ensuring prevention and recovery from hardware and software errors and from malicious data denials making the database system not available.

The importance assigned to the above aspects greatly depends on the considered environment. For example, secrecy is the most relevant aspect of military environments, whereas in commercial environments most attention is devoted to integrity.

In many environments, such as public institutions, secrecy and integrity are often needed in combination. This is the case, for instance, of hospitals, airline companies or credit institutions, in which, besides privacy constraints, also data correctness is vital.

Data protection is ensured by different components of a DBMS. In particular, the *access control mechanism* ensures data secrecy. Whenever a subject tries to access an object, the access control mechanism checks the right of the subject against a set of authorizations, stated usually by some security administrator. An *authorization* states which subject can perform which action on which object. Authorizations are granted according to the security policies of the organization.

Data integrity is jointly ensured by the access control mechanism and by semantic integrity constraints. Whenever a subject tries to modify some data, the access control mechanism verifies that the subject has the right to modify the data, whereas the semantic integrity subsystem verifies that the updated data are semantically correct. Finally, the recovery subsystem and the concurrency control mechanism ensure that data are available and correct despite hardware and software failures and despite data accesses from concurrent application programs.

Authorization mechanisms provided as part of commercial DBMSs and research prototypes can be classified into two main categories: *discretionary* and *mandatory* authorization mechanisms. Discretionary access control mechanisms are characterized by a high degree of flexibility that makes them suitable for a large variety of application domains. They govern the access of subjects to data on the basis of subjects' identity and authorization rules. Under discretionary policy whenever a subject wishes to authorize another subject to exercise a privilege on a given object, it must insert into the system an authorization rule explicitly stating this right. When an access request is submitted, the access control mechanism checks if there exists a rule authorizing such access. In this case, the access is authorized; otherwise, it is denied. Such mechanisms are discretionary in that they allow subjects to grant other subjects authorization to access the data at their discretion. The main drawback of discretionary access control models is their vulnerability to malicious attacks, such as Trojan Horses, embedded in application programs. The reason is that discretionary authorization models do not impose any control on how information is propagated and used once it has been accessed by properly authorized subjects. The following example clarifies the discussion.

EXAMPLE 1. Consider a relational DBMS and suppose that *John* creates a table *Salary* containing information on the salary of each employee in his department. Suppose moreover that there exists an employee, say *Matt*, who is not authorized to know how much the employees of the department are paid. Suppose that *Matt* wants to get the sensitive information on the employee salary. *Matt* creates a table *Stolen* and gives *John* the write privilege on it. Then *Matt* modifies a worksheet application to include two hidden operations: a read operation on table *Salary* and a write operation on table *Stolen* (see Fig. 1a). Then he gives this application to *John*. When *John* executes the application, all the accesses performed by the application are checked against *John*'s authorizations. As a result, the sensitive information on the employees salaries are transferred into table *Stolen* and thus they are made accessible to *Matt* (see Fig. 1b).

By contrast, mandatory policies ensure a high degree of protection in that they prevent any illegal flow of information. They are therefore suitable for contexts, like the military one, which require a high degree of protection. Mandatory policies require a strict classification of subjects and objects into security levels. Access control is based on a set of axioms stating the relations that must occur between the security level of an object and that of a subject because the subject is granted access to the object. This type of security has also been referred to as *multilevel* security.

In this chapter, we focus on multilevel security. In particular, we deal with multilevel access control mechanisms and on secure concurrency control for DBMSs adopting a mandatory access control policy. We refer the reader to [4] for details on discretionary access control mechanisms. Moreover, a detailed description of transaction models and recovery mechanisms can be found in [5], whereas details on semantic integrity control can be found in any database textbook (see for instance [41]).

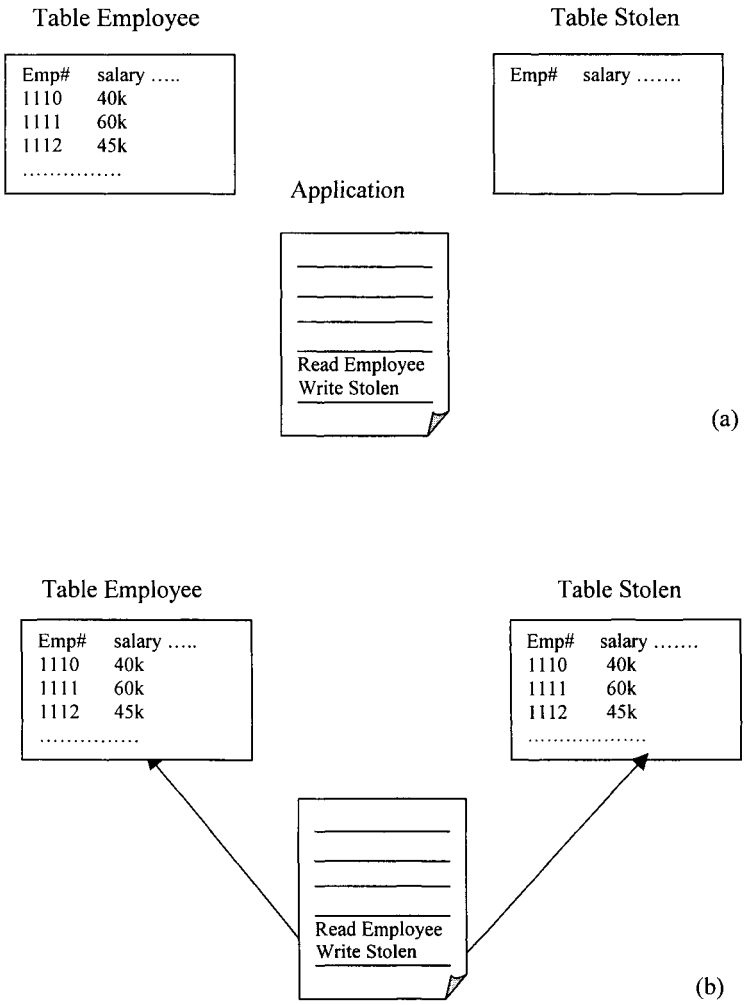


FIGURE 1 An example of Trojan Horse.

The remainder of this chapter is organized as follows. Section II surveys the basic concepts of access control. Section III illustrates mandatory access control policies and related access control models. Sections IV and V deal with multilevel access control in relational and object-oriented DBMSs, respectively, whereas Section VI deals with secure concurrency control. Finally, Section VII concludes the chapter and discusses future research directions.

II. ACCESS CONTROL: BASIC CONCEPTS

Access control regulates the *privileges subjects* can exercise on the *objects* in the system. The following is a discussion on objects, subjects, and privileges.

A. Authorization Objects

Authorization objects are the passive components of a system to which protection from unauthorized accesses should be given. Objects to be considered depend on the underlying data model. For instance, files and directories are examples of objects of an operating system, whereas if we consider a relational DBMS, resources to be protected are relations, views, and attributes. With respect to the object dimension we can classify access control mechanisms according to the granularity of access control, that is, according to whether it is possible to authorize a subject to access only selected components within an object.

Access control models can be further classified according to whether the set of objects to be protected represents a flat domain or whether the objects are organized into a hierarchy. In the latter case, the semantics assigned to the hierarchy greatly depends on the object nature. For instance, consider an object-oriented context. If objects to be protected are classes, the hierarchy represents the inheritance relations among classes. If objects represent class instances, the hierarchy reflects the way objects are organized in terms of other objects.

B. Authorization Subjects

Authorization subjects are the entities in the system to which authorizations are granted. Subjects can be classified into the following categories:

- **Users**, that is, single individuals connecting to the system.
- **Groups**, that is, sets of users.
- **Roles**, that is, named collection of privileges needed to perform specific activities within the system.
- **Processes**, that is, programs executed on behalf of users.

Note that the above categories are not mutually exclusive. For instance, a model can support both roles and groups, or both users and processes as authorization subjects.

Often, both roles and groups are hierarchically organized. The hierarchy imposed on groups usually reflects the membership of a group to another group. A nested group inherits the privileges of the groups preceding it in the nesting. By contrast, the role hierarchy usually reflects the relative position of roles within an organization. The higher is the level of a role in the hierarchy, the higher is its position in the organization. Thus, a role has all the privileges of the roles in a lower position in the hierarchy.

Processes need system resources to carry on their activities. Generally, processes refer to memory addresses, use the CPU, call other processes, and operate on data. All these resources must be protected from unauthorized accesses. Usually, a process is granted accesses only to essential resources, that is, those necessary to the completion of its tasks. This limits possible damage deriving from faults of the protection mechanism.

As far as users are concerned, sometimes it would be useful to specify access policies based on user qualifications and characteristics, rather than user identity (for example, a user can be given access to an R rated video, only if he/she is older than 18 years). This is the case, for instance, of digital library

environments. In access control models supporting these possibilities [1,39] users must provide information, typically about themselves, allowing the access control mechanism to decide whether the access must be authorized.

As a final remark, note that authorization subjects can be further classified into *active* and *passive* subjects. Active subjects can directly require accesses to the objects, whereas passive subjects can only appear as subjects in an authorization but they cannot perform accesses to the objects. Groups are examples of passive subjects, whereas users are examples of active subjects.

C. Authorization Privileges

Authorization privileges state the types of operations that a subject can exercise on the objects in the system. The set of privileges a subject can exercise depends on the resources to be protected. For instance, read, write, and execute privileges are typical of an operating system environment, whereas in a relational DBMS typical privileges are select, insert, update, and delete. Moreover, new environments, such as the digital library environment, are characterized by new access modes, such as the usage or copying of access rights.

Often, privileges are hierarchically organized and the hierarchy represents a subsumption relation among privileges. Privileges toward the bottom of the hierarchy are subsumed by privileges toward the top (for instance, the write privilege is at a higher level in the hierarchy with respect to the read privilege, since write subsumes read operations). Thus, an authorization for a given privilege p implies an analogous authorization for all the privileges following p in the hierarchy.

III. MANDATORY ACCESS CONTROL

Mandatory (or multilevel) access control policies specify the access that subjects have to objects. According to mandatory policies both objects and subjects are given a classification level. Access control is then regulated by a set of axioms that establish the relations that must exist between the classification level of a subject s and an object o, to allow s to access o. In general, such relations depend on the considered privilege. Database systems that satisfy multilevel security properties are called *multilevel secure database management systems* (MLS/DBMSs) or *trusted database management systems* (TDBMSs). Many of the MLS/DBMSs have been designed based on the Bell and LaPadula policy [3] specified for operating systems. In the following subsections we first review the basic concepts of the Bell and LaPadula model [3], then we illustrate another milestone in multilevel access control, that is, the model proposed by Denning [13].

A. The Bell and LaPadula Model

In the Bell and LaPadula model, five different types of privileges are considered, with the following meaning:

1. read: a subject having the read privilege on an object can read the information in the object without modifying its content;

2. **append**: the append privilege allows a subject to modify the content of an object, but it does not allow the subject to extract information from the object (i.e., it does not subsume the read privilege);
3. **write**: the write privilege subsumes both the read and the append privilege in that it allows a subject to read information in an object and to modify the object content;
4. **execute**: this privilege allows a subject to execute an object. It applies to objects that contain a program or a routine. The execute privilege does not imply read, append, or write privileges;
5. **control**: it allows the delegation of privileges to other subjects. If a subject has the control privilege on an object, then he/she can extend to another subject the privileges he/she posses on the object.

In the Bell and LaPadula model, both subjects and objects are assigned an *access class*.

An access class consists of two components: a *security level* and a *set of categories*. The set of security levels forms a totally ordered lattice. The lattice usually considered consists of the following levels: Unclassified (U), Confidential (C), Secret (S), and TopSecret (TS), with Unclassified < Confidential < Secret < TopSecret. The security level associated with an object reflects the sensitivity of the information contained in the object; that is, it is a measure of the potential damage that could result from unauthorized disclosure of the information. By contrast, the security level associated with a subject (usually called *clearance*) is a measure of the subject's trustworthiness not to disclose sensitive information to other subjects not cleared to access it. The set of categories is an unordered set. Categories depend on the considered domain. Examples of categories in a military context are NATO, Nuclear, Navy, and Army.

Access classes are partially ordered according to a *dominance relation* (written $\geq$) defined as follows: given two access classes $ac_1 = (L_1, C_1)$ and $ac_2 = (L_2, C_2)$, we say that ac_1 *dominates* ac_2 (written $ac_1 \geq ac_2$) if and only if both the following conditions hold:

1. The security level of ac_1 is greater than or equal to the security level of $ac_2 : L_1 \geq L_2$
2. The set of categories of ac_1 includes the set of categories of $ac_2 : C_2 \subseteq C_1$.

If both the above conditions strictly hold (that is, $L_1 > L_2$ and $C_2 \subset C_1$), we say that the access class ac_1 *strictly dominates* the access class ac_2. Finally, if neither $ac_1 \geq ac_2$ nor $ac_2 \geq ac_1$ holds, ac_1 and ac_2 are said to be *incomparable*.

EXAMPLE 2. Consider the following access classes:

$$ac_1 = (\text{TS}, \{\text{Nuclear}, \text{Army}\})$$

$$ac_2 = (\text{TS}, \{\text{Nuclear}\})$$

$$ac_3 = (\text{C}, \{\text{Army}\})$$

$ac_1 \geq ac_2$, since {Nuclear} $\subseteq$ {Nuclear,Army}; $ac_1 > ac_3$, since TS > C and {Army} $\subset$ {Nuclear,Army}. Finally, ac_2 and ac_3 are incomparable with respect to the dominance relation. Indeed, ac_2 does not dominate ac_3, since {Army} is not contained in {Nuclear}, and ac_3 does not dominate ac_2 since TS > C.

In the Bell and LaPadula model, the state of the system is described by a 4-tuple (A,M,L,G), where:

- A is the set of current accesses. Elements of A are triples (s,o,p). If (s,o,p) belongs to A, it means that subject s is currently exercising privilege p on object o;
- M is the access matrix; that is, it is a matrix containing for each object and subject in the system the privileges the subject can exercise on the object;
- L is a function that given an object or a subject returns its access class;
- G is the current hierarchy of objects; that is, it is a hierarchy denoting the objects that are currently accessible in the system.

Modifications to the state of the system are caused by *requests*. Requests can be of four different types:

1. a request by a subject to exercise a particular privilege on an object;
2. a request by a subject to grant another subject an access authorization;
3. a request by a subject to create an object;
4. a request by a subject to delete an object.

In the following, we consider only access requests, that is, request of the first type (these are requests for the read, append, write, and execute privilege). We refer the reader to [3] for the other types of requests.

The answer to a given request is called *decision*. Given a state of the system and a request, the result of the request is decided based on a set of *security axioms*. A system is *safe* if the requests are processed based on these axioms.

As far as access requests are concerned, the Bell and LaPadula model is governed by the following two axioms:

Simple Security Property. A state (A,M,L,G) satisfies the simple security property if for each element $a = (s,o,p)$ in A one of the following conditions is verified:

1. p = execute or p = append;
2. p = read or p = write and $L(s) \geq L(o)$.

***-Property (Read Star Property).** A state (A,M,L,G) satisfies the *-property if for each element $a = (s,o,p)$ in A one of the following conditions is verified:

1. p = execute;
2. p = append and $L(s) \leq L(o)$;
3. p = write and $L(s) = L(o)$.

For example, a subject with access class (C,{Army}) cannot read objects with access classes (C,{Navy,NATO}) or (U,{Nato}), since these read operations would violate the simple security property. Moreover, a subject with access class (C,{Army,Nuclear}) cannot write an object with access class (U,{Army}), since this write operation would violate the *-property.

For the sake of simplicity, in the remainder of this chapter we assume an empty set of categories. This allows us to consider only security levels.

B. Denning Model

The model by Denning [13] is an extension of the Bell and LaPadula model presented in the previous section. The main difference between the two models is that in [13] the concept of *security class* is introduced, which unifies the concepts of category and security level of the Bell and LaPadula model. The Denning model consists of five basic components:

1. a set of objects O, representing the resources to be protected;
2. a set of processes P, which are the active entities of the system, requiring accesses to the objects;
3. a set of security classes SC;
4. a flow relation, denoted as $\rightarrow$;
5. a binary operator Θ: $SC \times SC \rightarrow SC$.

The operator Θ receives as input two security classes sc_1 and sc_2, and returns the security class that should be assigned to the result of any operation that combines information contained into objects whose security class is sc_1 and sc_2, respectively. The flow relation specifies the legal information flow. For example, $sc_1 \rightarrow sc_2$ specifies that a flow of information may take place from objects with security class sc_1 to objects with security class sc_2. Denning proved that the triple $(SC, \rightarrow, \Theta)$ is a finite lattice under the following hypothesis:

1. SC is a finite set;
2. $\rightarrow$ is a partial order relation over SC;
3. Θ is a total function with a least upper bound.

EXAMPLE 3. Consider a database with the following characteristics:

- the database contains three different types of records: medical records (m); financial records (f), and criminal records (c);
- the set of security classes consists of all the subsets of {m,f,c};
- flow of information from a class sc_1 to a class sc_2 is permitted if and only if $sc_1 \subseteq sc_2$;
- Θ is the union operator.

In this case, the triple $(SC, \rightarrow, \Theta)$ is a finite lattice, since all the hypotheses stated above are verified (see Fig. 2).

IV. MULTILEVEL SECURITY IN RELATIONAL DBMSs

This section discusses multilevel security models for relational database systems. Object DBMSs will be considered in the next section. Note that there are several other significant issues concerning multilevel security for database systems. These include inference problems, transaction management, and multilevel

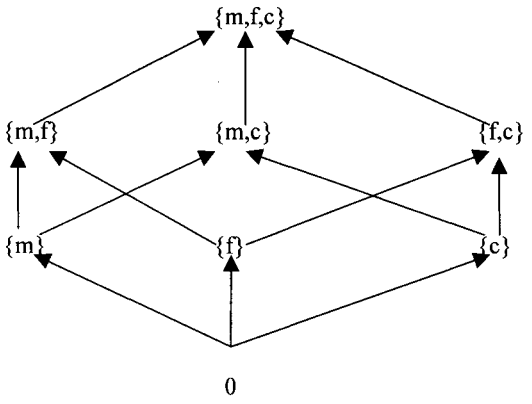


FIGURE 2 Lattice for Example 3.

security for distributed, heterogeneous, and federated database systems. We do not discuss such issues in this chapter, except for concurrency control protocols which are described in Section VI. For details on inference problems we refer the reader to [28]; for information on secure distributed and heterogeneous databases as well as secure federated databases we refer the reader to [38]. In the following, we first review the basic concepts of the multilevel relational data model, we then present some of the most relevant access control models for multilevel relational DBMSs.

A. Multilevel Relational Data Model

In a multilevel database not all of the data are assigned the same security level. If such a database is based on the relational model, the objects of classification may be the entire database, relations, tuples, or attributes. Access to these objects is governed by the mandatory policy discussed in Section II. A multilevel DBMS should protect the multilevel database from unauthorized access or modification by subjects cleared to different security levels. A multilevel relational database consists of a set of relations. The corresponding model is called a *multilevel relational data model.*

A goal of a multilevel relational database designer is to represent multiple versions of the same entity, action, or event at different security levels without violating the integrity or security rules. One of the mechanisms being proposed to represent multiple versions of an entity at different security levels is *polyinstantiation*; that is, the same object can have different interpretations and values at different levels. For example, at the unclassified level an employee's salary may be 30 K and at the Secret level the salary may be 70 K. With multilevel relational models one can have both entries but with their security levels as an additional attribute. One of the main motivations toward handling polyinstantiation is to avoid what is called *covert channels*. For example, if there is an entry at the Secret level that John's salary is 70 K and if an Unclassified subject wants to enter that John's salary is 30 K, and if the update is not permitted there could be a signalling channel from a higher level

to a lower level. Over time this could become a covert channel. Polyinstantiation enables two tuples with the same primary key to exist in a relational database at different security levels. However, having two tuples with the same primary key violates the entity integrity property of the standard relational data model.

Several discussions and debates took place on polyinstantiation in the early 1990s. No consensus was reached. Some argued that polyinstantiation is necessary if we are to design multilevel database systems with higher levels of assurance (see, for example, [14]). Some argued that it is important to maintain the integrity of the database and that polyinstantiation violates the integrity (see, for example, [9]). Some used partial polyinstantiation together with security constraint enforcement in their design (see, for example, [32]). An interesting operational example showing the disastrous effects of polyinstantiation is given in the paper by Wiseman [40]. Even among those who support polyinstantiation, there has been much discussion on the correct update semantics to use. A logic for formalizing concepts in multilevel relations and which supports polyinstantiation is given in [36].

B. Sea View

SeaView is a multilevel relational data model, developed in the context of the SeaView project [14]. The SeaView project is a joint project by SRI International and Gemini Computers, Inc. The project also defined MSQL, an extension of SQL to handle multilevel data.

The SeaView security model consists of two components; the MAC (Mandatory Access Control) model and the TCB (Trusted Computing Base) model. The MAC model defines the mandatory security policy. Each subject is assigned a *readclass* and a *writeclass*. A subject can read an object if the subject's readclass dominates the access class of the object. A subject can write into an object if the object's access class dominates the writeclass of the subject.

The TCB model defines discretionary security and supporting policies for multilevel relations, views, and integrity constraints, among others. The data model on which SeaView is based is a multilevel relational data model. Multilevel relations are implemented as views over single level relations, that is, over relations having a single access class associated with them. Implementing multilevel relations as virtual relations (or views) allows subjects to issue insert, delete, and update requests on these views. Appropriate algorithms are then used to map updates on views onto updates on the base relations which are single level. An advantage of the SeaView approach is that the labels of the data elements need not be stored.

Each database operation is executed by a single level subject. When a subject at level L issues a request, a database system subject operating at level L will process the subject's request. This subject then cannot have read access to objects not classified at or below the level L.

Polyinstantiation is the mechanism introduced by SeaView to handle cover stories as well as signaling channels. For example, in a multilevel world, it is possible to have multiple views of the same entity at different security levels.

In the SeaView model, the two views may be represented, say, by two tuples with the same primary key, but at different security levels. The primary key constraint is not violated since in the multilevel relational data model proposed by SeaView a modified entity integrity property is defined. Additional integrity properties such as referential integrity property and polyinstantiation integrity property are also defined in the SeaView model.

C. LDV

In LDV [32], the relational query language SQL is enhanced with constructs for formulating security assertions. These security assertions serve to imply sensitivity labels for all atomic values, contexts, and aggregations in a database. The labeled data are partitioned across security levels, assigned to containers with dominating security markings or levels, and may only flow upward in level unless authorized otherwise.

LDV is based on the LOCK security policy, which consists of both a discretionary and a mandatory security policy. The discretionary security policy regulates the sharing of objects among the various subjects. The mandatory security policy controls the potential interferences among subjects and consists of a mandatory access control policy and a type enforcement policy. The mandatory access control policy is based on the Bell and LaPadula policy. The type enforcement policy restricts accesses of subjects to objects based on the domain of the subject and the type of the object.

Moreover, LDV addresses the problem of updating and querying a multilevel database. The update classification policy addresses the problem of proper classification of the database data. When the database is updated, the classification level of the data is determined. The data are then inserted into an object whose level dominates the level of the data. The response classification policy addresses the problem of proper classification of response to queries. This is a problem because the response may be built based on the data in many base relations. In the process of manipulating and combining the data, it is possible that the data will be used in a manner that reveals higher level information. The problem becomes more acute when one realizes that the response will be released into an environment in which many responses may be visible. Thus, the problem becomes one of aggregation and inference over time as well as across relations. In the LDV model, a response can only be released if it is placed in an object whose level dominates the derived level of the response, where the derived level is the maximum level of any information that can be deduced from the response by a subject reading this response.

Subjects interact with LDV through a request importer and a request exporter. Access to data as well as metadata is controlled by LOCK. Information in the database as well as the meta-database is stored in single-level files, i.e., LOCK objects. LOCK ensures that these database files may be manipulated only by subjects executing at the appropriate levels, and in the appropriate database domains. The three major operations performed by LDV are query, update, and metadata management. Each of these operations is an interaction between a non-LDV subject representing a user, and the LDV subjects that manipulate the database.

D. Jajodia and Sandhu Model

Jajodia and Sandhu [21] proposed a reference model for multilevel relational DBMSs and addressed on a formal basis entity integrity and update operations in the context of multilevel databases.

In the model by Jajodia and Sandhu a multilevel relation scheme is denoted as $R(A_1,C_1,\ldots,A_n,C_n,\text{TC})$, where A_i is an attribute over a domain D_i, and C_i is a classification attribute for A_i, $i = 1,\ldots,n$. The domain of C_i is the set of access classes that can be associated with attribute A_i. TC is the classification attribute of the tuples. Furthermore, for each access class c, a relational instance R_c is defined. Elements of R_c are of the form $R(a_1, c_1, \ldots a_n, c_n, \text{tc})$, where a_i is a value in the domain D_i, c_i is a classification attribute for a_i, $i = 1,\ldots,n$, and tc is the classification attribute of the tuples; tc is determined by computing the least upper bound of each c_i in the tuple. The relational instance R_c represents a view of the multilevel relation for subjects having access class c. The instance at level c is obtained from the multilevel relation by masking all attribute values whose classification is higher than or incomparable with c. This is obtained by substituting them with null values. Thus, subjects with different access classes have different views of the same multilevel relation. The entity integrity property of the standard relational data model is restated as follows: a multilevel relation R satisfies the entity integrity property if, for all instances R_c of R, and for each tuple t of R_c, the following conditions are satisfied:

1. the attributes of the primary key must be not null in t;
2. the attributes of the primary key must have the same access class in t;
3. the access class associated with a nonkey attribute must dominate the access classes associated with the attributes in the primary key.

The model by Jajodia and Sandhu supports both attribute and tuple polyinstantiation. Similar to the SeaView model, the key of a multilevel relation is defined as a combination of attributes, their classifications, and the classification of all the other attributes in the relation.

E. The MLR Data Model

The Multilevel Relational (MLR) data model proposed by Chen and Sandhu in [11] is an extension of the model proposed by Jajodia and Sandhu [21]. The data model is basically the one presented in the previous subsection, the main difference being that in the MLR data model the constraint that there can be at most one tuple in each access class for a given entity is imposed. The MLR model tries to overcome some of the ambiguities contained in the Jajodia and Sandhu model. To illustrate the problem consider a multilevel relation Employee with attributes SS#, NAME, SALARY, and DEPT#. Let SS# be the primary key. Suppose that Employee consists of the following tuples: (Sm101,U,Smith,U,10K,U,100,U,U) and (Sm101,U,Smith,U,20K,S,100,U,U). According to the Jajodia and Sandhu model, a TS subject sees a null value for attribute salary, even if there are two values for salary classified at lower levels. The alternative can be to return to the TS subject the highest value, among those available at lower levels (that is, those classified at level S).

However, there can be situation in which the TS subject would prefer the value at the lowest level, instead of that at the highest level. Thus, in the MLR model a new semantics for data classified at different levels is proposed, based on the following principles:

1. The data accepted by a subject at a given security level consist of two parts: (i) the data classified at his/her level; and (ii) the data borrowed from lower levels;
2. The data a subject can view are those accepted by subjects at his/her level and by subjects at lower levels;
3. A tuple with classification attribute c contains all the data accepted by subjects of level c.

V. MULTILEVEL SECURITY IN OBJECT DBMSs

Applying the Bell and LaPadula paradigm to object-oriented data models is not straightforward. Objects combine the properties of passive information repositories, represented by attributes and their values, with the properties of active entities, represented by methods and their invocations. Moreover, notions such as complex objects and inheritance hierarchies, make the object-oriented data model intrinsically complex. However, despite this complexity, the use of an object approach offers several advantages from the security perspective. The notion of encapsulation, which was originally introduced in object-oriented systems to facilitate modular design, can be used to express security requirements in a way that is comprehensible to the users. Moreover, messages are the only means by which objects exchange information. Due to these characteristics information flow in object systems has a very concrete and natural embodiment in terms of messages and their replies. Thus, information flows in object systems can be controlled by mediating message exchanges among objects.

In this section we present some of the major multilevel object-oriented access control models described in the literature. We start by reviewing the basic concepts of the object data model [7].

A. Object Data Model

According to the object-oriented paradigm, each real-word entity is modeled as an object. An object is associated with a unique object identifier (*oid*), which is fixed for the whole life of the object, a set of instance attributes, also called *instance variables*, and a set of procedures, called *methods*. Attributes represent relevant features of the entity being modeled (for instance, name, age, nationality, address, and ssn are examples of attributes of an object modeling a person). The value of an attribute can be an object or a set of objects. Each attribute is defined by specifying its *name* and its *domain*. Each method is defined by specifying a *signature* and a *body*. The signature specifies the name of the method and the input and output parameters. The body provides the implementation of the method. Input parameters are defined by specifying their name and their domain. Output parameters are defined by specifying their domain. Similar to attributes, the value of each parameter of a method can be either an object or a

set of objects. The methods of an object can be invoked by sending a message to the object. Upon receipt of the message, the corresponding method is executed, and a reply is returned to the object sender of the message. The reply can be an oid, a primitive value, or a special null value that denotes that no value is returned. Read, write, and create operations are enforced by means of primitive messages invoking the corresponding system-defined primitive methods. Primitive methods are elementary in the sense that they cannot invoke other methods, that is, they cannot send any messages. Primitive messages can only be sent by an object to itself. If the need arises of invoking them from other objects, they must be properly encapsulated in nonprimitive methods.

The values of the attributes of an object represent the object *state*, whereas the set of methods of an object represents the object *behavior*. Objects sharing the same set of attributes and methods are grouped together into a class. Each object belongs to (is an instance of) a class. Classes are hierarchically organized into an inheritance hierarchy. Such hierarchy represents which classes are subclasses of (inherit from) other classes. Each class inherits all the attributes and methods of its superclasses. Additionally, it can have its own methods and attributes, and it can refine the inherited ones. Each class is defined by specifying its name, the names of its superclasses, and the attributes and methods of its instances.

B. SODA Model

The SODA model [22], proposed by Keefe *et al.*, was the first to incorporate multilevel security in object-oriented data models. SODA has been designed to enforce the Bell and LaPadula principles described in Section III. A. The rules governing the model are the following:

1. Any method activation can read a value within an object only if the access class of the object is dominated by the access class of the method. If the access class of the object dominates the current access class of the method, then the method's classification is raised to the level of the object being read.
2. A method activation may modify or create a new object of a particular classification if the method's current classification equals that of the object in question, the method's current classification is dominated by the upper bound of the classification range (as specified by the constraint), and the lower bound of the classification range specified by the constraint is dominated by the subject's clearance.

If the above rules are not satisfied, then a write/create operation fails.

C. SORION Model

Thuraisingham investigated security issues for the ORION object-oriented data model [35]. The secure model was called SORION.

In SORION, subjects and objects are assigned security levels. The following rules constitute the policy:

1. A subject has read access to an object if the subject's access class dominates that of the object.

2. A subject has write access to an object if the subject's access class is equal to that of the object.
3. A subject can execute a method if the subject's access class dominates the access class of the method and that of the class with which the method is associated.
4. A method executes at the level of the subject who initiated the execution.
5. During the execution of a method m_1, if another method m_2 must be executed, then m_2 can execute only if the execution level of m_1 dominates the level of m_2 and the class with which m_2 is associated.
6. Reading and writing of objects during method execution are governed by rules 1 and 2.

D. Millen–Lunt Model

Millen and Lunt have proposed a secure object model for knowledge-based applications [26]. The security properties enforced by the model are the following:

1. The *hierarchy property* states that the level of an object dominates that of its class.
2. The *subject-level property* states that the level of a subject created to handle a message dominates both the level of the subject that originated the message and the level of the object receiving the message.
3. The *object locality property* states that a subject can execute methods or read variables only in the object where it is located, or any superclass of that object. It can write variables only in that object.
4. The **-property* states that a subject may write into an object only if its security level is equal to that of the object.
5. The *return value property* states that an invoking subject can receive a return value from a message, only if the message handler subject is at the same security level as the invoking subject.
6. The *object creation property* states that the security level of a newly created object must dominate the level of the subject that requested its creation.

E. Jajodia–Kogan Model

Information flow in object systems can be checked by mediating message exchanges among objects. This is the idea underlying the *message filtering* approach proposed by Jajodia and Kogan in [19] to prevent illegal information flows during message exchange among objects. Under this approach, every message and every reply is intercepted by a trusted component of the object system (i.e., the *message filter*) in charge of preventing illegal information flow. In particular, information can legally flow from an object o_i to an object o_j if and only if the security level of o_i is lesser than the security level of o_j. All the other flows are illegal.

Upon the receipt of a message or a reply, the message filter decides appropriate actions to be taken, based on the message (or reply) and on the classifications

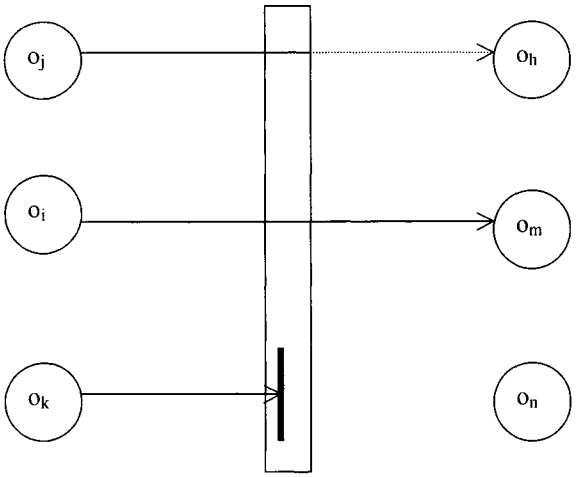


FIGURE 3 The message filter model.

of the sender and receiver object. Several actions are possible. For instance, the message can be sent unaltered to the receiver object, or it may be rejected. The latter action is taken when a low-classified object sends a message to a higher-classified object requesting to read some attributes of the latter.

A third possible action is to send the message to the receiver with the constraint that the invoked method is executed in *restricted* mode. This means that, even though the receiver can see the message, the execution of the corresponding method on the receiver should leave the state of the receiver (as well as of any other object at a level not dominated by the level of the receiver) as it was before the execution. Thus, the attributes of the receiver (and of any other object at a level not dominated by the level of the receiver) should not be modified by the method invoked upon receipt of the message.

Figure 3 exemplifies the message filter model. In the graphical representation, objects are denoted by nodes, and messages by oriented arcs from the sender object to the receiver. Each arc is intercepted by the message filter. An arc that does not pass through the message filter denotes a rejected message; an example is the message from object o_k in Fig. 3. An arc that after passing through the message filter becomes a dashed arc denotes a message to be executed by the receiver in restricted mode; an example is the message from object o_j in Fig. 3. Finally, an arc that passes through the message filter without any change denotes a message sent to the receiver, without requiring execution in restricted mode. An example is the message from object o_i in Fig. 3.

Note that information does not necessarily flow every time a message is sent between objects in that an object acquires information only by modifying its internal state, that is, by changing the values of some of its attributes. Thus, no information flow is enacted if the attribute values of an object have not been modified as part of the method executed in answer to the message. In such cases, the forward flow is said to be ineffective. Similarly, whenever a null reply is sent back as reply to a message, the backward information flow is ineffective.

In an object-oriented system, information flow can take place either (i) when a message is sent from an object to another or (ii) when an object is created. In case (i), information flow can be from the sender to the receiver, or vice versa. The forward flow is from the message sender to the receiver and is carried through the message arguments. The backward flow is from the message receiver to the sender and is carried through the message reply. In case (ii), information flow is only in the forward direction through the attribute values with which the newly created object is to be initialized. Those flows are called direct flows.

Information flow can also be indirect in that a message from an object to another may result in a method execution as part of which a message to a third object is sent. Consider for instance the case of an object o_i sending a message g_i to another object o_j. Suppose that o_j does not change its internal state as a result of receiving g_i, but instead sends a message g_j to a third object o_k. Moreover, suppose that the arguments of g_j contain information derived from the arguments of g_i (e.g., by copying some arguments of g_i to g_j). If the corresponding method execution in o_k results in updating the state of o_k, a transfer of information has taken place from o_i to o_k. Note that the flow from o_i to o_k has been enacted, even though no message exchange has been performed between o_i and o_k. Note, moreover, that a flow from o_i to o_k does not necessarily imply a flow from o_i to o_j.

The message filter intercepts every message and reply exchanged among objects to prevent both direct and indirect illegal flows of information. For an object system to be secure, all flows must be from lower level objects to higher level objects. To prevent all the illegal flows of information, the message filter makes use of a special indicator. Such an indicator, denoted in the following as *rlevel*, keeps track, for each method invocation t, of the least upper bound of the levels of all objects encountered in the sequence of method invocations starting from the object that began the computation and ending with t.

The message filter works as follows. Let o_1 and o_2 be the sender and receiver object, respectively. Moreover, let t_1 denote the method invocation on o_1 as part of which the message g_1 is sent to o_2; t_2 denotes the method invocation on o_2 performed upon receiving message g_1. Two major cases arise depending on whether g_1 is a primitive message. Let us first consider the case of nonprimitive messages. The following cases arise:

1. the sender and receiver are at the same level: the message and the reply are allowed to pass;
2. the levels of the sender and receiver are incomparable: the message is blocked and a null reply is returned to method t_1;
3. the receiver has a higher level than the sender: the message is passed through; however, the actual reply from t_2 is discarded and a null reply is returned to t_1. To prevent timing channels the null value is returned before executing t_2;
4. the receiver has a lower level than the sender: the message and reply are allowed to pass. t_2 is executed in restricted mode; that is, it is restricted from modifying the state of the receiver or creating a new object (i.e., the method invocation is memoryless). Moreover, this restriction is propagated along with further messages sent out by t_2 to other objects.

The security mediation of primitive messages is performed according to the following principles: (1) read operations are always secure because read-up operations are never allowed. Indeed, read operations are confined to an object's methods, and cannot be directly invoked by methods of other objects. The results of a read operation can thus only be exported by messages or replies, which are filtered by the message filter; (2) write operations succeed only if the status of the method invoking the operations is unrestricted. Finally, (3) create operations succeed only if the *rlevel* of the method invoking the operation is dominated by the level of the created object.

F. Modeling Multilevel Entities

An aspect common to all the proposals illustrated so far is the requirement that objects be single-level; i.e., all attributes of an object must have the same access class. The main advantage of a single-level object model, as opposed to a multilevel object model, is its simplicity and its compatibility with a security kernel. The main drawback of single-level models is that they do not reflect the real world, in which applications often need to represent multilevel entities. The solution is to use a single-level object system and map the multilevel entities onto several single-level objects. This approach has two main variants, depending on whether inheritance or aggregation is used to support the multilevel view.

For instance, Thuraisingham in [35] and Jajodia and Kogan in [19] propose an approach based on the use of inheritance hierarchies. In their proposals, each entity is mapped onto several objects, one for each access class appearing in the entity. An inheritance hierarchy is then defined over the objects onto which an entity has been mapped. The use of the inheritance hierarchy, however, has several drawbacks. First, it leads to a replication of information: since a multilevel entity is modeled as several single-level objects in a class inheritance hierarchy, some attributes of high-level objects are replicas of attributes of low-level objects (because of inheritance). Second, if not carefully monitored, updates may lead to mutual inconsistency of replicated data. In particular, changes to low-level copies of the information must be propagated to the corresponding high-level copies. Changes to high-level copies of the data should not be allowed because they cannot be propagated to low-level copies and would therefore introduce inconsistencies. Finally, this approach overloads the notion of inheritance, which is used both for conceptual specialization and for supporting multilevel entities.

The approach proposed by Millen and Lunt in [27] to store values at different classifications is based on the use of back references going from higher to lower objects. This choice is motivated by the requirement that an object cannot store references to objects at higher levels. In this approach, however, values classified at higher levels are not directly reachable by accessing the object representing the entity. Since no multilevel entity interface is provided, the burden of collecting all attributes and values referring to the same entity remains with the subject.

The approach by Boulahia-Cuppens *et al.* [8] is based on the use of reference links. In this approach, each multilevel entity is mapped onto (possibly several) single-level objects, one for each security level in the entity. Each object contains all the attributes of the multilevel entity. Values at a level lower than the level

of the object are reachable through reference links. The approach is based on the assumption that the database schema is not protected; i.e., all attribute definitions are visible to every subject (even if the subject cannot see attribute values).

Finally, Bertino *et al.* [6] have recently proposed an approach to model multilevel entities by using composite objects and delegation [17]. The approach basically consists in mapping each multilevel entity type onto a corresponding set of single-level classes, which are stored in the underlying object DBMS. Those classes are related by appropriate composite references. Moreover, each class is equipped with a set of accessor methods, which allow the retrieval of information from component objects. One important feature of the approach presented in [6] is that an attribute of a multilevel entity type may take on values at different security levels. For each level, the application designer may specify the desired policy with respect to polyinstantiated attribute values. For example, the designer may specify that a low value is a cover story, and thus it should not be considered at higher levels, or that it should be considered at a higher level only if no other value is specified for that level. The generation of such class schema can be however quite difficult. Thus, in [6] a methodology that takes as input the specification of a set of multilevel entity types organized into aggregation and inheritance hierarchies and returns a set of corresponding single-level classes is defined. Each class is equipped with the proper composite references and methods to read and write the attributes of their instances. Algorithms implementing this methodology are also presented in [6]. The single-level object representation is thus transparent to the users in that the generated single-level objects provide the same interfaces, i.e., respond to the same messages, as if multilevel objects were directly supported. The methodology proposed in [6], called Class Schema Generation (CSG), methodology consists of four steps to be sequentially executed. These steps are illustrated in Fig. 4. The first phase of the methodology, called schema completion phase, modifies the input entity-type schema to make easier the execution of the subsequent phases. The second phase is the class generation phase, which generates the single-level classes corresponding to the multilevel entity types received as input. During this phase only the class names and the inheritance relationships are defined. Attributes and methods are defined by the two subsequent phases, called attribute generation and method generation phase, respectively.

VI. SECURE CONCURRENCY CONTROL

Concurrency control in MLS/DBMSs must address two main issues: (1) ensure the correct execution of concurrent transactions; and (2) ensure multilevel security [2]. However, ensuring that transaction executions do not violate the Bell and LaPadula principles is not enough to prevent all possible leakages of information. In particular, those principles do not prevent *signaling channels*. A signaling channel may arise when two transactions access the same data. For example, a higher-level transaction, by reading lower level data, may cause a delay in the execution of a lower level transaction by attempting to write these data. In such a way, a higher level transaction may transmit sensitive

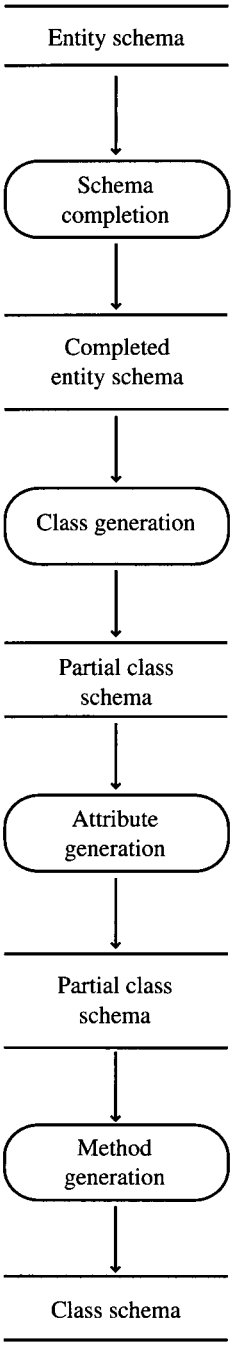


FIGURE 4 The CSG methodology.

information to lower level subjects. Thus, in addition to verify the Bell and LaPadula principles, a concurrency control protocol for MLS/DBMSs must be free of signaling channels.

In the remainder of this section we first survey the most important architectures on which secure concurrency control relies; we then illustrate some of the most relevant secure concurrency control protocols for MLS/DBMSs.

A. Architectures

Most of the research on secure transaction processing in MLS/DBMSs can be categorized into two broad categories: one based on a *kernelized architecture* and the other based on a *replicated architecture*. Both architectures rely on the notion of trusted front end (TFE), which cannot be bypassed. In the kernelized architecture (illustrated in Fig. 5) a multilevel database is partitioned into a set of single-level databases, each of which stores data at one particular level. The TFE component ensures that when a subject submits a query to the system, this query is submitted to the DBMS with the same security level as the subject. By contrast, the trusted back end makes sure that the Bell and LaPadula principles are satisfied. The main drawback of the kernelized architecture is that the query execution performance greatly degrades when queries access data from multiple security levels, since data at different levels are stored separately. In contrast, the main advantage of the kernelized architecture is that the transaction scheduler can be decomposed into several untrusted schedulers, one for each security level.

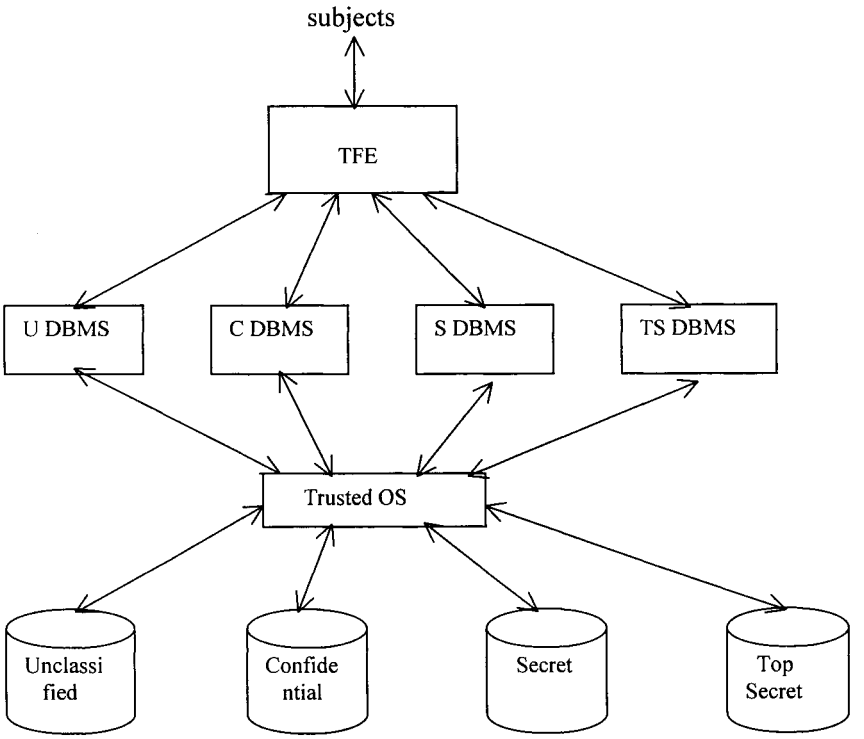


FIGURE 5 The kernelized architecture.

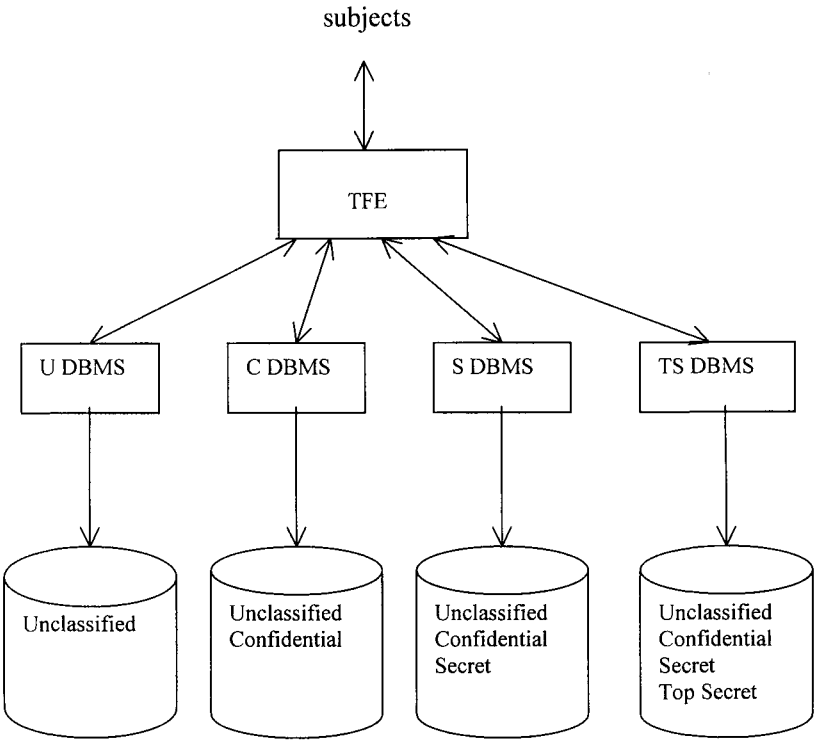


FIGURE 6 The replicated architecture.

The replicated architecture, like the kernelized architecture, uses different databases to store data at different levels. For each security level, a different database exists. However, unlike the case of the kernelized architecture, the database at a security level L contains all the data which are classified either at level L or at a level lesser than L.

The replicated architecture is illustrated in Fig. 6. In this architecture, when a transaction at a given level wishes to read data at lower levels, it will be given the replica of the lower level data stored in the database at its own security level.

Although, the replicated architecture makes query execution more efficient with respect to the kernelized architecture, it suffers from several drawbacks. First, this architectures is not practical for a large number of security levels. The propagation of updates from lower to higher levels is also a critical issue. For these reasons, in the following we consider the kernelized architecture only. We refer the interested reader to [12,16,20] for secure update propagation protocols in replicated architectures.

B. Secure Concurrency Control Protocols

The problem of secure concurrency control in MLS/DBMSs has been extensively investigated and several proposals can be found in the literature [18,23–25,31]. Algorithms for concurrency control can be divided into two main categories: *two-phase locking algorithms* and *timestamp-ordering algorithms*.

In two-phase locking algorithms, transactions should acquire locks on data before performing any actions on them. A read lock should be obtained before reading data, whereas a write lock should be obtained before writing data. With respect to locks, a transaction execution must be divided into two phases: a *growing phase*, in which the transaction should acquire all the locks it needs for its execution, and a *shrinking phase*, during which the transaction unlocks the data it has accessed. The constraint is imposed that once a transaction releases a lock, it cannot acquire any more lock.

In timestamp-ordering algorithms a unique timestamp is assigned to each transaction. Moreover, for each data, two distinct timestamps are maintained: a read timestamp and a write timestamp. A transaction is allowed to read a data only if the timestamp of the data is not greater than the timestamp of the transaction. The same constraint is applied to write operations. Upon a read or write operation the timestamp of the data is updated by setting it equal to the timestamp of the transaction.

The main problem of both the two-phase locking and the timestamp-ordering algorithms is that they are not free from signaling channels. The following example clarifies this problem.

EXAMPLE 4. Consider a database storing data at two different security levels and suppose that a two-phase locking algorithm is adopted for concurrency control. According to the multilevel security principles, a low transaction can only read and write low-level data, whereas a high transaction can read and write high data and, in addition, can also read (but not modify) low data. Consider the concurrent execution of two transactions T_1 and T_2 of level low and high, respectively. Suppose that T_2 wishes to read a data x, classified at level low. According to the two-phase locking protocol, it thus requires a read lock on x. Suppose that next T_1 wishes to modify data x. It thus requires a write lock on x, but, since x has been already locked by T_2, transaction T_1 is forced to wait until T_2 releases the lock. Thus, by selectively issuing requests to read low data, transaction T_2 could modulate the delay experienced by transaction T_1, and thus transmitting to transaction T_1 high-level information that the transaction is not authorized to access. Note that the same problem happens also in the case of timestamp-ordering algorithms.

To avoid signaling channels several approaches that apply to both the two-phase locking and the timestamp-ordering protocols have been proposed. In the following, we illustrate the most relevant proposals for both these protocols.

A method for avoiding the problem of signaling channel when a two-phase locking protocol is used is to abort a higher-level transaction having a lock on lower-level data, whenever a lower-level transaction requires the access to these data. Clearly, this approach avoids the problem of signaling channels; however, its main drawback is that it can cause transaction *starvation*, that is, it may cause a transaction to always be aborted and never complete its execution.

McDermott and Jajodia proposed a method for reducing transaction starvation [25]. The idea is that whenever a high transaction must release a lock because of a write request of a lower-level transaction, it does not abort. Rather, it enters a queue containing all the high-level transactions waiting for reading

that data. The main drawback of this approach is, however, that it does not always produce serializable schedules.

As far as timestamp-ordering protocols are concerned, the main problem with this approach is that when a high transaction reads lower-level data, it cannot modify the read timestamp of such data since this will result in a write-down operation (that violates the Bell and LaPadula principles). Several approaches to eliminate this problem and to avoid the problem of transaction starvation have been proposed. One of the solutions is to maintain multiple versions of the same data. When multiple copies of the same data are maintained, the notion of correctness for concurrent transaction executions is restated as follows. The concurrent execution of transactions is said to be correct when its effect is equivalent to that of a serial execution of the transactions on a one-copy database (this property is called *one-copy serializability*). In the following, we briefly describe two of the most relevant proposals of timestamp-ordering protocols for secure concurrency controls [18,23]. More details on this topic can be found in [2].

The approach proposed by Keefe and Tsai [23] uses a variation of the timestamp-ordering protocol, which differs from the conventional one, in the way of assigning timestamps to transactions. To avoid the problem of signaling channels a transaction is assigned a timestamp that is smaller than the timestamps of all the other transactions with a lower security level. The Keefe and Tsai method ensures secure concurrent execution of transactions, guarantees one-copy serializability, and avoids the problem of starvation. However, it uses a multilevel scheduler which, therefore, needs to be trusted.

The approach proposed by Jajodia and Atluri [18] assigns timestamps to transactions based on their arrival order. Whenever a transaction reads data from lower levels, it must postpone its committing until all the transactions from these lower levels with smaller timestamps have committed. The protocol uses a single-level scheduler that does not need to be trusted. Moreover, it guarantees secure concurrency control as well as one-copy serializability, and it avoids the problem of starvation.

VII. CONCLUSIONS

This chapter has provided a fairly comprehensive overview of the developments in secure multilevel database systems. We have first reviewed the basic concepts of access control. Then we have discussed the basic principles of mandatory access control, and we have presented two milestones in the development of access control models for multilevel DBMSs: the model by Bell and LaPadula and the model by Denning. Furthermore, we have provided an overview of the proposals for multilevel access control both in relational and in object database systems. Next, we have provided details on secure concurrency control in multilevel database systems by illustrating the architectures that can be used and the protocols that have been proposed.

Directions in secure database systems will be driven by the developments in system architectures. Database systems are no longer stand-alone systems. They are being integrated into various applications such as multimedia, electronic

commerce, mobile computing systems, digital libraries, and collaboration systems. Therefore, security issues for all these new generation systems will be very important. Furthermore, there are many developments on various object technologies such as distributed object systems and components and frameworks. Security for such systems is being investigated. Eventually, the security policies of the various subsystems and components must be integrated into policies for the entire systems. There will be many challenges in formulating policies for such systems. New technologies such as data mining will help solve security problems such as intrusion detection and auditing. However, these technologies can also violate the privacy of individuals. This is because adversaries can now use the mining tools and extract unauthorized information about various individuals. Migrating legacy databases and applications will continually be a challenge. Security issues for such operations cannot be overlooked. These new developments in data, information, and knowledge management will involve numerous opportunities and challenges for research in database security.

REFERENCES

1. Atluri V., Adam, N., Bertino E., and Ferrari, E. A content-based authorization model for digital libraries, *IEEE Trans. Knowledge Data Eng., in press.*
2. Atluri, V., Jajodia, S., and Bertino, E. Transaction processing in multilevel secure databases with kernelized architectures. *IEEE Trans. Knowledge Data Eng.* 9(5):697–708, 1997.
3. Bell, D., and LaPadula. L. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, Hanscom Air Force Base, Bedford, MA, 1975.
4. Bertino. E. Data security. *Data Knowledge Eng.* 25(1–2):199–216, 1998.
5. Bertino, E., Catania, B., and Vinai A. Transaction modeling and architectures. In *Encyclopedia of Computer Science and Technology.* Marcel Dekker, New York, 2000.
6. Bertino, E., Ferrari, E., and Samarati, P. Mandatory security and object-oriented systems: A multilevel entity model and its mapping onto a single-level object model. *Theory Practice Object Systems* 4(4):1–22, 1998.
7. Bertino, E., and Martino, L. *Object-Oriented Database Systems: Concepts and Architectures.* Addison-Wesley, Reading, MA, 1993.
8. Boulahia-Cuppens, N., Cuppens, F., Gabillon, A., and Yazdanian, K. Decomposition of multilevel objects in an object-oriented database. In *Computer Security—ESORICS 94* (D. Gollmann, Ed.), Lecture Notes on Computer Science 875, Springer-Verlag, Berlin, 1994.
9. Burns, R. Referential secrecy. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1990.
10. Castano, S., Fugini, M. G., Martella, G., and Samarati, P. *Database Security.* Addison-Wesley, Reading, MA, 1995.
11. Chen F., and Sandhu, R. S. The semantics and expressive power of the MLR data model. In *Proc. of the IEEE Symposium on Security and Privacy,* Oakland, CA, May 1995.
12. Costich, O. Transaction processing using an untrusted scheduler in a multilevel database with replicated architecture. *Database Security V: Status and Prospects.* North-Holland, Amsterdam, 1992.
13. Denning. D. E. *Cryptography and Data Security.* Addison-Wesley, Reading, MA, 1982.
14. Denning, D. E., and Lunt, T. A multilevel relational data model. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, April 1987.
15. Goguen, J., and Messeguer, J., Noninterference security policy. In *Proc. of the IEEE Symposim on Security and Privacy*, Oakland, CA, April 1982.
16. Kang, I. E., and Keefe, T. F. On transaction processing for multilevel secure replicated databases. In *Proc. European Symposyum In Research in Computer Security (ESORICS 92)*, 1992.
17. Kim, W., Bertino, E., and Garza, J. F. Composite object revisited. In *Proc. ACM Sigmod International Conference on Management of Data*, Portland, OR, 1991.

18. Jajodia, S., and Atluri, V., Alternative correctness criteria for concurrent execution of transactions in multilevel secure databases. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1992.
19. Jajodia, S., and Kogan, B. Integrating an object-oriented data model with multilevel security. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1990.
20. Jajodia, S., and Kogan, B. Transaction processing in multilevel-secure databases using replicated architeture. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1990.
21. Jajodia, S., and Sandhu, R. S. Toward a multilevel secure relational data model. In *Proc. ACM Sigmod International Conference on Management of Data*, Denver, CO, May 1991.
22. Keefe, T., Tsai, T. W., and Thuraisingham, B. SODA—A secure object-oriented database system. *Comput. Security* 8(6):1989.
23. Keefe, T., and Tsai, T. W. Multiversion concurrency control for multilevel secure database systems. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1990.
24. Lamport, L. Concurrent reading and writing. *Commun. ACM* 20(11):806–811, 1977.
25. McDermott, J., and Jajodia, S. Orange locking: Channel free database concurrency control via locking. In *Database Security VI: Status and Prospects*. North-Holland, Amsterdam, 1993.
26. Millen, J., and Lunt, T. Security for knowledge-based systems. In *Proc. of the IEEE Symposium on Security and Privacy,* Oakland, CA, 1992.
27. Millen, J., and Lunt, T. Security for object-oriented database systems. In *Proc. of the IEEE Symposium on Security and Privacy,* Oakland, CA, 1992.
28. Morgenstern, M. Security and inference in multilevel database and knowledge base systems. In *Proc. of the ACM Sigmod International Conference on Management of Data*, San Francisco, CA, 1987.
29. Morgenstern, M. A security model for multilevel objects with bidirectional relationships. In *Proc. of the 4th IFIP 11.3 Working Conference in Database Security*, Halifax, England, 1990.
30. Rosenthal, A., Herndon, W., Graubart, R., Thuraisingham, B. Security for object-oriented systems. In *Proc. of the IFIP 11.3 Working Conf. on Database Security*, Hildesheim, August 1994.
31. Reed, D. P., and Kanodia, R. K. Synchronization with event counts and sequencers. *Commun. ACM* 22(5):115–123, 1979.
32. Stachour, P., and Thuraisingham, M. B. Design of LDV—A multilevel secure database management system. *IEEE Trans. Knowledge Data Eng.* 2(2):1990.
33. Summers, R. C. *Secure Computing: Threats and Safeguard.* McGraw-Hill, New York, 1997.
34. TDI, Trusted database interpretation. Department of Defense Document, 1991.
35. Thuraisingham, B. Mandatory security in object-oriented database management systems. In *Proc. of the ACM Conference on Object-oriented Programming Systems, Languages and Applications (OOPSLA)*, New Orleans, LA, 1989.
36. Thuraisingham, B., NTML. A nonmonotonic types multilevel logic for secure databases. In *Proc. of the Computer Security Foundations Workshop*, Franconia, NH, June 1991.
37. Thuraisingham, B. A tutorial in secure database systems, MITRE Technical Report, June 1992.
38. Thuraisingham, B. Multilevel security for distributed heterogeneous and federated databases. *Comput. Security* 14, 1994.
39. Winslett, M., Ching, N., Jones, V., and Slepchin, I. Using digital credentials on the World-Wide Web. *J. Comput. Security* 5, 1997.
40. Wiseman, S. On the problem of security in databases. In *Proc of the IFIP 11.3 Conference on Database Security,* Monterey, CA, September 1989.
41. Ullman, J. *Principles of Database and Knowledge-Base Systems*, Vol. 1. Computer Science Press, Rockville, MD, 1988.

This Page Intentionally Left Blank

6

FUZZY QUERY PROCESSING IN THE DISTRIBUTED RELATIONAL DATABASES ENVIRONMENT

SHYI-MING CHEN

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan, Republic of China

HSIN-HORNG CHEN

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

This chapter presents a fuzzy query translation method based on the α-cuts operations of fuzzy numbers to translate fuzzy queries into precise queries in the distributed relational databases environment. We also present a method for estimating incomplete data when the relations stored in a failed server failed to access in the distributed relational databases environment. We have implemented a system to translate fuzzy SQL queries to precise queries in the distributed relational databases environment. The proposed methods allow the users to deal with fuzzy information retrieval in a more flexible manner in the distributed relational databases environment.

I. INTRODUCTION

Data processing is an important activity in business processing. There is a lot of information generated when business is running. A database system keeps that

Database and Data Communication Network Systems, Vol. 1

information for the enterprise. There are many types of database systems on the commercial market. Relational database systems are most widely used in the enterprise. Existing relational database systems only provide precise query operations. They cannot deal with imprecise queries. For example, if a user would only like to know whose salaries are high in his company, he cannot get the answer from the existing database systems because the query condition "high" for salary is unknown for the traditional relational database systems. Moreover, users cannot actually know what data they need. If they submit queries in current relational database systems, sometimes they cannot get the right answers. Thus, how to provide a user-friendly query mechanism or relaxed query condition to let the user get the required answer is more and more important.

Sometimes a company is distributed logically into divisions. Thus, a distributed system enables the structure of the database to mirror the structure of the company, where local data can be kept locally and the remote data can be accessed when necessary by means of computer networks. It may happen that one of the distributed databases stored in a failed server fails to access when queries are submitted from the end users. In this case, we need a mechanism for deriving the incomplete information which is nearly equal to the failed data before the failed server has recovered to the normal state.

Since Zadeh proposed the theory of fuzzy sets in 1965 [33], some researchers have investigated the application of fuzzy set theory for query translations for relational databases systems. In [9], Chen *et al.* present techniques of fuzzy query translation for relational database systems. In [31], Yeh and Chen present a method for fuzzy query processing using automatic clustering techniques. In [4], Chang and Ke present a database skeleton and introduce its application to fuzzy query translation. In [5], Chang and Ke present a method for translation of fuzzy queries for relational database systems. In [1], Bosc *et al.* propose an extension of DBMS querying capabilities in order to allow fuzzy queries against a usual database. In [14], Hou and Chen apply the fuzzy set theory to the structural query language of relational database systems. In [23], Nakajima and Senoh investigate the operations of fuzzy data in fuzzy SQL language. In [35], Zemankova proposes a fuzzy intelligent information system. In [16], Kacprzyk *et al.* developed a "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers.

In this chapter, we propose a method based on the α-cuts operations for translating fuzzy queries into precise queries in a distributed relational databases environment. We also propose a method for estimating incomplete data when one of the relations failed to access in the distributed relational databases environment. Based on the proposed methods, we also implement a system for translating fuzzy SQL queries into precise queries in the distributed relational databases environment. The proposed methods allow the users to access the distributed relational databases in a more flexible manner.

This paper is organized as follows. In Section II, we briefly review some basic concepts of fuzzy set theory from [33]. In Section III, we briefly review the fuzzy query translation method for relational database systems from [8]. In Section IV, we introduce a method for fuzzy query translation in the distributed

relational databases environment based on [8]. In Section V, we propose a method for dealing with fuzzy query translation in which some relations in the distributed databases environment failed to access. Based on the proposed method, we also implement a system for estimating the values of the attributes in the relations that failed to access in the distributed relational databases environment. The conclusions are discussed in Section VI.

II. FUZZY SET THEORY

In traditional crisp sets, an element in a set is definitely included or excluded in the set. According to this feature, some applications are limited to dealing with vague and imprecise information about the real world. Since Zadeh proposed the fuzzy set theory [33], it has been widely used to express uncertain and imprecise knowledge. The fuzzy set theory has been successfully applied in many kinds of fields, such as automatic control, decision making, pattern recognition, psychology, economics, and medical diagnosis. In the following, we briefly review some basic concepts of fuzzy set theory.

DEFINITION 1. Let U be the universe of discourse. A fuzzy set A of the universe of discourse U can be defined as

$$A = \{(u_i, \mu_A(u_i)) \mid u_i \in U\}, \tag{1}$$

where μ_A, μ_A: $U \rightarrow [0, 1]$, is the membership function of the fuzzy set A, and $\mu_A(u_i)$ is the membership grade in which the element u_i belongs to the fuzzy set A.

If the universe of discourse U is a finite set, $U = \{u_1, u_2, \ldots, u_n\}$, then the fuzzy set A can be expressed as

$$\begin{aligned} A &= \sum_{i=1}^{n} \mu_A(u_i)/u_i \\ &= \mu_A(u_1)/u_1 + \mu_A(u_2)/u_2 + \cdots + \mu_A(u_n)/u_n, \end{aligned} \tag{2}$$

where "+" means "union" and the symbol "/" means the separator.

If the universe of discourse U is an infinite set, then the fuzzy set A can be expressed as

$$A = \int_U \mu_A(u)/u, \quad u \in U. \tag{3}$$

There are three basic operations between fuzzy sets, i.e., intersection, union, and complement. Let A and B be two fuzzy sets of the universe of discourse U, $U = \{u_1, u_2, \ldots, u_n\}$, and let μ_A and μ_B be the membership functions of the fuzzy sets A and B, respectively, where

$$\begin{aligned} &\mu_A : U \rightarrow [0, 1], \\ &\mu_B : U \rightarrow [0, 1], \\ &A = \{(u_i, \mu_A(u_i)) \mid u_i \in U\}, \\ &B = \{(u_i, \mu_B(u_i)) \mid u_i \in U\}. \end{aligned}$$

The three basic operations of the fuzzy sets are reviewed from [33] as follows.

DEFINITION 2. Let $A \cap B$ be the intersection of the fuzzy set A and B defined as

$$\mu_{A\cap B}(u_i) = \min(\mu_A(u_i), \mu_B(u_i)), \quad \forall u_i \in U. \tag{4}$$

DEFINITION 3. Let $A \cup B$ be the union of the fuzzy set A and B defined as

$$\mu_{A\cup B}(u_i) = \max(\mu_A(u_i), \mu_B(u_i)), \quad \forall u_i \in U. \tag{5}$$

DEFINITION 4. Let $\bar{A}$ be the complement of the fuzzy set A defined as

$$\mu_{\bar{A}}(\mu_i) = 1 - \mu_A(u_i), \quad \forall u_i \in U. \tag{6}$$

One of the most important applications of fuzzy sets is the computational linguistics. In [34], Zadeh proposed the concept of a linguistic variable and its application to approximate reasoning. A linguistic variable is a variable whose values are linguistic terms. For example, in natural language, we use "old" or "young" to describe someone's age. Thus, we can say that "age" is a linguistic variable. In this case, the words, "old" and "young" are called fuzzy terms. The meaning of a fuzzy term is subjectively defined, and it depends on the problem domain. We can use a fuzzy membership function to represent a fuzzy term. A linguistic hedge can perform a modification on a fuzzy term. It modifies the meaning of a fuzzy set. For example, "very" and "slightly" are the linguistic hedges. After we add "very" before the fuzzy term "young," it becomes a composite fuzzy term "very young." If a fuzzy term contains an "AND" or "OR" connector, then it is called a compound fuzzy term.

There are some commonly used operations on fuzzy sets. Assume that A is a fuzzy set of the universe of discourse U and μ_A is the membership function of the fuzzy set A, then

(1) Concentration CON(A):

$$\mu_{\mathrm{CON}(A)}(x) = (\mu_A(x))^2, \quad \forall x \in U. \tag{7}$$

Figure 1 illustrates the concentration operation. This operator can be used to approximate the effect of the linguistic modifier "very." That is, for any fuzzy set A,

$$\text{very } A = A^2.$$

(2) Dilation: DIL(A)

$$\mu_{\mathrm{DIL}(A)}(x) = (\mu_A(x))^{0.5}, \forall x \in U. \tag{8}$$

Figure 2 illustrates the dilation operation. It is obvious that

$$A = \mathrm{DIL}(\mathrm{CON}(A)) = \mathrm{CON}(\mathrm{DIL}(A)). \tag{9}$$

The dilation operator can be used to approximate the effect of the linguistic modifier *More or Less*. Thus, for any fuzzy set A,

$$\textit{More or Less } A = A^{0.5} = \mathrm{DIL}(A). \tag{10}$$

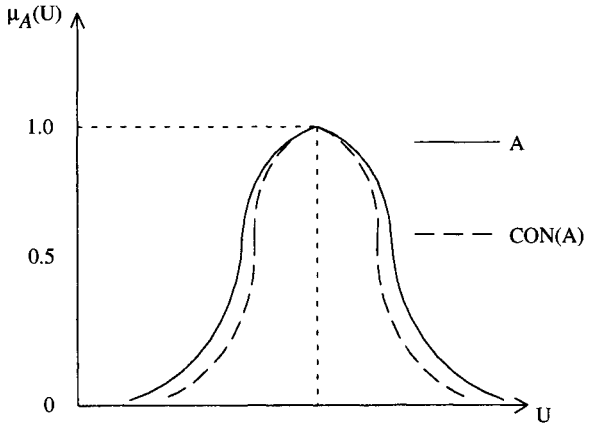


FIGURE 1 Concentration of a fuzzy set.

III. FUZZY QUERY TRANSLATION BASED ON THE α-CUTS OPERATIONS OF FUZZY NUMBERS

In this section, we briefly review a method for translating fuzzy queries into standard SQL queries from [8]. Based on [8], we will present a method for processing fuzzy queries in the distributed relational databases environment.

A fuzzy number [18] is a fuzzy set defined on the universe of discourse U that is both convex and normal. We can say that a fuzzy set A is convex if and only if for all u_1, u_2 in U,

$$\mu_A(\lambda u_1 + (1 - \lambda)u_2) \geq \text{Min}(\mu_A(u_1), \mu_A(u_2)), \tag{11}$$

where $\lambda \in [0, 1]$. A fuzzy set A of the universe of discourse U is called a normal fuzzy set if there exists $u_i \in U$ such that $\mu_A(u_i) = 1$. A fuzzy number M of the universe of discourse U may also be parametrized by a quadruple (a, b, c, d) as shown in Fig. 3.

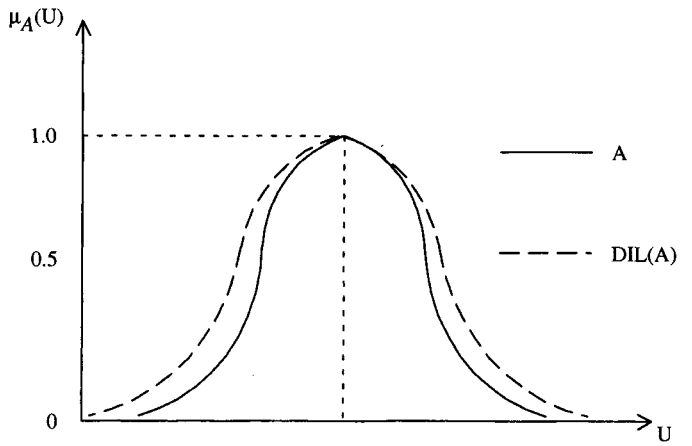


FIGURE 2 Dilation of a fuzzy set.

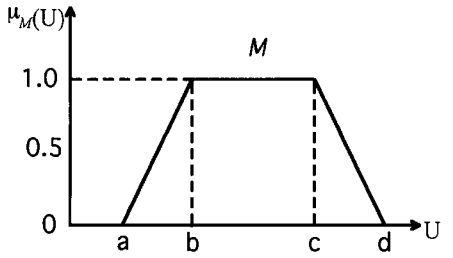


FIGURE 3 A trapezoidal fuzzy number.

We also can express the trapezoidal fuzzy number shown in Fig. 3 by the membership function shown as

$$\mu_M(x : a, b, c, d) = \begin{cases} \dfrac{x-a}{b-a}, & a \le x \le b, \\ 1, & b \le x \le c, \\ \dfrac{x-d}{c-d}, & c \le x \le d. \end{cases} \tag{12}$$

We can regard each part of this function as a linear function. Thus, if we perform the α-cuts operation on this trapezoidal membership function as shown in Fig. 4, we can obtain the α-cut range $[u_{1\alpha}, u_{2\alpha}]$ by the formulas

$$\begin{aligned} u_{1\alpha} &= \alpha(b-a) + a, \\ u_{2\alpha} &= \alpha(c-d) + d, \end{aligned} \tag{13}$$

where $\alpha \in [0, 1]$.

In this section, we briefly review a method from [8] for translating a fuzzy query into a standard query based on α-cuts operations of trapezoidal fuzzy numbers. First, we define the syntax of Fuzzy Structural Query Language (Fuzzy SQL) used in this section. It is shown as

```
SELECT  ⟨attributes⟩
FROM    ⟨relations⟩
WHERE   ⟨query conditions⟩
WITH    RSV = ⟨retrieval threshold value⟩,
```
(14)

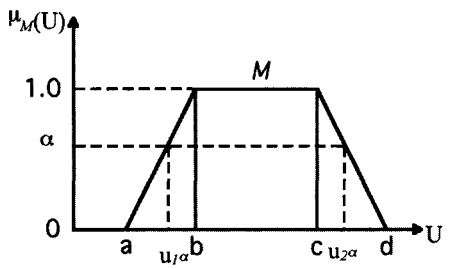


FIGURE 4 α-cuts operations of a trapezoidal fuzzy number.

where

⟨attributes⟩: List the attributes to be projected.
⟨relations⟩: Identify the tables where attributes will be projected and possibly will be joined.
⟨query condition⟩: Include the conditions for tuple/row selection within a single table or between tables implicitly joined. It may contain either standard query conditions or fuzzy query conditions.
⟨retrieval threshold value⟩: A retrieval threshold value between 0 and 1.0.

Assume that a user submits a query shown as

SELECT A
FROM T
WHERE A = C
WITH RSV = α,

where C is either a simple fuzzy term, a composite fuzzy term, or a compound fuzzy term, and α is a retrieval threshold value, where $\alpha \in [0, 1]$. We consider the following cases [8]:

Case 1. If C is a simple fuzzy term represented by a fuzzy number in the universe of discourse U, then

Step 1: Find the meaning $M(C)$ of the fuzzy term C as

$$M(C) = \{(u_i, \mu_C(u_i)) \mid u_i \in U\}. \tag{15}$$

Step 2: Perform the α-cuts operation $M(C)_\alpha$ on $M(C)$ as

$$M(C)_\alpha = \{u_i \mid \mu_C(u_i) \geq \alpha, u_i \in U, \text{and } \alpha \in [0, 1]\}. \tag{16}$$

Step 3: If $M(C)_\alpha = [u_a, u_b]$, then the original query can be translated into the statements

SELECT A
FROM T
WHERE A $\geq$ u_a AND A $\leq$ u_b
WITH RSV = α. (17)

If $M(C)_\alpha = \phi$, then print "No tuples selected."

Case 2. If C is a composite fuzzy term (i.e., $C = m_1 m_2 \ldots m_k C_j$), where $m_1, m_2 \ldots, m_k$ are some hedges and C_j is a simple fuzzy term, then:

Step 1: Find the meaning $M(C_j)$ of the fuzzy term C_j as

$$M(C_j) = \{(u_i, \mu_{Cj}(u_i)) \mid u_i \in U\}. \tag{18}$$

Step 2: Apply the semantic rules to find the meaning $M(C)$ of the composite fuzzy term C as

$$M(C) = \{(u_i, \mu_C(u_i)) \mid u_i \in U\}. \tag{19}$$

Step 3: Perform the α-cuts operation $M(C)_\alpha$ on $M(C)$ as

$$M(C)_\alpha = \{u_i \mid \mu_C(u_i) \geq \alpha, u_i \in U, \alpha \in [0, 1]\}. \tag{20}$$

Step 4: If $M(C)_\alpha = [u_a, u_b]$, then the original query can be translated into the statements

$$\begin{array}{ll} \text{SELECT} & \text{A} \\ \text{FROM} & \text{T} \\ \text{WHERE} & \text{A} \geq \text{u}_a \text{ AND A} \leq \text{u}_b \\ \text{WITH} & \text{RSV} = \alpha. \end{array} \tag{21}$$

If $M(C)_\alpha = \phi$, then print "No tuples selected."

Case 3. If C is a compound fuzzy term composited by fuzzy terms $C_1, C_2, \ldots, C_n$ connected by AND operators (i.e., $C = C_1$ AND C_2 AND ... AND C_n), then

Step 1: Find the meaning $M(C_j)$ of the fuzzy term C_j as

$$M(C_j) = \{(u_i, \mu_{Cj}(u_i)) \mid u_i \in U\}, \text{where } 1 \leq j \leq n. \tag{22}$$

Step 2: Perform the intersection operation on $M(C_1), M(C_2), \ldots, M(C_n)$ to obtain the result $M(C)$, where

$$\begin{aligned} M(C) &= M(C_1) \cap M(C_2) \cap \ldots \cap M(C_n) \\ &= \{(u_i, \mu_C(u_i)) \mid \mu_C(u_i) = \text{Min}(\mu_{C1}(u_i), \\ &\qquad \mu_{C2}(u_i), \ldots, \mu_{Cn}(u_i)), u_i \in U\}. \end{aligned} \tag{23}$$

The symbol "$\cap$" is the intersection operator between fuzzy numbers.

Step 3: Perform the α-cuts operation $M(C)_\alpha$ on $M(C)$ as

$$M(C)_\alpha = \{u_i \mid \mu_C(u_i) \geq \alpha, u_i \in U, \alpha \in [0, 1]\}. \tag{24}$$

Step 4: If $M(C)_\alpha = [u_a, u_b]$, then the original query can be translated into the statements

$$\begin{array}{ll} \text{SELECT} & \text{A} \\ \text{FROM} & \text{T} \\ \text{WHERE} & \text{A} \geq \text{u}_a \text{ AND A} \leq \text{u}_b \\ \text{WITH} & \text{RSV} = \alpha. \end{array} \tag{25}$$

If $M(C)_\alpha = \phi$, then print "No tuples selected."

Case 4. If C is a compound fuzzy term composited by fuzzy terms $C_1, C_2, \ldots, C_n$ connected by OR operators (i.e., $C = C_1$ OR C_2 OR ... OR C_n), then

Step 1: Find the meaning $M(C_j)$ of the fuzzy term C_j as

$$M(C_j) = \{(u_i, \mu_{Cj}(u_i)) \mid u_i \in U\}, 1 \leq j \leq n. \tag{26}$$

Step 2: Perform the union operation on $M(C_1), M(C_2), \ldots, M(C_n)$ to obtain the result $M(C)$, where

$$\begin{aligned} M(C) &= M(C_1) \cup M(C_2) \cup \ldots \cup M(C_n) \\ &= \{(u_i, \mu_C(u_i)) \mid \mu_C(u_i) = \text{Max}(\mu_{C1}(u_i), \\ &\qquad \mu_{C2}(u_i), \ldots, \mu_{Cn}(u_i)), u_i \in U\}. \end{aligned} \tag{27}$$

The symbol "∪" is the union operator over fuzzy numbers.

Step 3: Perform the α-cuts operation $M(C)_\alpha$ on $M(C)$ as

$$M(C)_\alpha = \{u_i \mid \mu_C(u_i) \geq \alpha, u_i \in U, \alpha \in [0, 1]\}. \tag{28}$$

Step 4: If $M(C)_\alpha = [u_a, u_b]$, then the original query can be translated into the statements

```
SELECT   A
FROM     T
WHERE    A ≥ u_a AND A ≤ u_b
WITH     RSV = α.
```
(29)

If $M(C)_\alpha = \phi$, then print "No tuples selected."

Step 5: If $M(C)_\alpha = [u_a, u_b] \cup [u_c, u_d]$, then the original query can be translated into the statements

```
SELECT   A
FROM     T
WHERE    A ≥ u_a AND A ≤ u_b OR A ≥ u_c AND A ≤ u_d
WITH     RSV = α.
```
(30)

If $M(C)_\alpha = \phi$, then print "No tuples selected."

In the following we use two examples to illustrate the fuzzy query translation process of a relational database system. Assume that the fuzzy terms for the linguistic variable "SALARY" are "high," "medium," and "low," and each of them are expressed by trapezoidal membership functions as shown in Fig. 5. We can subjectively define the membership functions of the fuzzy terms as

$$\begin{aligned} \text{high} &= (50000, 60000, 70000, 70000), \\ \text{medium} &= (25000, 35000, 45000, 55000), \\ \text{low} &= (0, 0, 23000, 30000). \end{aligned}$$

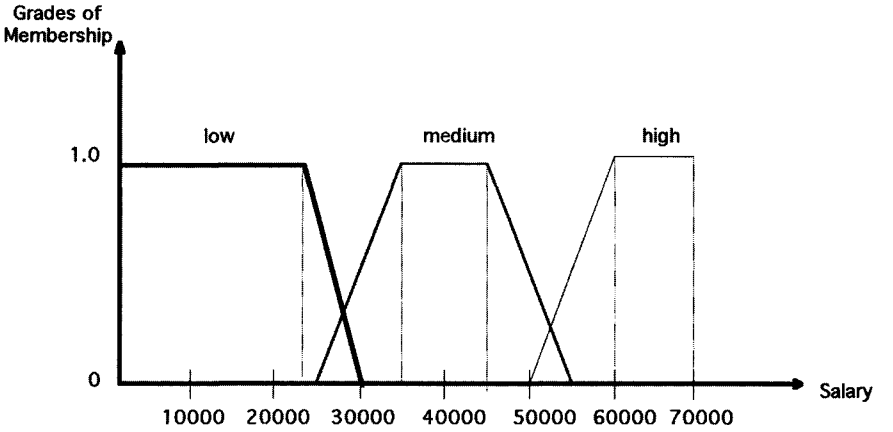


FIGURE 5 Membership functions of the fuzzy terms "low," "medium," and "high" for the linguistic variable "SALARY."

EXAMPLE 1. Assume that the user issues the fuzzy SQL query

```
SELECT ID, SALARY
FROM   EMPLOYEE
WHERE SALARY = medium OR high
WITH   RSV = 0.95.
```

In the condition part of the above fuzzy SQL query, we can see that it is a compound fuzzy query, where "medium" and "high" are the fuzzy terms of the linguistic variable "SALARY." The meanings of "medium" and "high" for the attribute SALARY are

$$M(\text{high}) = (50000, 60000, 70000, 70000),$$
$$M(\text{medium}) = (25000, 35000, 45000, 55000),$$

and the retrieval threshold value 0.95 is defined in the "WITH" clause. Thus, we can perform the union operation on $M(\text{high})$ and $M(\text{medium})$ to get $M(C)$ as shown in Fig. 6.

After performing the 0.95-cut operation on $M(C)$, we can get two intervals. We can calculate each interval by formula (13), respectively, shown as

$$u_{M(\text{medium})\_1\_0.95} = 0.95 * (35000 - 25000) + 25000 = 34500,$$
$$u_{M(\text{medium})\_2\_0.95} = 0.95 * (45000 - 55000) + 55000 = 45500,$$
$$u_{M(\text{high})\_1\_0.95} = 0.95 * (60000 - 50000) + 50000 = 59500,$$
$$u_{M(\text{high})\_2\_0.95} = 0.95 * (70000 - 70000) + 70000 = 70000.$$

Then, we can translate the user's original fuzzy query into the statements

```
SELECT   ID, SALARY
FROM     EMPLOYEE
```

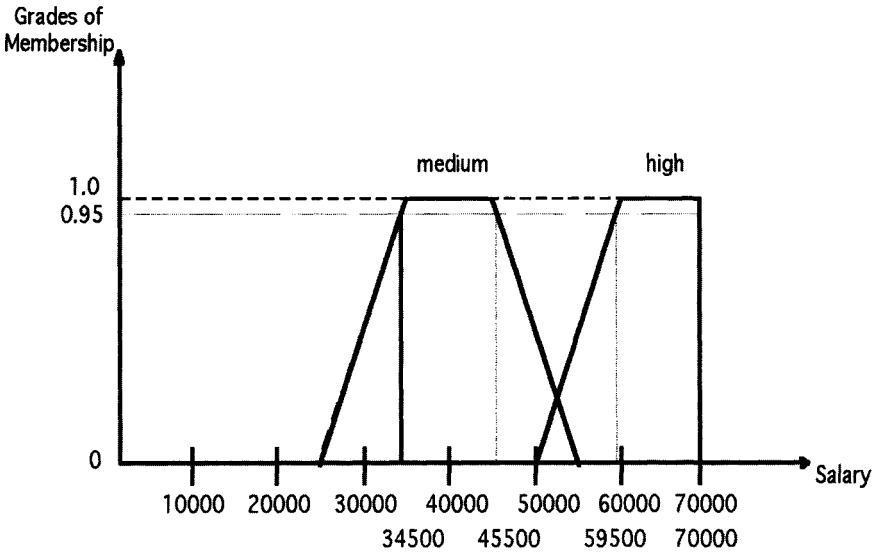


FIGURE 6 0.95-cut operations for Example 1.

```
WHERE   SALARY ≥ 34500 AND SALARY ≤ 455000 OR
        SALARY ≥ 59500 AND SALARY ≤ 70000.
WITH    RSV = 0.95.
```

EXAMPLE 2. Assume that the user issues the fuzzy SQL query:

```
SELECT  ID, NAME, SALARY
FROM    EMPLOYEE
WHERE   SALARY = very high
WITH    RSV = 0.95.
```

In the condition part of the above fuzzy SQL query, we can see that it is a composited fuzzy query; "high" is a fuzzy term of the linguistic variable "SALARY," and "very" is a linguistic hedge to modify the fuzzy term "high". The meaning of "high" is shown as

$$M(\text{high}) = (50000, 60000, 70000, 70000).$$

After we perform the linguistic hedge "very" on the fuzzy term "high", we can obtain the meaning M(very high) of the fuzzy term "very high" as shown in Fig. 7.

After performing 0.95-cut on M(very high), we can obtain an interval [59746, 70000] shown in Fig. 7. Then we can translate the user's original fuzzy query into the statement

```
SELECT  ID, NAME, SALARY
FROM    EMPLOYEE
WHERE   SALARY ≥ 59746 AND SALARY ≤ 70000.
WITH    RSV = 0.95.
```

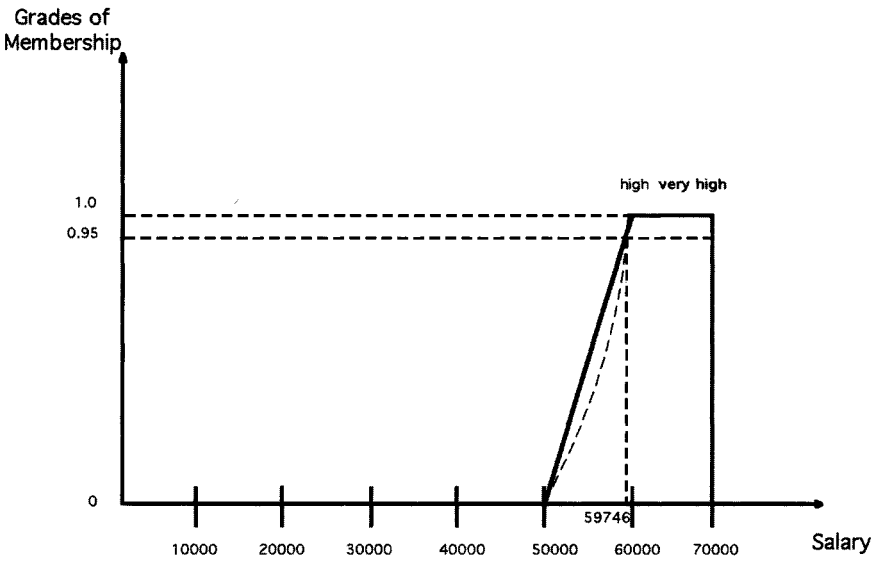


FIGURE 7 0.95-cut operation on $M(C)$ for Example 2.

IV. FUZZY QUERY TRANSLATION IN THE DISTRIBUTED RELATIONAL DATABASES ENVIRONMENT

In this section, we briefly introduce the developing tools that we use to implement the distributed relational databases query environment, Borland Delphi [22] and ODBC, and then we will present the architecture and our distributed relational databases query environment.

Delphi is a new-generation developing tool in the Windows environment. It represents a new way of developing applications for Windows. It combines the speed and ease of use of a visual development environment with the power, flexibility, and reusability of a fully object-oriented Pascal language. Briefly speaking, application developers can develop very beautiful, highly complicated, and more user-friendly applications within a very short time.

All Windows API (Application Programming Interface) are encapsulated as components in Delphi. Components are the building blocks of Delphi applications. Although most components represent visible parts of the user interface, components can also represent nonvisual elements in a program, such as timers and databases. Application developers design their applications just via the assembly of the components and within very few codes. A RAD (Rapid Application Development) developing environment is one of the powerful features of Delphi. This feature shortens all aspects of the application development cycle, and it also provides an enhanced interface and many automated features that make it easier for the application developers to develop application software.

Another feature of Delphi is that it can let application developers build sophisticated client/server applications in a short time. This is a relatively new area of DBMS programming, where large companies need to downsize those huge databases installed on mainframe computers.

The Open Database Connectivity (ODBC) interface allows applications to access data in database management systems (DBMS) using Structured Query Language (SQL) as a standard for accessing data.

The ODBC architecture has four components:

1. *Application:* It performs processing and calls ODBC functions to submit SQL statements and retrieve results.
2. *Driver manager:* It loads drivers according to the behavior of an application.
3. *Driver:* It processes ODBC function calls, submits SQL requests to a specific data source, and returns results to the application. If necessary, the driver modifies an application's request such that the request conforms to the syntax supported by the associated DBMS.
4. *Data source:* It consists of the data which the user wants to access and of its associated operating system, DBMS, and network platform (if any) used to access the DBMS.

The hierarchy of the ODBC architecture is shown in Fig. 8.

There are four parts of components in our architecture as shown in Fig. 9. We describe these components as follows.

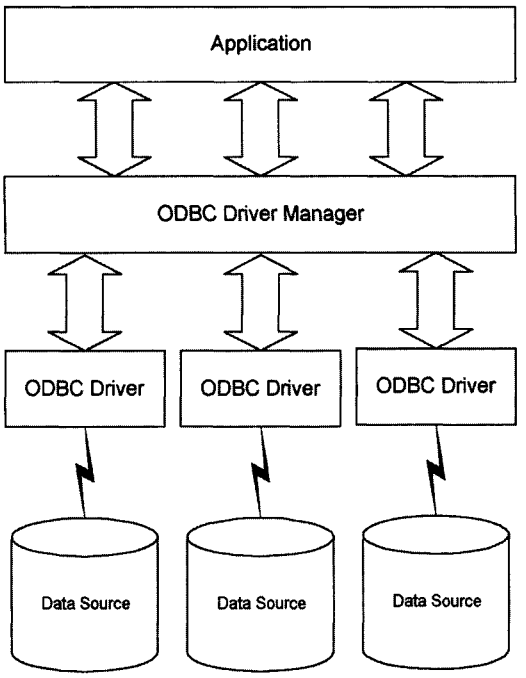


FIGURE 8 The hierarchy of the ODBC architecture.

1. *Fuzzy SQL statements:* This component contains the SQL statements that are issued by the users. The statements can be either standard SQL statements or fuzzy SQL statements.

2. *Fuzzy SQL translator:* This component parses the query statements issued from the component Fuzzy SQL statements and performs the following operations:

(1) When the query statement is not the legal SQL statement or fuzzy SQL statement that we define in Section III, this component will reject this query statement.

(2) If the query statement is a fuzzy SQL statement, this component will translate it into a standard SQL statement. According to the fuzzy term corresponding to the attribute in the condition part and the retrieve threshold value written in the query statement, the translator will perform the α-cut operation on the fuzzy term and translate the condition part into the precise SQL statement, and then pass this translated query statement to the next component. The technique used in this operation has been described in Section III.

(3) If the query statement is a standard SQL statement, this component just passes the whole SQL statement to the next component without any modification.

The membership function library is used in this component. It keeps the membership functions which are used when doing the α-cuts operations.

3. *SQL-type database management environment:* This component takes charge of access data from data sources. Database operations are actually

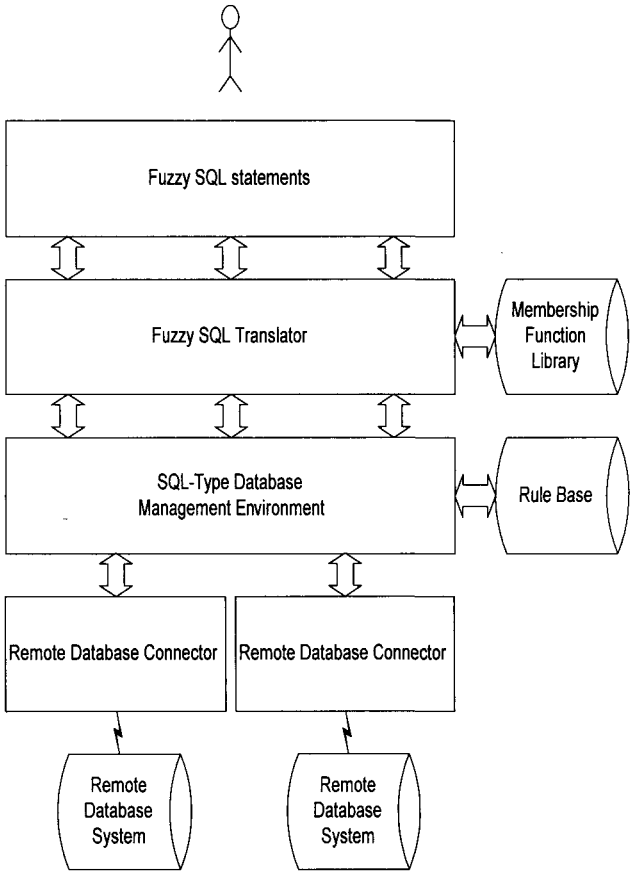


FIGURE 9 Fuzzy query translation in the distributed relational databases environment.

performed by this component. When the fuzzy SQL translator passes the precise SQL statement to this component, it will take some necessary operations to deal with this query statement. For example, if there are some databases residing in remote sites, this component maintains the communication links between the remote databases and the local program to ensure the successful access to the databases.

4. *Remote database connectors:* These components serve as the communication devices between the local program and the remote database servers. These remote database connectors are database dependent; i.e., each type of database has its specific driver. Through these drivers, the program will not necessarily decide what kind of database system it will deal with. Therefore, the methods for accessing databases are totally transparent to the program.

The architecture of this system is based on the client/server database operating environment. This system resides at the user's local site as a client. The databases can be located either in the local site or in the remote site. These two database servers act as the server sites. In our implementation, we distribute the environment to a system and two databases as shown in Fig. 10.

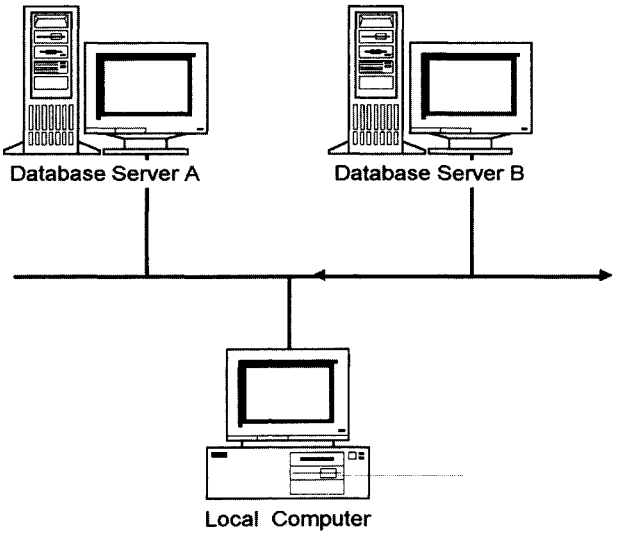


FIGURE 10 The environment of the implementation.

V. DATA ESTIMATION IN THE DISTRIBUTED RELATIONAL DATABASES ENVIRONMENT

A distributed relational database environment is a new trend for business data processing. Sometimes a business distributes its data to each related department. Each department manages its own data. This is for the convenience of speed and management. However, there are some cases where database servers fail when end users are querying databases. In this situation, we must provide a method for estimating the data stored in the failed server to approach the original values.

In the following, we use a fuzzy similarity matrix to represent a fuzzy relation. Suppose that we define a linguistic variable V that contains linguistic terms $v_1, v_2, \ldots, v_n$. Then we can make a fuzzy relation matrix for the linguistic variable V as shown in Fig. 11, where u_{ij} is the closeness degree between v_i and v_j, $u_{ij} \in [0, 1]$, $1 \leq i \leq n$, and $1 \leq j \leq n$.

We can say that the closeness degree $\mathrm{CD}(v_i, v_j)$ between the fuzzy term v_i and the fuzzy term v_j is u_{ij}, denoted as

$$CD_v(v_i, v_j) = R_v[v_i][v_j] = u_{ij}, \tag{31}$$

where each value of u_{ij} in the fuzzy relation matrix V is defined by an experienced database administrator.

	v_1	v_2	...	v_n
v_1	1	u_{12}	...	u_{1n}
v_2	u_{21}	1	...	u_{2n}
⋮	⋮	⋮	⋮	⋮
v_n	u_{n1}	u_{n2}	...	1

FIGURE 11 A fuzzy relation matrix for the linguistic variable V.

	Ph.D.	Master	Bachelor
Ph.D.	1.0	0.7	0.4
Master	0.7	1.0	0.6
Bachelor	0.4	0.6	1.0

FIGURE 12 A fuzzy relation for the linguistic variable "DEGREE".

For example, in Fig. 12, "DEGREE" is a linguistic variable. Suppose this linguistic variable contains fuzzy terms {Ph.D., Master, Bachelor}. We can construct the fuzzy relation matrix shown in Fig. 12 for the linguistic variable "DEGREE."

We can construct the fuzzy relation matrix illustrated in Fig. 12 by using the fuzzy relation table shown in Fig. 13.

From Fig. 13, we can see the closeness degree between the fuzzy terms "Master" and "Ph.D." is CD_{degree} (Master, Ph.D.) = 0.7.

We also define a fuzzy term ranking function to keep the ranks of each fuzzy term in the same linguistic domain. Suppose the linguistic term V contains fuzzy term $\{v_1, v_2\}$, and fuzzy term v_1 is prior to fuzzy term v_2; then it can be defined as

$$\text{Rank}(v_1) > \text{Rank}(v_2).$$

A rule base is used for keeping relationships in which one attribute determines other attributes. Each rule in the rule base is given by the experts according to their experiences of the data. For example, if the attributes $A_1, A_2, \ldots,$ and A_n determine the attribute AA, then we can construct a set of fuzzy rules as shown in Fig. 14, where we can see that a_{ij} denotes the attribute value, w_{ij} is the weight for each attribute in the antecedent part of the rule, and each w_{ij} is assigned by an experienced expert, where $w_{ij} \in [0, 1], 1 \leq i \leq M$, and $1 \leq j \leq n$. For example, in a relation EMPLOYEE, we can define that the attribute "DEGREE" and the attribute "EXPERIENCE" determine the attribute "SALARY," and we can assign each weight value to the attributes "DEGREE" and "EXPERIENCE", then we can have the rules for the attribute "SALARY" as shown in Fig. 15.

The rule base shown in Fig. 15 stores important information about estimating null values. We will use it to estimate failed attribute values of a relation in the distributed relational databases environment.

X	Y	CD(X,Y)
Ph.D.	Ph.D.	1.0
Master	Ph.D.	0.7
Bachelor	Ph.D.	0.4
Ph.D.	Master	0.7
Master	Master	1.0
Bachelor	Master	0.6
Ph.D.	Bachelor	0.4
Master	Bachelor	0.6
Bachelor	Bachelor	1.0

FIGURE 13 A fuzzy relation table.

Rule 1:	IF $A_1 = a_{11}$ (W=w_{11})	AND $A_2 = a_{12}$ (W = w_{12}) AND ...	AND $A_n = a_{1n}$ (W = w_{1n}) THEN AA = t_1
Rule 2:	IF $A_1 = a_{21}$ (W=w_{21})	AND $A_2 = a_{22}$ (W = w_{22}) AND ...	AND $A_n = a_{2n}$ (W = w_{2n}) THEN AA = t_2
⋮			⋮
Rule m:	IF $A_1 = a_{m1}$ (W=w_{m1})	AND $A_2 = a_{m2}$ (W = w_{m2}) AND...	AND $A_n = a_{mn}$ (W = w_{mn}) THEN AA = t_m

FIGURE 14 Rules for determining the attribute *AA*.

Sometimes we may partition a relation into two or more related relations by the method of the relation normalization [3]. Moreover, we may distribute them to different locations in the business, just for convenient management of the database. These relations are obtained by vertically partitioning the original relation. It might happen that one of the relations fails when a query is submitted. In such a case, we propose a method for estimating the failed attribute value of a relation in the distributed relational database environment.

Consider the relation EMPLOYEE shown in Fig. 16. We can vertically partition this relation into two relations EMP_INFO and EMP_SALARY as shown in Fig. 17 and Fig. 18. Then we can join these two relations (i.e., EMP_INFO and EMP_SALARY) into the original relation EMPLOYEE with the SQL statements

```
SELECT   ID, DEGREE, EXPERIENCE, SALARY
FROM     EMP_INFO, EMP_SALARY
WHERE    EMP_INFO.ID = EMP_SALARY.ID.
WITH     RSV = 0.90.
```

Suppose these two relations are distributed into two different locations. Then we can create a view called EMP by the SQL statements

```
CREATE   VIEW EMP AS
SELECT   ID, DEGREE, EXPERIENCE, SALARY
FROM     EMP_INFO, EMP_SALARY
WHERE    EMP_INFO.ID = EMP_SALARY.ID.
WITH     RSV = 0.90.
```

We know that the results of the relation EMPLOYEE and the view EMP will be identical. Now, suppose the relation EMP_SALARY failed to access when the query is submitted as

```
SELECT   *
FROM     EMP
WHERE    ID = "S1".
WITH     RSV = 0.90.
```

IF DEGREE = Master (W = 0.6) AND EXPERIENCE = 3.6 (W = 0.4)	THEN SALARY = 41000
IF DEGREE = Ph.D. (W = 0.6) AND EXPERIENCE = 7.8 (W = 0.4)	THEN SALARY = 65000
IF DEGREE = Bachelor (W = 0.65) AND EXPERIENCE = 5 (W = 0.35)	THEN SALARY = 36000

FIGURE 15 Rules for determining the attribute "SALARY".

Relation EMPLOYEE

ID	DEGREE	EXPERIENCE	SALARY
S1	Ph.D.	7.2	63000
S2	Master	2.0	37000
S3	Bachelor	7.0	40000
S4	Ph.D.	1.2	47000
S5	Master	7.5	53000
S6	Bachelor	1.5	26000
S7	Bachelor	2.3	29000
S8	Ph.D.	2.0	50000
S9	Ph.D.	3.8	54000
S10	Bachelor	3.5	35000
S11	Master	3.5	40000
S12	Master	3.6	41000
S13	Master	10.0	68000
S14	Ph.D.	5.0	57000
S15	Bachelor	5.0	36000
S16	Master	6.2	50000
S17	Bachelor	0.5	23000
S18	Master	7.2	55000
S19	Master	6.5	51000
S20	Ph.D.	7.8	65000
S21	Master	8.1	64000
S22	Ph.D.	8.5	70000

FIGURE 16 Relation EMPLOYEE.

Relation EMP_SALARY

ID	SALARY
s1	63000
s2	37000
s3	40000
s4	47000
s5	53000
s6	26000
s7	29000
s8	50000
s9	54000
s10	35000
s11	40000
s12	41000
s13	68000
s14	57000
s15	36000
s16	50000
s17	23000
s18	55000
s19	51000
s20	65000

FIGURE 17 Relation EMP_SALARY.

Relation EMP_INFO

ID	DEGREE	EXPERIENCE
s1	Ph.D.	7.2
s2	Master	2.0
s3	Bachelor	7.0
s4	Ph.D.	1.2
s5	Master	7.5
s6	Bachelor	1.5
s7	Bachelor	2.3
s8	Ph.D.	2.0
s9	Ph.D.	3.8
s10	Bachelor	3.5
s11	Master	3.5
s12	Master	3.6
s13	Master	10.0
s14	Ph.D.	5.0
s15	Bachelor	5.0
s16	Master	6.2
s17	Bachelor	0.5
s18	Master	7.2
s19	Master	6.5
s20	Ph.D.	7.8
s21	Master	8.1
s22	Ph.D.	8.5

FIGURE 18 Relation EMP_INFO.

In this case, the query cannot be processed correctly. Thus, we must propose a method for estimating the attribute values in a failed relation. The basic idea of this method is that the failed tuple compares to each rule in the rule base shown in Fig. 15 to see which is the closest rule. Then we can estimate the failed tuples by their closeness degrees.

Assume that relation R_1 contains attributes $A_1, A_2, \ldots, A_m$ and assume that relation R_2 contains attributes B_1 and B_2 Furthermore, assume that attribute A_1 and B_1 are the primary keys of R_1 and R_2 respectively, and A_1 and B_1 are in the same domain. If we perform a join operation on R_1 and R_2 with a condition "$R_1.A_1 = R_2.B_1$," then we can derive a new relation named R_3 where relation R_3 contains attributes $A_1, A_2 \ldots, A_m$, and B_2 If attribute B_2 is a failed attribute, then we can estimate attribute B_2 by the following method.

Case 1. First, we scan each tuple T_i in relation R_3. If attribute B_2 is in a numerical domain. We can compute the closeness degree between tuple T_i and the chosen rule according to the rules that determine attribute B_2. If rule r_j is the closest to T_i, then we pick that rule as the base rule, where rule r_j is shown as

$$\text{IF } A_1 = A_{j1}(W = w_{j1}) \text{ AND } A_2 = A_{j2}(W = w_{j2}) \text{ AND} \ldots \text{AND}$$
$$A_n = A_{jn}(W = w_{jn}) \text{ THEN } B_2 = N_j$$

where attribute A_k may be either a numerical or a nonnumerical domain, where $1 \leq k \leq n$. We consider these two cases:

1. A_k is in a numerical domain. If A_k is in a numerical domain, we can directly compute its closeness degree by the formula

$$\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k) = \frac{T_i.A_k}{r_j.A_k}. \tag{32}$$

2. A_k is in a nonnumerical domain. If A_k is in a nonnumerical domain and the linguistic term rank Rank $(T_i.A_k)$ > Rank $(r_j.A_k)$, then we can look up the closeness degree $\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k)$ from the fuzzy relation matrix shown in Fig. 12, where

$$\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k) = \frac{1}{R_{\mathrm{domain}}[r_j.A_k][T_i.A_k]}. \tag{33}$$

If A_k is in a nonnumerical domain and the linguistic term priority $\mathrm{Rank}(T_i.A_k) \leq \mathrm{Rank}(r_j, A_k)$, then we can look up the closeness degree $\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k)$ from the fuzzy relation matrix shown in Fig. 12, where

$$\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k) = R_{\mathrm{domain}}[r_j.A_k][T_i.A_k]. \tag{34}$$

The total closeness degree between tuple T_i and Rule r_j is

$$\mathrm{CD}(r_j, T_i) = \sum_{k=1}^{n} \mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k)^* w_{jk}. \tag{35}$$

After we compute all closeness degrees between tuple T_i and all rules r_j, we choose the closest rule to T_i. Suppose the following rule is the closest one to tuple T_i:

$$\mathrm{IF}\, A_1 = A_{j1}(W = w_{j1}) \ \mathrm{AND}\ A_2 = A_{j2}(W = w_{j2})$$
$$\mathrm{AND} \ldots \mathrm{AND}\, A_n = A_{jn}(W = W_{jn})\ \mathrm{THEN}\, B_2 = N_j.$$

Then we can estimate the attribute value of B_2 in tuple T_i by the calculation

$$T_i.B_2 = \mathrm{CD}(r_j, T_i)^* N_j. \tag{36}$$

Repeat each tuple T_i until all failed attributes have been estimated.

Case 2. If the failed attribute B_2 is in a nonnumerical domain, we can also compute the closeness degree between tuple T_i and the chosen rule according to the rules that determine attribute B_2. If rule r_j is the closest to T_i, then we pick that rule as the base rule, where Rule r_j is shown as

$$\mathrm{IF}\, A_1 = A_{j1}(W = w_{j1}) \ \mathrm{AND}\ A_2 = A_{j2}(W = w_{j2}) \ \mathrm{AND} \ldots \mathrm{AND}$$
$$A_n = A_{jn}(W = w_{jn})$$
$$\mathrm{THEN}\ B_2 = W_j$$

where attribute A_k may be either a numerical or a nonnumerical domain, where $1 \leq k \leq n$. We consider these two cases:

1. A_k is in a numerical domain. If A_k is in a numerical domain, we can directly compute its closeness degree by the formula

$$\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k) = \frac{T_j.A_k}{r_j.A_k}. \tag{37}$$

2. A_k is in nonnumerical domain. If A_k is in a nonnumerical domain and the linguistic term rank Rank $(T_i.A_k)$ > Rank $(r_j.A_k)$, then we can look up the closeness degree $\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k)$ from the fuzzy relation matrix shown in Fig. 12, where

$$\mathrm{CD}v_{ji}(r_j.A_k, T_i.A_k) = \frac{1}{R_{\mathrm{domain}}[r_j.A_k][T_i.A_k]}. \tag{38}$$

If A_k is in a nonnumerical domain and the linguistic term priority $\mathrm{Rank}(T_i.A_k)$ < $\mathrm{Rank}(r_j.A_k)$, then we can look up the closeness degree from the fuzzy relation matrix

$$\mathrm{CD}_{ji}(r_j.A_k, T_i.A_k) = R_{\mathrm{domain}}[r_j.A_k][T_i.A_k]. \tag{39}$$

The total closeness degree between tuple T_i and Rule r_j is

$$\mathrm{CD}(r_j, T_i) = \sum_{k=1}^{n} CD_{ji}(r_j.A_i, T_k.A_k)^* w_{jk}. \tag{40}$$

After we compute the all closeness degrees between tuple T_i and all rules, we choose the closest rule to T_i. Suppose the following rule is the closest one to tuple T_i:

$$\mathrm{IF}A_1 = A_{j1}(W = w_{j1}) \text{ AND } A_2 = A_{j2}(W = w_{j2}) \text{ AND} \ldots \text{AND}$$
$$A_n = A_{jn}(W = w_{jn}) \text{ THEN } B_2 = W_j.$$

Then the estimated attribute value of B_2 is W_j, where

$$T_i.B_2 = W_j \tag{41}$$

Repeat each tuple T_i until all failed attributes have been estimated.

EXAMPLE 3. Consider the relation EMPLOYEE shown in Fig. 16. We can vertically partition this relation into two relations EMP_INFO and EMP_SALARY as shown in Fig. 17 and Fig. 18.

Then we can join these two relations into the original relation EMPLOYEE with the SQL statements

```
SELECT   ID, DEGREE, EXPERIENCE, SALARY
FROM     EMP_INFO, EMP_SALARY
WHERE    EMP_INFO.ID = EMP_SALARY.ID
WITH     RSV = 0.90.
```

Suppose these two relations are distributed in two different locations. Then we create a view called EMP by the SQL statements

```
CREATE   VIEW EMP AS
SELECT   ID, DEGREE, EXPERIENCE, SALARY
```

ID	DEGREE	EXPERIENCE	SALARY
s1	Ph.D.	7.2	?????

FIGURE 19 A failed join result.

```
FROM    EMP_INFO, EMP_SALARY
WHERE   EMP_INFO.ID = EMP_SALARY.ID.
WITH    RSV = 0.90.
```

We can see that the results of the relation EMPLOYEE and the view EMP are identical. Now suppose the relation EMP_SALARY failed to access when a query is submitted shown as

```
SELECT  *
FROM    EMP
WHERE   ID = "S1".
WITH    RSV = 0.90.
```

In this case, the query cannot be processed correctly. Thus, we must estimate the attributes in the failed relation. According to the problem described above, when the database system that stores the relation EMP_SALARY failed, the query submitted by the user cannot be executed. In this case, the join result may be like the failed result shown in Fig. 19.

Now, we will estimate all the data in the relation EMP_SALARY. We scan every tuple in the relation EMP_INFO. First, we pick the first tuple in the relation EMP_INFO. From the rules for the attribute "SALARY," we know that the attribute "DEGREE" and the attribute "EXPERIENCE" determine the attribute "SALARY." Thus, we calculate the closeness degree between the first tuple of the relation EMP_SALARY to each rule for the attribute "SALARY." For example, we choose the first rule (i.e., Rule 1) in the rule base shown in Fig. 15 as

IF DEGREE = "Master" ($W = 0.6$) AND EXPERIENCE = 3.6 ($W = 0.4$)

THEN SALARY = 41000.

In the condition part of this rule, we can see that "DEGREE = "Master"" is a nonnumerical domain. We cannot directly calculate the closeness degree between "Master" and "Ph.D." We can see that Rank(Ph.D.) > Rank(Master). Therefore, we compute their closeness degree using formula (33):

$$\begin{aligned} CD_{degree}(\text{Master, Ph.D.}) &= \frac{1}{R_{degree}[\text{Master}]\,[\text{Ph.D.}]} \\ &= \frac{1}{0.7} \\ &= 1.428571428571. \end{aligned}$$

For the attribute "EXPERIENCE," we can see that it is a numerical domain attribute. According to formula (32), we can directly calculate their closeness degree:

$$\mathrm{CD}_{\mathrm{experience}}(3.6, 7.2) = \frac{7.2}{3.6} = 2.$$

From Rule 1 shown in Fig. 15, we can see that that the weights for the attributes "DEGREE" and "EXPERIENCE" are 0.6 and 0.4, respectively. Then we calculate the total closeness degree between the first tuple of the relation EMP_INFO and Rule 1 using formula (35):

$$\mathrm{CD}(\mathrm{Rule1}, \mathrm{EMP}.T_1) = 1.428571428571 * 0.6 + 2 * 0.4 = 1.6571428.$$

Again, we calculate the closeness degree between the first tuple of the relation EMP_INFO and the rest rules in the rule base for the attribute "SALARY." We use the same method described above to calculate each CD value:

$$\mathrm{CD}(\mathrm{Rule2}, \mathrm{EMP}.T_1) = 1 * 0.6 + 0.9230769230769 * 0.4 = 0.969230769,$$

$$\mathrm{CD}(\mathrm{Rule3}, \mathrm{EMP}.T_1) = 2.5 * 0.65 + 1.44 * 0.35 = 2.129.$$

In this case, we choose the closest rule to estimate the value of the attribute "SALARY," say, Rule 2. By formula (36), we can estimate the first tuple of the relation of the failed attribute "SALARY" as

$$\mathrm{EMP}.T_1.\mathrm{SALARY} = 0.969230769 * 65000 = 63000.$$

After scanning the rest of the tuples in the relation to estimate the values for the failed attribute "SALARY," we can get the estimated relation shown in Fig. 20.

Compared to the original relation, the estimated values and the estimation errors are shown in Fig. 21.

From Fig. 21, we can see that the average estimated error between the original value and the estimated value is 0.061.

EXAMPLE 4. Consider the relation EMPLOYEE shown in Fig. 16. We can vertically partition this relation into two relations EMP_INFO and EMP_SALARY as shown in Fig. 17 and Fig. 18. Then we can join these two relations into the original relation EMPLOYEE with the SQL statements

```
SELECT   ID, DEGREE, EXPERIENCE, SALARY
FROM     EMP_INFO, EMP_SALARY
WHERE    EMP_INFO.ID = EMP_SALARY.ID.
WITH     RSV = 0.90.
```

Suppose these two relations are distributed in two different locations. Then we create a view called EMP by the SQL statements

```
CREATE   VIEW EMP AS
SELECT   ID, DEGREE, EXPERIENCE, SALARY
FROM     EMP_INFO, EMP_SALARY
WHERE    EMP_INFO.ID = EMP_SALARY.ID.
```

Estimated Relation for EMP_INFO

ID	DEGREE	EXPERIENCE	SALARY (ESTIMATED)
s1	Ph.D.	7.2	63000
s2	Master	2.0	33711
s3	Bachelor	7.0	46648
s4	Ph.D.	1.2	36216
s5	Master	7.5	56200
s6	Bachelor	1.5	27179
s7	Bachelor	2.3	29195
s8	Ph.D.	2.0	39861
s9	Ph.D.	3.8	48061
s11	Master	3.5	40544
s10	Bachelor	3.5	32219
s12	Master	3.6	41000
s13	Master	10.0	64533
s14	Ph.D.	5.0	55666
s15	Bachelor	5.0	35999
s16	Master	6.2	51866
s17	Bachelor	0.5	24659
s18	Master	7.2	55200
s19	Master	6.5	52866
s20	Ph.D.	7.8	65000
s21	Master	8.1	58200
s22	Ph.D.	8.5	67333

FIGURE 20 An estimated relation for the relation EMP_INFO.

ID	DEGREE	EXPERIENCE	SALARY (ORIGINAL)	SALARY (ESTIMATED)	ESTIMATED ERROR
s1	Ph.D.	7.2	63000	63000	+0.00
s2	Master	2.0	37000	33711	-0.09
s3	Bachelor	7.0	40000	46648	+0.17
s4	Ph.D.	1.2	47000	36216	-0.23
s5	Master	7.5	53000	56200	+0.06
s6	Bachelor	1.5	26000	27179	+0.05
s7	Bachelor	2.3	29000	29195	+0.01
s8	Ph.D.	2.0	50000	39861	-0.20
s9	Ph.D.	3.8	54000	48061	-0.11
s10	Bachelor	3.5	35000	32219	-0.08
s11	Master	3.5	40000	40544	+0.01
s12	Master	3.6	41000	41000	+0.00
s13	Master	10.0	68000	64533	-0.05
s14	Ph.D.	5.0	57000	55666	-0.02
s15	Bachelor	5.0	36000	35999	-0.00
s16	Master	6.2	50000	51866	+0.04
s17	Bachelor	0.5	23000	24659	+0.07
s18	Master	7.2	55000	55200	+0.00
s19	Master	6.5	51000	52866	+0.04
s20	Ph.D.	7.8	65000	65000	+0.00
s21	Master	8.1	64000	58200	-0.09
s22	Ph.D.	8.5	70000	67333	-0.04

FIGURE 21 An estimated relation compared to the original relation.

We can see that the results of the relation EMPLOYEE and the view EMP are identical. Now suppose the relation EMP_SALARY failed to access when a query is submitted as

```
SELECT  *
FROM    EMP
WHERE   ID = "S1".
WITH    RSV = 0.90.
```

In this case, the query cannot be processed correctly. Thus, we must estimate the attributes in the failed relation. When the database system that stores the relation EMP_INFO failed, the query submitted previously cannot be processed. In this case, the join result may be like the failed result shown in Fig. 22.

We can see that the attribute "DEGREE" failed in the relation EMP. Therefore, we must estimate the values in the attribute "DEGREE" in the relation EMP. We scan every tuple in the relation EMP, and pick the first tuple in the relation EMP. From the rules for the attribute "DEGREE," we know that the attribute "EXPERIENCE" and the attribute "SALARY" determine the attribute "DEGREE". Thus, we calculate the closeness degree between the first tuple of the relation EMP and each rule. Suppose we define the rules for estimating the attribute "DEGREE" as shown in Fig. 23.

We choose the first rule in the rule base shown in Fig. 23 as

IF EXPERIENCE = 5.0 ($W = 0.2$) AND SALARY = 57000 ($W = 0.8$)

THEN DEGREE = Ph.D.

Relation EMP

ID	DEGREE	EXPERIENCE	SALARY
s1	???	7.2	63000
s2	???	2.0	37000
s3	???	7.0	40000
s4	???	1.2	47000
s5	???	7.5	53000
s6	???	1.5	26000
s7	???	2.3	29000
s8	???	2.0	50000
s9	???	3.8	54000
s10	???	3.5	35000
s11	???	3.5	40000
s12	???	3.6	41000
s13	???	10.0	68000
s14	???	5.0	57000
s15	???	5.0	36000
s16	???	6.2	50000
s17	???	0.5	23000
s18	???	7.2	55000
s19	???	6.5	51000
s20	???	7.8	65000
s21	???	8.1	64000
s22	???	8.5	70000

FIGURE 22 A failed join result in the relation EMP.

IF EXPERIENCE = 5.0 (W = 0.2) AND SALARY = 57000 (W =0.8) THEN DEGREE = Ph.D.
IF EXPERIENCE = 3.5 (W = 0.2) AND SALARY = 40000 (W =0.8) THEN DEGREE = MASTER
IF EXPERIENCE = 2.3 (W = 0.2) AND SALARY = 29000 (W =0.8) THEN DEGREE = Bachelor

FIGURE 23 Rules for determining the attribute "DEGREE."

For the attribute "EXPERIENCE," we can see that it is a numerical domain attribute. According to formula (37), we can directly calculate its closeness degree:

$$\mathrm{CD}_{\mathrm{experience}}(5.0, 7.2) = \frac{7.2}{5.0} = 1.44.$$

For the attribute "SALARY," we can see that it is also a numerical-domain attribute. According to formula (37), we can directly calculate its closeness degree:

$$\mathrm{CD}_{\mathrm{salary}}(57000, 63000) = \frac{63000}{57000} = 1.105263158.$$

From Rule 1 of the rule base shown in Fig. 23, we can see that the weights for the attributes "EXPERIENCE" and "SALARY" are 0.2 and 0.8, respectively. Then we calculate the total closeness degree between the first tuple of the relation EMP and Rule 1 using formula (40):

$$\mathrm{CD}(\mathrm{Rule1}, \mathrm{EMP}.T_1) = 1.44 * 0.2 + 1.105263158 * 0.8 = 1.088.$$

Again, we can calculate the closeness degree between the first tuple of the relation EMP and the rest rules in the rule base for the attribute "DEGREE." We can use the same method described above to calculate each CD value:

$$\mathrm{CD}(\mathrm{Rule2}, \mathrm{EMP}.T_1) = 2.05714 * 0.2 + 1.575 * 0.8 = 1.2114,$$

$$\mathrm{CD}(\mathrm{Rule3}, \mathrm{EMP}.T_1) = 3.13043 * 0.2 + 2.172 * 0.8 = 2.2261.$$

In this case, we choose the closest rule to estimate the value of the attribute "DEGREE," i.e., Rule 1. By formula (41), we can estimate the failed attribute "DEGREE" as

$$\mathrm{EMP}.T_1.\mathrm{DEGREE} = \mathrm{Rule1.Degree} = \mathrm{Ph.D.}$$

After scanning the rest of the tuples in the relation to estimate the values for the failed attributes "DEGREE," we can get the estimated relation shown in Fig. 24.

The estimated values and the estimated errors are compared to the original relation in Fig. 25.

The architecture of the implemented system based on the proposed methods is shown in Fig. 26. There are five parts of components in the implemented system, i.e., fuzzy SQL statements, fuzzy SQL translator, SQL-type database management environment, remote database connectors, and failed data generator.

The components of fuzzy query statements, fuzzy SQL translator, SQL-type database management environment, and remote database connector have already been described in Section IV, while the failed data generator is based

Relation EMP

ID	DEGREE (ESTIMATED)	EXPERIENCE	SALARY
s1	Ph.D.	7.2	63000
s2	Bachelor	2.0	37000
s3	Master	7.0	40000
s4	Bachelor	1.2	47000
s5	Master	7.5	53000
s6	Bachelor	1.5	26000
s7	Bachelor	2.3	29000
s8	Bachelor	2.0	50000
s9	Master	3.8	54000
s10	Bachelor	3.5	35000
s11	Master	3.5	40000
s12	Master	3.6	41000
s13	Ph.D.	10.0	68000
s14	Ph.D.	5.0	57000
s15	Bachelor	5.0	36000
s16	Master	6.2	50000
s17	Bachelor	0.5	23000
s18	Master	7.2	55000
s19	Master	6.5	51000
s20	Ph.D.	7.8	65000
s21	Ph.D.	8.1	64000
s22	Ph.D.	8.5	70000

FIGURE 24 An estimated relation for the relation EMP.

on the method presented in this section. Again, we briefly describe these components as follows:

1. *Fuzzy SQL statements:* It contains the SQL statements that are issued by the users.
2. *Fuzzy SQL translator*: It parses the query statements issued from the component fuzzy SQL statement and performs the syntax checking and the

ID	DEGREE (ORIGINAL)	EXPERIENCE	SALARY	DEGREE (ESTIMATED)
s1	Ph.D.	7.2	63000	Ph.D.
s2	Master	2.0	37000	Bachelor
s3	Bachelor	7.0	40000	Master
s4	Ph.D.	1.2	47000	Bachelor
s5	Master	7.5	53000	Master
s6	Bachelor	1.5	26000	Bachelor
s7	Bachelor	2.3	29000	Bachelor
s8	Ph.D.	2.0	50000	Bachelor
s9	Ph.D.	3.8	54000	Master
s10	Bachelor	3.5	35000	Bachelor
s11	Master	3.5	40000	Master
s12	Master	3.6	41000	Master
s13	Master	10.0	68000	Ph.D.
s14	Ph.D.	5.0	57000	Ph.D.
s15	Bachelor	5.0	36000	Bachelor
s16	Master	6.2	50000	Master
s17	Bachelor	0.5	23000	Bachelor
s18	Master	7.2	55000	Master
s19	Master	6.5	51000	Master
s20	Ph.D.	7.8	65000	Ph.D.
s21	Master	8.1	64000	Ph.D.
s22	Ph.D.	8.5	70000	Ph.D.

FIGURE 25 An estimated relation compared to the original relation.

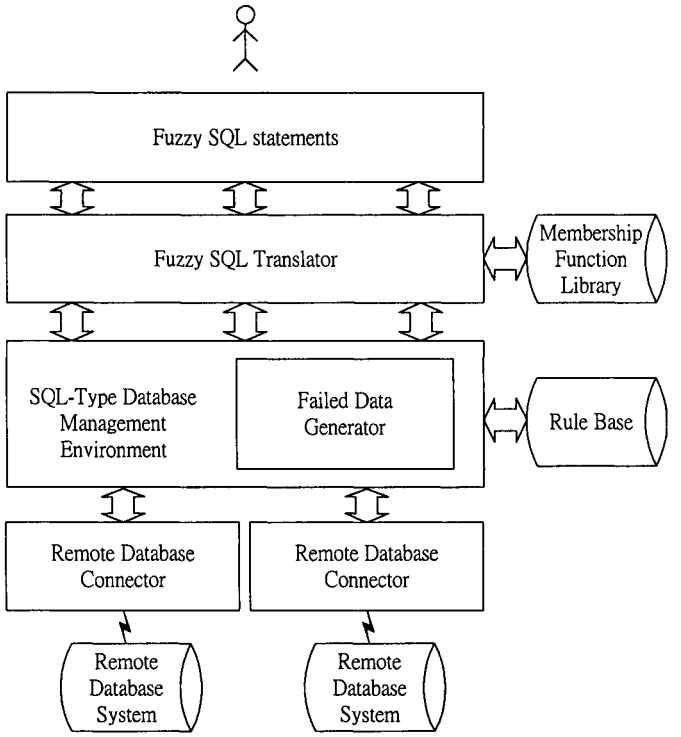


FIGURE 26 Fuzzy query translation in the distributed relational databases environment including the failed data generator.

fuzzy query statement translation. The membership function library is used in this component. It also keeps the membership functions that are used when doing the α-cut operations.

3. *SQL-type database management environment:* It offers data access from the data sources. Database operations are actually performed by this component.

4. *Failed data generator:* If one of the remote databases failed when users submit query statements, the failed data generator component will be activated to perform the failed data generation based on the proposed method described previously. These data are all estimated to the original data as close as possible. When the estimated data are generated, the other part of the fuzzy query translation process in the SQL-type database management environment component will be continued until the whole process of the fuzzy query translation is performed completely. The rule base is used in the failed data generator component. It provides any necessary information for estimating the failed data.

5. *Remote database connectors:* These components communicate between the local program and remote database servers. ODBC and BDE serve as this function. These components make database access transparent to the program.

VI. CONCLUSIONS

We have presented a method for dealing with fuzzy query translation based on the α-cuts operations of trapezoidal fuzzy numbers for the distributed relational databases environment. It allows the users to access the information in the distributed relational databases environment in a more flexible manner. It also allows the users to submit fuzzy SQL queries and to translate the fuzzy SQL queries into standard SQL queries. We also present a method for estimating incomplete data when one of the relations failed to access in the distributed relational databases environment. Based on the proposed methods, we have implemented a system on a Pentium PC by using the Borland Delphi version 3.0 developing environment to translate fuzzy SQL queries into precise SQL queries. It is transparent to the user who submits fuzzy queries to the relational database system, and it also provides a method for null values estimation when the queried relations failed to access in the distributed relational databases environment. The proposed methods allow the users to deal with fuzzy information retrieval in a more flexible manner in the distributed relational databases environment.

REFERENCES

1. Bosc, P., Galibourg, M., and Hamon, G. Fuzzy querying with SQL: Extensions and implementation aspects. *Fuzzy Sets Systems* 28(3): 333–349, 1988.
2. Bosc, P., and Pivert, O. SQLf: A relational database language for fuzzy querying. *IEEE Trans. Fuzzy Systems* 3(1): 1–17, 1995.
3. Codd, E. F. Normalized data base structure: a brief tutorial. In *Proceedings of 1971 ACM SIGFIDET Workshop on DATA Description, Access, and Control*, San Diego, CA, November 1971.
4. Chang, S. K., and Ke, J. S. Database skeleton and its application to fuzzy query translation. *IEEE Trans. Software Engrg.* 4(1): 31–43, 1978.
5. Chang, S. K., and Ke, J. S. Translation of fuzzy queries for relational database systems *IEEE Trans. Pattern Anal. Mach. Intell.* 1(3): 281–294, 1979.
6. Chen, H. H., and Chen, S. M. Fuzzy query translation for information retrieval in the distributed relational database environment. In *Proceedings of the 6th National Conference on Science and Technology of National Defense*, Taoyuan, Taiwan, Republic of China, vol. 2, pp. 433–439, 1997.
7. Chen, S. M. Using fuzzy reasoning techniques for fault diagnosis of the J-85 jet engines. In *Proceedings of the Third National Conference on Science and Technology of National Defense*, Taoyuan, Taiwan, Republic of China, vol. 1, pp. 29–34, 1994.
8. Chen, S. M., and Jong, W. T. Fuzzy query translation for relational database systems. *IEEE Trans. Systems Man Cybern et. Part B* 27(4): 714–721, 1997.
9. Chen, S. M., Ke, J. S., and Chang, J. F. Techniques of fuzzy query translation for database systems. In *Proceedings of 1986 International Computer Symposium*, Tainan, Taiwan, Republic of China, vol. 3, pp. 1281–1290, 1986.
10. Date, C. J. *An Introduction to Database Systems*, 6th ed. Addison-Wesley, Reading, MA, 1995.
11. Grant, J. Null values in a relational data base. *Inform. Process. Lett.* 6(5): 156–157, 1977.
12. Huemer, C., Happel, G., and Vieweg, S. Migration in object-oriented database systems - a practical approach. *Software—Practice Exper.* 25:1065–1096, 1995.
13. Jeng, B., and Liang, T. Fuzzy indexing and retrieval in case-based systems. In *Proceedings of 1993 Pan Pasific Conference on Information Systems*, Taiwan, Republic of China, pp. 258–266, 1993.

14. Hou, Y. C., and Chen, C. M., Apply the fuzzy set theory to the structural query language of database. In *Proceedings of International Joint Conference of CFSA/IFIS/SOFT'95 on Fuzzy Theory and Applications*, Taipei, Taiwan, Republic of China, pp. 107–113, 1995.
15. Hou Y. C., and Wu, W. T. Fuzzy query processing in fuzzy database. In *Proceedings of International Joint Conference of CFSA/IFIS/SOFT'95 on Fuzzy Theory and Applications*, Taipei, Taiwan, Republic of China, pp. 114–120, 1995.
16. Kacprzyk, J., Zadrozny, S, and Ziokkowski, A. FQUERY III +: A "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers. In *Proceedings of the Second International Fuzzy Systems Association Congress*, Tokyo, Japan, pp. 443–453, 1987.
17. Kacprzyk, J., and Zadrozny, S. Fuzzy query in Microsoft Access V.2. In *Proceedings of the FUZZ-IEEE/IFES'95 Workshop on Fuzzy Database Systems and Information Retrieval*, Yokohama, Japan, pp. 61–66, March 1995.
18. Kaufman, A., and Gupta, M. M. *Introduction to Fuzzy Arithmetic.* Van Nostrand Reinhold, New York, 1985.
19. Kaufmann, A., and Gupta, M. M. *Fuzzy Mathematical Models in Engineering and Management Science.* North-Holland, Amsterdam, 1988.
20. Kim, W. Object-oriented database systems: promises, reality and future. In *Proceedings of the 19th International Conference on Very Large Data Bases*, pp. 676–687, 1994.
21. Motro, A. VAGUE: A user interface to relational database that permits vague queries. *ACM Trans. Office Inform. Systems* 187–214, 1988.
22. Mueller, J. *Peter Norton's Guide To Delphi 2*, premier ed. Sams, Indianapolis, IN, 1996.
23. Nakajima, H., and Senoh, Y. Development of fuzzy add-in macros with spread sheet. In *Proceedings of the FUZZ-IEEE/IFES'95 Workshop on Fuzzy Database Systems and Information Retrieval*, Yokohama, Japan, pp. 61–66, 1995.
24. Nakajima, H., Sogoh, T., and Arao, M. Operations on fuzzy data in fuzzy SQL language. In *Proceedings of 1993 Pan Pacific Conference on Information Systems*, Taiwan, Republic of China, pp.1–6, 1993.
25. Prade, H. Lipski's approach to incomplete information databases restated and generalized in the setting of Zadeh's possible theory. *Inform. Systems* 9(1): 27–42, 1984.
26. Prade, H., and Testemale, C. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Inform. Sci.* 115–143, 1984.
27. Umano, M., Hatono, I., and Tamura, H. Fuzzy database systems. In *Proceedings of the FUZZ-IEEE/IFES'95 Workshop on Fuzzy Database Systems and Information Retrieval*, Yokohama, Japan, pp. 35–36, 1995.
28. Vassiliadis, S. A fuzzy reasons database question answering system. *IEEE Trans. Knowledge Data Engrg.* 6(6): 868–882, 1994.
29. Vila, M. A., Cubero, J. C., Medina, J. M., and Pons, O. "A logic approach to fuzzy relational databases. *Int. J. Intell. Systems* 9(5): 449–460, 1994.
30. Vila, M. A., Medina, J. M., Pons, O., and Cubero, J. C. Toward the computer implementation of fuzzy relational and deductive database system. In *Proceedings of the FUZZ-IEEE/IFES'95 Workshop on Fuzzy Database Systems and Information Retrieval*, Yokohama, Japan, pp. 67–73, March 1995.
31. Yeh. M. S., and Chen, S. M. A new method for fuzzy query processing using automatic clustering techniques, *J. Computers* 6(1): 1–10, 1994.
32. Yen, S. J. Neighborhood/conceptual query answering with imprecise/incomplete data. Master Thesis, Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R. O. C., June 1993.
33. Zadeh, L. A. Fuzzy sets. *Inform. Control* 8: 338–353, 1965.
34. Zadeh, L. A. The concept of a linguistic variable and its application to approximate reasoning *Inform. Sci.* 8(3): 199–249, 1975 (part I); 8, no. (4): 301–357, 1975 (part II); 9(1): 43–80, 1975 (part III).
35. Zemankova, M. FIIS: A fuzzy intelligent information system. *Data Engrg.* 11(2): 11–20, 1989.
36. Zemankova, M., and Kandel, A. Implementing imprecision in information systems. *Inform. Sci.* 37:107–141, 1985.
37. Zimmermann, H. J. *Fuzzy Set Theory and Its Applications*, 2nd ed. Kluwer Academic, Boston, 1991.

7

DATA COMPRESSION: THEORY AND TECHNIQUES

GÁBOR GALAMBOS
JÓZSEF BÉKÉSI

Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

I. INTRODUCTION

Despite continuing improvements in storage and transmission technology the rapidly growing amount of information stored on and transmitting between computers has increased the need for text compression. A huge amount of information can be stored on a single CD but sometimes this is not enough to avoid multivolume applications. It is known that a simple ad hoc method can compress an English text to around 70% of its original size, and the best techniques have a compression ratio about 25%. In case of Braille the used compression processes can reduce the size of books by 20%. For computers,

Database and Data Communication Network Systems, Vol. 1

compression can increase the amount of information stored on a disk. If archive—and compressed—data are stored on magnetic tapes not only fewer tapes are needed, but fewer shelves are used to store them and less time is taken to read them. The compression of transmitted data has another benefits: the cost of transmission is reduced and the effective speed of transmission also increases.

Data compression has a side effect for secret services: using compression some encryption is achieved because the lack of redundancy in a compressed file removes the opportunity to use statistical regularities to break a code. Some schemes generally change their coding after each character is transmitted, so the codebreaker's task is compounded.

Since the pioneer work of Shannon [60] it has been known that most of the data are represented by more bits than the minimal needed considering its information contents. In other works we can say that the description has a redundancy. In a data compression process the aim is to decrease the redundancy of a given data description. The standard processes that use this information in a computer system require standard forms e.g., ASCII and BMP. Despite the fact that computer storage capacity and the speed of a transmission line are growing extraordinary rapidly, it seems to be evident that these two fields of computer science are the most common applications of data compression processes. In both cases first we compress the data, so either we use smaller space to store the data in a secondary storage equipment or we need to transmit a decreased amount of bytes through a transmission channel. Thereafter, for further processing we decompress them again for getting the earlier standard form.

Information can appear in different forms like text, image, or voice. From a data compression point of view there is a fundamental difference among them: In case of a textual information performing a compression/decompression process we must recover exactly the original data, while in the case of pictures or voices—without getting into deep trouble—it is allowed to get an approximation of the original information. In this chapter we deal with the textual data compressions; i.e., we consider only the so-called lossless techniques.

Intuitively, it is not difficult to see that if we know the relative frequencies of each character—i.e., we know the distribution of the characters—in a given text, then using a compression process the shorter the codes we can order of the more frequently appearing characters, the better the result is. In other words we can say: if we can give a good prediction for the next character then we can reach a better compression. This prediction can be made in a model where we estimate the probability distribution for the next character. The more the model resembles the source of the text, the better the messages can be compressed. The task of finding a suitable model for text is an extremely important problem in compression, and a lot of work has been done in this direction. A good overview of the different models can be found in [10]. Separating the compression process into two parts—predicting by a model and encoding the text into binary digits by an encoder—is one of the major advances in the theory of data compression over the past years. In practice we encode a given text by a model and after transmitting the compressed text through a communication channel we decode the text. It is very important that in this later process

we must use the same model. If we use a fixed model for different texts then the transmission may become inefficient because the model is not appropriate for the text. Such a problem can occur if we use a Morse code handling numeric data only. There are different ways to maintain a model in a compression process:

- In a *static modeling* the encoder and decoder use a fixed model, regardless of the text to be encoded.
- In a *semiadaptive scheme* before sending the text the encoder checks it and prepares a "codebook." Then he transmits the codebook, followed by the coded text. The decoder first gets this codebook and then uses it to decode the text. This technique has two disadvantages: firstly, the encoder must see the entire text before transmitting it, and secondly, we must send the codebook for each text, which could take a substantial amount of time. So, two passes are required for encoding.
- If we use an *adaptive model* then the encoder sends a text character by character. By agreement, when a part of the text (we can refer to it as "word") has been transmitted—and also received—twice, then both the decoder and encoder add it to the codebook. In the future that word can be transmitted using the new code. So, if we use adaptive coding, then the code of a particular character (or word) is based on the text already transmitted.

There is another possibility to distinguish different techniques. Certain classical statistical methods use static models, and they transmit texts character by character. Unfortunately, sometimes we do not know the whole text in advance, so we need one of the other techniques to perform a compression. On the other hand, there are such methods that use the so-called set of dictionary coding techniques. They use a—static or adaptive—dictionary in which they correspond different parts of the source text to other—coded—sequences. In the following firstly we overview the information theoretical backgrounds and the basic definitions of the data compression. Section III deals with the different statistical coding methods. Some dictionary coding techniques are considered in Section IV. In Section V we discuss some universal coding methods, and further special techniques are mentioned in Section VI.

II. FUNDAMENTALS OF DATA COMPRESSION

DEFINITION 1. Compression refers to a process that produces a shorter $\Delta(D)$ information from a given source information D.

DEFINITION 2. Lossless compression is a procedure in which D can be exactly restored from $\Delta(D)$. We call lossy compression such a method in which the original information can be decoded only approximately.

In case of lossless compression the later decompressed data must be identical to the original source code. In this paper we deal only with lossless compression methods, so we always assume that our aim is the exact reconstruction. Usually this kind of method can be used in the cases of database, text, and other

data files. The efficiency of the compression method is significantly affected by the characteristic of the source information and the applied technique.

From an information theoretical point of view, if we talk about compression the most important factor is the information content of the data we would like to compress. Usually the real amount of storage occupied by the information does not reflect this. Many times data contain some redundancy and require more storage than their real information content would contain. The main aim of the compression is to remove this redundancy from the information. Before the detailed description of compression methods we would like to mention a simple theorem and also present a very short proof. This theorem is very important and determines many investigations into the theory and techniques of compression methods.

THEOREM 2.1 [48]. *There is no such lossless compression method that compresses each data.*

Proof. Suppose we have such an algorithm, and for a given string S_i we denote the compressed string by $\Delta_A(S_i)$. We also suppose that the length of S_i is n bits. We know that the total number of different binary series with length n is 2^n.

Take now all the binary series of length less than n. The total number of such series is

$$1 + 2 + 2^2 + \cdots + 2^{n-1} = 2^n - 1.$$

So there are at least two different strings S_i and S_j for which $\Delta_A(S_i) = \Delta_A(S_j)$. Decoding these coded-strings we cannot distinguish between S_i and S_j, which is a contradiction. ■

A. Information Theoretical Background

The theoretical background of data compression is mainly based on some results of information theory. The fundamental of this theory was worked out by Claude Shannon more than 50 years ago [60], and later many books and publications appeared in this topic (see, e.g., [1,28,33]). This theory investigates questions of information storage and communication, too, and data compression became one of its most important application. Before overviewing these important results in information theory that have been used in the field of data compression we introduce some definitions and basic theorems. As we mentioned, in this chapter we are dealing only with text compression. So, to be precise we need to introduce a model. The model is based on the fact that a letter in a text has been chosen from an alphabet.

DEFINITION 3. We refer to a nonempty set $A_n = \{a_1, \ldots, a_n\}$, $\infty > n \geq 1$, as an alphabet and the elements of A_n are called characters or symbols. A series of characters of A_n is called a string. Usually we assume that the length of a string is also finite.

DEFINITION 4. We call the source of such a procedure F_n, which emits the characters of A_n. A source is called first order if we can order to the characters of

A_n some independent probabilities $p_1, \ldots, p_n$, where $p_i > 0, i = 1, \ldots, n$ and $\sum_{i=1}^{n} p_i = 1$. We refer to the p_i's as the emitting probabilities of the characters.

DEFINITION 5. Let $K_n = \{\alpha_1, \ldots, \alpha_n\}$ be a set of binary series of finite length where the lengths of certain series can be different. We call such a procedure character coding, which orders a binary series $\alpha_i \in K_n$ to each character of an alphabet A_n. K_n is called the code and the elements $\alpha_1, \ldots, \alpha_n$ are called the code-words. The binary series of α_i will be denoted by $\omega_1\omega_2 \cdots \omega_{l_i}$ where $\omega_k \in \{0, 1\}$.

DEFINITION 6. A code K_n is called uniquely decipherable, if an arbitrary binary series can be uniquely divided into products of code-words. (Here product means the concatenation of the code-words).

It is clear that in the case of text compression we are interested in only uniquely decipherable codes. Before thinking on how we can produce a compression method we must decide whether the text in hand is compressible. This raises the first questions: how can we measure the information content of a text, or how can we decide whether a message has redundancy?

Shannon investigated the idea of *entropy* to answer the above questions. Suppose we have a first-order source F_n, which emits the characters of alphabet A_n with probability $p_i, i = 1, \ldots, n$. Suppose that we have coded the symbols of alphabet A_n by the—not necessarily binary—code K_n. Then Shannon introduced the following.

DEFINITION 7. $H(F_n) = -k\sum_{i=1}^{n} p_i \log p_i = k\sum_{i=1}^{n} p_i \log \frac{1}{p_i}$ is the entropy of the source F_n, where k is a positive integer governing the units in which we measure the entropy. Usually, the units are bits, where k = 1, and so—for binary coding—the entropy is $H(F_n) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$.

Shannon proved that the function of entropy satisfies the following requirements, and he also demonstrated that it is the only function which does so:

- $H(F_n)$ is a continuous function of the probabilities $p_1, \ldots, p_n$.
- In the case of uniform distributions if $n_1 > n_2$ then $H(F_{n_1}) > H(F_{n_2})$.
- If S_1 and S_2 are two stages with probability p_1 and p_2 respectively, and F_n^1 and F_n^2 are the sources belonging to the stages, then $H(F_n) = pH(F_n^1) + p_2H(F_n^2)$.

From the above definition it is clear that more likely messages that appear with greater probability contain less informations; i.e., the more surprising the message, the more information content it has.

Now, to measure the goodness of code K_n we introduce the following.

DEFINITION 8. For the source F_n the expression $L(K_n) = \sum_{i=1}^{n} p_i l_i$ is the cost of the code K_n.

As we showed above, the entropy applies only to a probability distribution, and it gives us the average information content of a given text emitted by a source F_n. However, we are often more interested in qualifying the entropy of a particular choice from the text. Supposing that the probability of this event

is p_i, we can get the desired entropy using the definition $H_i(F_n) = \log \frac{1}{p_i}$. Now, making n subsequent decisions with probabilities $p_1, \ldots, p_n$ we get that the average entropy of the earlier processed decisions is $H(F_i) = \sum_{i=1}^{n} p_i H_i(F_n) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$, which is the overall entropy.

Now, performing a compression step by step we will get an "entropy-like" coding process if after having compressed the text for each individual choice of an arbitrary string (or a letter) its compressed length $t_i = \log \frac{1}{p_i}$. So, the only question remaining is to decide whether this result can be beaten.

DEFINITION 9. A uniquely decipherable code K_n^0 is optimal for the source F_n, if for any uniquely decipherable code K_n for F_n, $L(K_n) \geq L(K_n^0)$.

Our aim is to find such a code, which is optimal, i.e., has a minimal cost, considering the given probabilities. (Of course we would like to assure the recoverability using uniquely decipherable codes.) The next theorems present that we can answer the above raised question negatively.

THEOREM 2 [1]. *For an arbitrary uniquely decipherable code K_n for F_n, the relation $L(K_n) \geq H(F_n)$ holds.*

Abramson also proved that there exists a type of coding that produces a very good compression. In [1] he introduced the following.

DEFINITION 10. We say that a code-word $\alpha_i = \omega_1\omega_2 \cdots \omega_{l_i}$ is a prefix of another $\alpha_j = \omega_1'\omega_2' \cdots \omega_{l_j}'$ if $l_i \leq l_j$ and there exists a $l_s \leq l_i$, such that $\omega_k = \omega_k'$, if $k \leq l_s$. A code K_n is a prefix, if no code-word is a real prefix of another.

The prefix codes have great importance, since it is easy to see that if we use a prefix code, then the decoding process can be done uniquely.

THEOREM 3 [1]. *Each prefix code is uniquely decipherable.*

THEOREM 4 [1]. *For an arbitrary source F_n there exists an optimal prefix code.*

By Theorem 4 there exists an optimal prefix code for a given source F_n. The following—so-called Noiseless Coding Theorem—gives an interesting upper bound on the cost of such a code.

THEOREM 5 [60]. *Let K_n^0 be the optimal prefix code for a given source F_n. Then the relation $L(K_n^0) \leq H(F_n) + 1$ holds.*

From Theorems 2 and 5 it follows that the cost of an optimal prefix code is very close to the entropy of the given source. So, this theorem shows us that an entropy-like coding process gives us an optimal coding method. So, the next question is in hand: can we produce such a code?

Because of the above theorem, as a first step of our investigations we can concentrate on the existence of prefix codes. Taking into account the following theorem we can get two important results: On the one hand, the constraint gives us an exact bound for constructing a prefix code. On the other hand, if we have a code in our hand we can decide easily whether it is a prefix one.

THEOREM 6 [44]. *For any sequence $l_1, \ldots, l_n$ of natural numbers, there is a prefix code K_n with code-words of length $l_1, \ldots, l_n$ if and only if*

$$\sum_{i=1}^{n} 2^{-l_i} \leq 1. \tag{1}$$

Inequality (1) is known as the Kraft Inequality.

B. Models

As we saw earlier, the entropy of a given text depends on the probabilities of the symbols being coded. Similarly, the average length of the compressed text (or as we called it, the cost of a code) is also a function of the probabilities. It is also obvious that we can state certain models to compute the probabilities, and in different models these values would be different. Since the entropy serves a lower bound for the average length of the compressed text it is important to use the adequate model in our process.

The fundamental role of a model is to supply probabilities for the message. In a model—usually—we do not calculate the probabilities in one step. Instead, they are built up incrementally, starting at the beginning of the text, and processing through it character by character. Depending on this criterion we can distinguish between static and adaptive models. In the following we will show the most popular models used in connection with the data compression.

1. Static Models

In the case of a static model we calculate explicitly the probabilities in advance, we fix them, and we supply them for each example. So, we need a preprocessing step before starting the compression itself, and it supposes a deeper—and longer—statistical investigation for the structure of the text being later compressed. The deeper the investigation is, the better the model is.

Finite-Context Models

One of the simplest models for counting the probabilities is what allocates a fixed probability to each character irrespective of its position in the text. To do this we need to know all characters that may occur in a message. It is so in the case of different natural languages. A collection of American English text is known as the Brown corpus, and it has been widely used in investigating different language statistics. It based on 500 separate 2000-word samples of natural language text representing a wide range of styles and authors. The alphabet of the corpus contains 94 symbols. The first 15 most frequent characters is shown by Table 1. By analyzing a given English text one can estimate for example the probability of the word "ofor" (here the symbol "o" means the space character) as follows: One can find the probabilities of each character in Table 1, which are o = 0.1741, f = 0.0176, o = 0.059, and r = 0.0474. So the probability of the entire text-fragment is

$$0.1741 \times 0.0176 \times 0.059 \times 0.0474 = 4.515 \times 10^{-5} = 2^{-14.435},$$

TABLE I Letter Statistics from the Brown Corpus*

Letter	Prob. (%)	Letter	Prob. (%)	Letter	Prob. (%)
o	17.41	i	5.51	l	3.19
e	9.76	n	5.50	d	3.05
t	7.01	s	4.94	c	2.30
a	6.15	r	4.77	u	2.10
o	5.90	h	4.15	m	1.87

*Ref. 10. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

so the optimum coding of the string in hand is 14.435 bits in this simple model. In this model we did not take into account the preceding characters while having to calculate the probabilities. In this case we refer to the model as an *order-0 model.* It is easy to recognize that the probability of a character will differ depending on the symbols that precede it. So, we can generalize the above idea: we will say that a model is an *order-k* model if it considers k preceding characters while determining the probability of the next symbol. If we want to use an order-k model we will have some difficulties at the beginning of the text: there will be an insufficient number of characters to supply our formula. In that case some ad hoc solution is used: either we assume that the text is primed with a k-length sequence of default characters or we use an order-0 model for the first $k-1$ characters.

Models of up to order 11 have been reported in the book of [10]. It is a natural feeling that we must use a high-order model to get good estimations for the probabilities. This is true but it is also easy to see that the number of possible contexts increases exponentially with the order of the model. Thus larger and larger samples are needed to make the estimates and so, we may need a huge amount of memory to store them. This will destroy all of the advantages of any compressing procedure.

Markov Models

Natural languages are good examples to use with a finite-context model, but there are other strings for which this type of model is not usable. The most relevant example for this type of string is DNA, in which four type of bases (A,C,G, or T) can occur in triplets. The probability of one of the bases occurring within a triplet is strongly influenced by its position. Unfortunately, any knowledge about the previous base or even a number of previous bases does not help much to estimate the probabilities, and a position can only be decided by counting back to the beginning of the DNA sequence.

Final-state probabilistic models are good tools for handling such type of strings. Because of the probabilistic theory background these models are often referred to as Markov models. For a given alphabet A_n one can imagine such a model as a finite-state machine with n different states denoted by S_i, $i = 1, 2, \ldots, n$. In each state we can give a set of transition probabilities p_{ij}, which gives the transition probability from the current state S_i to the state S_j. (It is clear that $\sum_{j=1}^{n} p_{ij} = 1$.) Using this model for a given string we get a unique

path through the model, and the probability of any string can be computed by multiplying the probabilities out of each state.

Markov models are very useful for modeling natural languages, and they are widely used for this purpose.

Some of the state models are *nonergodic* in the sense that in these models there are states from which parts of the system are permanently inaccessible. All state models used in connection with natural languages are ergodic.

Grammar Models

For those types of strings which represent a part of any language with strong syntactic rules, the above models are not usable. For example, in computer algebra (like MAPLE or MATHEMATICA) if the parentheses are balanced in a formula, the next character may not be a")". Similarly, for those of computer languages where the delimiter is a special character (e.g., "." or ";") this character may not occur "inside" a statement. For the above type of strings with—theoretically—infinite nesting possibilities, the so-called grammar models are adequate. In a grammar model we have well-defined productions (e.g., conditional or unconditional jump statements and subroutine calling sequences), and we can decide the probabilities of each subproduction in a given production. This gives us a multilevel hierarchy, containing in each level a discrete probability distribution. Now, if we have a string to be modeled by a grammar model it is parsed according to the grammar, and its probability may be computed by multiplying together the probability of each production that is used.

The use of grammars—principally for compressing Pascal programs—has been explored by Katajainen *et al.* [39] and Cameron [14]. Compressing strings from a formal—computer—language, this type of model is very successful, but it is almost usable for natural languages. It follows from pragmatic reasons that it is almost impossible to find a grammar for natural languages: their syntactic rules are very complicated and are not modelable "by hand." Constructing them mechanically from samples is also impossible because we cannot decide the exact boundary of the given language. It is worth mentioning that some heuristics are useful for modeling natural languages.

The Problem of Conditioning Classes

As we have seen, in a finite-context model there is a possibility of using an order-k model to count more precisely the probability of certain parts of a given text. Unfortunately, to get a good estimation a lot of different conditional probabilities must be considered. This problem was solved first by Rissanen and Langdon in [56], by introducing the *conditional classes*. Before giving a formal description we need to introduce some notations: a substring of length $k-i+1$ in a given text $S = s_1 s_2 \dots s_n$ from the position i to the position k will be denoted by $s_i s_{i+1} \dots s_k$. The probability of a text S is denoted as usual by $p(S)$, and $p'(S)$ is the probability that the string S is the prefix of any message. If Λ is the empty message then it is easy to see that

$$p(S) = p(s_1 s_2 \dots s_n) = \frac{p'(s_1)}{p'(\Lambda)} \frac{p'(s_1 s_2)}{p'(s_1)} \cdots \frac{p'(s_1 s_2 \dots s_n)}{p'(s_1 s_2 \dots s_{n-1})} \frac{p(s_1 s_2 \dots s_n)}{p'(s_1 s_2 \dots s_n)}.$$

If we realize that each fraction is a conditional probability, i.e.,

$$\frac{p'(s_1 s_2 \ldots s_i)}{p'(s_1 s_2 \ldots s_{i-1})} = p(s_i | s_1 s_2 \ldots s_{i-1}),$$

then the probability of the entire message is reduced to a sequence of choices, one for each symbol in the text. This reformulation can help only if we can calculate the conditional probabilities easier than the overall probability of the string. Usually, this is not the situation, but we can approximate the conditional probabilities. A common way to do that is to ignore a part of the message seen so far. For example, if we consider the last two characters instead of counting the whole formula then the probability $p(s_i|s_1 s_2 \ldots s_{i-1})$ can be substituted with $p''(s_i|s_{i-2}s_{i-1})$, where this last probability means that the character s_i follows the symbols s_{i-2} and s_{i-1} in this order. This simplified formula results in an approximation for the right probability, so the overall message cannot always be compressed optimally, but the estimation will be satisfactory in most cases. As Rissanen and Langdon pointed out in [56] this "two character estimation" can be generalized if we assign each possible context string to some conditioning class. So, we can use the estimation

$$p(s_i|s_1 s_2 \ldots s_{i-1}) \approx p''(s_i|U(s_1 s_2 \ldots s_{i-1})),$$

where $U(s_1 s_2 \ldots s_{i-1})$ is the *conditioning class* of the string $s_1 s_2 \ldots s_{i-1}$. It is worth mentioning that—despite the fact that any computable function can be used to calculate the conditioning classes—in practice the above type of estimation is common; i.e., a finite number of symbols can be considered to be formulating a class. The interesting reader can find a good survey for the techniques in [4].

2. Adaptive Models

In the case of static models—based on some theoretical background—we produce a model in advance, and we supply it for any string having been generated from a given alphabet. In a static model we need to know the entire text being compressed in advance; otherwise we are not able to count the relative frequencies of the characters. *Adaptive models* have many advantages since usually they start with no statistical information and the model is built up from this initial state. So, we do not have to spend a lot of time checking whether the sender and the receiver have the same initial model, and we can use an adaptive model for any type of text. This flexibility is very important in applications of adaptive models where there is no possibility of seeing ahead of time what is likely to be transmitted.

Since the adaptive models do not know the message in advance we expect that they behave worse than the static ones. One can suppose that this is the price we need to pay for being on-line. On the other hand one can say that the adaptive model will fit step by step to the actual situation, and therefore an adaptive coding scheme must be better than a nonadaptive one. There is no general theorem that proves this latter conjecture, but there are nice theorems that state the opposite: an adaptive model will be only slightly worse than any static model:

THEOREM 7 [10]. *Given a set of conditioning classes with K members, an alphabet with q symbols, and a message of length n, there are adaptive models that will take at most Kq*log *n bits more than the best possible nonadaptive model.*

We remind the reader that in the case of semi-adaptive models the model must be sent with the message, and coding the model requires approximately $Kq \log n$ bits, so about the same number of bits will be used as for a fully adaptive model. Intuitively, it is also expected that for an adaptive model a "well-suited" message will be excellent compressible. However, what is the worst-case situation?

THEOREM 8 [10]. *There is an adaptive model where the compressed string will be at most* $(q + 1)/2 \log n$ *bits longer than the original string.*

After this surprising result one can say that an adaptive model may not be a good choice for processing a compression. Fortunately, there is another theorem that states that for any text there exists a static model that can expand it arbitrarily. So the consequence is nice:

COROLLARY 9. *There is a message where an adaptive code will do asymptotically better than the static model.*

So we can prefer an adaptive model in practice. However, there are two problems using this model that we need to face. The first is the *zero-frequency problem.* This is rooted in the fact that every time a character is encoded, all symbols that can possibly occur at that point must be predicted with a nonnegative probability, and this must be valid, even those symbols that have never occurred before the actual position of the coding. (If a symbol would have a zero probability then its code-length would be –log *0*, which is infinite.) This problem can be solved if we assign a very small probability to each nonoccurring symbol, and—at the same time—we adjust the other probabilities. Of course, this will deform the distribution and so, the compression may not be optimal.

The second problem is connected with counting the *conditioning classes.* In the literature there are two contrasting approaches. In the first one while estimating the probabilities one counts as much as possible attempts to minimize the statistical errors. If we apply the second method then we try to define a lot of conditioning classes that may result in a model in which the probabilities will accurately reflect the current context. A detailed discussion of these approaches for different adaptive models can be found in [10].

III. STATISTICAL CODING

Compression is usually made by coding. Character coding—as a special kind of coding—was defined in the previous section. Generally a code is a mapping from some source strings to some code words. Depending on the sizes of the source words and the code-words, codings can be categorized into different classes. If the lengths of all source words are equal and the lengths of all code words

are also constant—where the two lengths are not necessarily the same—we say that the coding is block-to-block type.

EXAMPLE 1. A block-to-block coding is as follows:

Source words	Code words
a	011
b	010
c	110
d	111

Similarly we can speak about block-to-variable, variable-to-block, and variable-to-variable codings. In each coding on the variable side the length of the codes can be varied. In the case of variable-to-block coding the length of the code words are fixed, but the lengths of the source words can be different. In variable-to-variable coding both of them can vary.

EXAMPLE 2. A variable-to-variable coding is as follows:

Source words	Code words
ab	11
b	010
cbd	101
dc	011

Statistical compression methods are based on the information theoretical results mentioned in the previous section. These algorithms try to give such a uniquely decipherable code, which has an optimal cost or its cost is close to the optimal. By Theorem 4 there exists such a code. By Theorem 2 the codes belonging to the symbols should be chosen in such a way that their lengths should be close to the logarithm of the reciprocal value of the probability of the symbol. If the relative frequency of the characters of the source string can be computed, then the optimal code lengths could be derived. In practice the problem is that the logarithm values are usually not integers. Many algorithms were developed to solve this problem. The most popular ones are Shannon–Fano coding [27], Huffman coding [35], and Arithmetic coding [32,54,55]. These methods always assume that the probabilities of the symbols are given. In practice the easiest method for evaluating the probabilities is to calculate the relative frequencies. So, usually these methods are not on-line in the sense that before using the coding technique we need to count the relative frequencies in a preprocessing step, and so we need to know the whole string in advance. In the following we present the most widely used statistical compression methods.

A. Shannon–Fano Coding

This technique was independently discovered in the late 1940s by C. E. Shannon and R. M. Fano. Shannon–Fano coding [27] is a block-to-variable coding, where code words are assigned to the symbols of a given alphabet. The algorithm divides recursively the symbols of the alphabet into some groups until each group contains only one symbol. During the division a 0 or 1 digit is ordered to each group, which will form the code. It is easy to see that this process

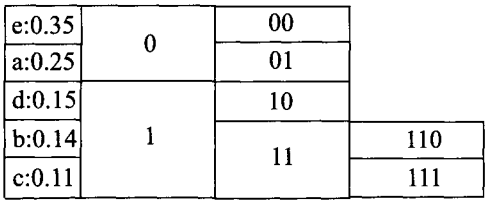


FIGURE 1 Shannon–Fano code of Example 3.

always produces a prefix code. An exact description of the procedure is the following:

- Suppose the probabilities of the symbols are given.
- List the symbols in decreasing order due to their probabilities.
- Divide the set of symbols into two parts such that each part has an equal or approximately equal probability.
- Assign to the code of the first part a 0 bit and to the code of the second part a 1 bit.
- Continue the division of both parts recursively until each subdivision contains only one symbol.
- The code of each symbol will be a series of 0's and 1's assigned to it.

EXAMPLE 3. Suppose we have five symbols in our alphabet. These are a, b, c, d, e. The probabilities of the symbols in a source are given in the table

Symbols	c	b	d	a	e
Probabilities	0.11	0.14	0.15	0.25	0.35

The construction of Shannon–Fano code for this example is represented by Fig. 1.

Shannon–Fano coding was the first attempt to solve the optimal statistical coding problem. Because the dividing step is not defined clearly, its optimality does not hold. However, in practice Shannon–Fano code is not too far from the optimum. As we saw above, this coding scheme strives to give high-probability characters short codes and low-probability ones longer codes. However, in fact, Shannon–Fano coding sometimes assigns a longer code to a more probable character than it does to a less probable one. The following coding does not suffer from this deficiency.

B. Huffman Coding

Huffman coding is also a block-to-variable coding technique. It lists the symbols in increasing probabilities, and gathers them into some sets recursively, assigning to each set the sum of the probabilities of the symbol in the set. At each step a bit is appended to the code of the symbols in the set. The algorithm can be implemented using a tree data structure. The exact method is the following:

- Suppose the probabilities of the symbols are given.
- List the symbols in increasing order due to their probabilities.
- Consider the symbols with the two least probabilities.

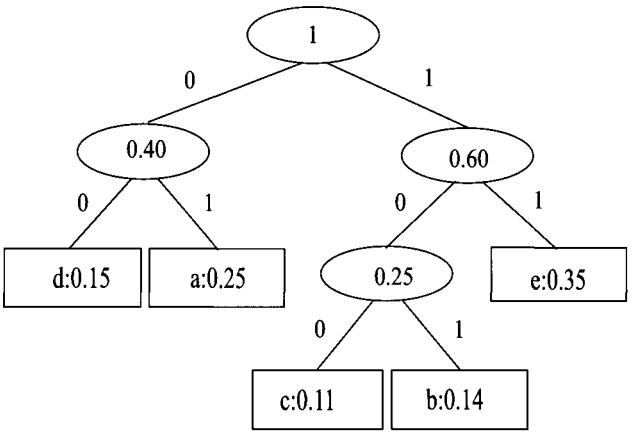


FIGURE 2 Huffman tree of Example 3.

- Substitute these symbols with a set, whose probability is the sum of the two probabilities. We order a bit of value 0 to the code of the first symbol and a bit of value 1 to the other.
- For these newly constrained sets repeat the previous three steps until we get a list containing only one element.

Using the above algorithm the following Huffman code can be constructed for Example 3. The symbols and the sets are represented by a tree, which is a usual technique in the implementations of the method (see Fig. 2). The code words for Example 3 are

Symbols	*a*	*b*	*c*	*d*	*e*
Code words	01	101	100	00	11

Huffman coding is one of the most popular compression methods. Many authors investigated the characteristics of the method from several points of view: Karp [38], Krause [45], and later Cot [22] and Mehlhorn [49] considered the case where the characters of the code alphabet have nonequal costs. Gilbert investigated the construction of Huffman codes based on inaccurate source probabilities [30].

In honor of the twenty-fifth anniversary of Huffman coding Gallager presented an interesting property of the Huffman trees [29]. This property is the so-called sibling property.

DEFINITION 11. A binary code tree has the sibling property if each node (except the root) has a sibling and if the nodes can be listed in order of nonincreasing probability with each node being adjacent in the list to its sibling.

The next theorem helps to decide whether a binary prefix code is a Huffman code.

THEOREM 10 [29]. *A binary prefix code is a Huffman code if and only if the code tree has the sibling property.*

Another interesting problem is to determine the maximum length of the Huffman codes. This problem was investigated by Buro [13]. He gave an upper

bound on the maximum length of a binary Huffman code and on the sum of the length of all code words. In the following we present some of his results.

DEFINITION 12. The external path length of a Huffman tree is the sum of all path lengths from the root to the leaves.

THEOREM 11 [13]. *Assume that we have a source alphabet consisting of* $n > 1$ *symbols. Denote the probabilities of the symbols by* $p_1, \ldots, p_n$ *and suppose that* $p_i \leq p_{i+1}$ *for all* $1 \leq i \leq n-1$. *Denote the length of the longest code word of the corresponding Huffman code by L. Then*

$$L \leq \min\left\{\left\lfloor \log_\Phi\left(\frac{\Phi+1}{p_1\Phi+p_2}\right)\right\rfloor, n-1\right\},$$

where $\Phi = \frac{1+\sqrt{5}}{2}$.

COROLLARY 12 [13]. *The external path length of a Huffman tree is at most*

$$\min\left\{\left\lfloor \log_\Phi\left(\frac{\Phi+1}{p_1\Phi+p_2}\right)\right\rfloor \cdot n, \frac{(n+2)(n-1)}{2}\right\}.$$

Because of the construction none of the assigned code words are a prefix of another in the Huffman code. So the derived code is a prefix and Theorem 3 holds. As a consequence decoding can be done easily: we search for the first matching code word in the compressed data, then we substitute it with the corresponding symbol, and we continue this process until the end of the compressed file.

The main problem with decoding is that storing the Huffman tree may require large memory. In the original paper the array data structure was used to implement the complete binary tree. Later Hashemian [34] presented an efficient decoding algorithm using a special array data structure. In the following we present a memory-efficient array data structure to represent the Huffman tree. This was published by K.-L. Chung in [19]. The memory requirement of this representation is $2n$–3 where n is the size of the basic alphabet. Based on this implementation a fast decoding is possible. The following algorithm creates the array structure H:

- Traverse the Huffman tree in a preorder way. For each left edge record the number of edges E and leaf nodes L in the subtree of that edge. Assign the $E + L + 1$ value to the node whose left edge has been investigated.
- Traverse again the Huffman tree in a preorder way. At each time emit the assigned value when a left edge is encountered, or a "1" when a right edge is encountered. Emit the source symbol when a leaf edge is encountered.
- Save the ordered emitted values in H.

EXAMPLE 4. For the Huffman tree in Example 3, H is the following:

$$H: 2, e, 1, 5, 2, c, 1, b, 1, 2, d, 1, a.$$

Now we present the Chung decoding algorithm [19]. The algorithm uses two pointers. One pointer points to the Huffman code C, the other to the array H. The pointers are denoted by *cp* and *ap*, respectively. In the code *len*(C) denotes

the length of the corresponding Huffman code C. Using the construction of the H array, the algorithm moves the pointers in such a way that if cp points to the end of a code word in C then ap points exactly to the corresponding source symbol in the H array. Using this process, decoding the next symbol can be done easily outputting $H[ap]$. The pseudo-code looks like this:

```
ap := 1; cp := 1;
while cp < len(C) do
    begin
    if H[cp] = 0 then
        begin
        cp := cp + 1;
        ap := ap + 1;
        end
    else
        begin
        cp := cp + 1;
        ap := H[ap] + 1;
        if not eof (C) then
            begin
            if H[cp] = 1 then
                begin
                cp := cp + 1;
                ap := H[ap] + 1;
                end
            else
                begin
                cp := cp + 1;
                ap := ap + 1;
                end
            end
        end
    Output H[ap]
    end
```

There are other approaches in the practice: in [50] a VLSI implementation of Huffman encoding and decoding was considered. Teng [62] presents a $O(\log(n)^2)$ parallel algorithm for constructing a Huffman tree with n leaves. Some years later in [46] an off-line construction was investigated for those types of Huffman codes where the depth of leaves is limited.

C. Redundancy of Huffman Codes

It is well known that the Huffman code is an optimal prefix code; i.e., its cost is minimal among the prefix codes of the given alphabet and probabilities. The question is how close is the code to the entropy of the given source? The difference between the entropy and the cost of the Huffman code is characterized by its *redundancy*. Given that each character in an alphabet must occupy an integral number of bits in the encoding, Huffman coding achieves a

"minimum redundancy"; i.e., it performs optimally if all probabilities are exact powers of $\frac{1}{2}$. Unfortunately, this is not the case usually in practice. Huffman codes—like the Shannon–Fano process—can take up one extra bit for each character.

DEFINITION 13. Let F be a given source and denote the Huffman code for F by K_H. Then the value

$$R = H(F) - L(K_H)$$

is the redundancy of the Huffman code.

When the Noiseless Channel Theorem is used, it immediately follows that $R \leq 1$ for all sources. Many authors investigated the problem of finding tight upper bounds for R. The first results were given by Gallager [29].

THEOREM 13 [29]. *Let p_1 be the probability of the most likely letter in a source. Then*

$$R \leq p_1 + \sigma,$$

where $\sigma = 1 - \log_2 e + \log_2(\log_2 e) \approx 0.086$.

THEOREM 14 [29]. *Let p_1 be the probability of the most likely letter in a source. If $p_1 \geq 0.5$, then*

$$R \leq 2 - \kappa(p_1) - p_1,$$

where κ is the binary entropy function, i.e.,

$$\kappa(x) = -x \log_2 (x) - (1 - x) \log_2 (1 - x).$$

The above bound is tight.

Later Johnsen [37], Capocelli *et al.* [15], and Capocelli and de Santis [16] proved some further results on the redundancy.

THEOREM 15 [37]. *Let p_1 be the probability of the most likely letter in a source. If $0.5 > p_1 \geq 0.4$, then*

$$R \leq \begin{cases} 1 + 0.5\,(1 - p_1) - \kappa(p_1) & \text{if } 0.4 \leq p_1 \leq \delta \\ 3 - 5p_1 - \kappa(2p_1) & \text{if } \delta \leq p_1 \leq 0.5, \end{cases}$$

where $\delta \approx 0.4505$. *The above bounds are tight.*

THEOREM 16 [15]. *Let p_1 be the probability of the most likely letter in a source. If $\delta > p_1 \geq \frac{1}{3}$, then*

$$R \leq 1 + 0.5\,(1 - p_1) - \kappa(p_1).$$

The above bound is tight.

THEOREM 17 [16]. *Let p_1 be the probability of the most likely letter in a source. Then*

$$R \leq \begin{cases} \frac{3}{4} + \frac{5p_1}{4} - \kappa(p_1) & \text{if } \frac{2}{9} < p_1 < \theta \\ 3 - (3 + 3\log 3)p_1 - \kappa(3p_1) & \text{if } \theta \leq p_1 \leq \frac{1}{3}, \end{cases}$$

where $\theta \approx 0.3138$. *The above bounds are tight.*

THEOREM 18 [16]. *Let p_1 be the probability of the most likely letter in a source. If for some $l \geq 3$*

$$\frac{2}{2^{l+1}+1} < p_1 < \frac{1}{2^l - 1},$$

then

$$R \leq \begin{cases} 3 - \kappa(p_1) - (1-p_1)(2\log 3 - \frac{5}{9}) & \text{if } \frac{2}{17} < p_1 < \gamma \\ 4 - 7(1+\log 7)p_1 - \kappa(7p_1) & \text{if } \gamma < p_1 < \frac{1}{7} \\ 1 - \kappa(p_1) - (1-p_1)A_{2^l-1} & \text{if } l \geq 4, \end{cases}$$

where $\gamma \approx 0.1422$, $A_j = \min_{w_j \in W_j} \{H(W_j) - w_j\}$ and $W_j, j \geq 2$ is the set of positive real numbers $w_1, w_2, \ldots, w_j$ that satisfy $w_1 \geq w_2 \geq \cdots \geq w_j \geq \frac{w_1}{2}$ along with $\sum_h w_h = 1$. The above bounds are tight.

The above theorems cover all the cases when we know the largest symbol probability. There are some other results on the redundancy of Huffman codes with some other assumptions. For example we assume that we know the largest and the smallest probabilities. Details can be found in [15,17]. In addition to the redundancy, another similar notion was introduced in the literature. This is the *maximum data expansion.*

DEFINITION 14. Assume we are given a code $K = \{\alpha_1, \ldots, \alpha_n\}$ and denote $l_1, \ldots, l_n$ as the lengths of the $\alpha_1, \ldots, \alpha_n$ code words. Let the probabilities of the characters of the alphabet be $p_1, \ldots, p_n$. The maximum data expansion $\delta(K)$ of the code K is defined as

$$\delta(K) = \sum_{\{i | l_i > \log n\}} (l_i - \log n) p_i.$$

As presented before, Huffman coding uses two phases to compress a source word. First it calculates the relative frequencies that approach the symbol probabilities. Then the compression algorithm substitutes each symbol with its code word. During this phase the size of the file may grow temporarily when the symbols with low probabilities are placed at the beginning of the file. The maximum data expansion gives some estimation on the ratio of this growth. Its notion was introduced by Cheng *et al.* [18]. It is an interesting question to give some upper bound on $\delta(K)$. The most recently known best result is due to De Prisco and De Santis [23].

THEOREM 19 [23]. *The maximum data expansion $\delta(K_H)$ of a Huffman code is bounded by*

$$\delta(K_H) < 1.39.$$

It is not known, whether the above bound is tight.

D. Arithmetic Coding

Beyond Huffman coding, arithmetic coding is one of the most known statistical coding algorithms (see [32,54,55]). This method represents the input text by a number between 0 and 1. The subsequent symbols divide a previously defined interval due to their probabilities. The symbols with small probability decrease

the length of the interval more significantly, giving more bits to the representation, while the symbols with higher probability decrease less the length of the interval.

The algorithm works in the following way:

- Let the starting interval be [0,1].
- Divide the corresponding interval due to the probabilities of the symbols.
- Consider the next symbol of the text and take the corresponding subinterval as the next interval.
- Repeat the previous two steps until we reach the end of the text. An arbitrary number from the last interval can represent the text.

EXAMPLE 5. Consider again the symbols and probabilities given in Example 3. Suppose that the input string is *deec*. The change of the intervals is the following:

first	[0, 1)
after *d*	[0.60, 0.75)
after *e*	[0.60, 0.6525)
after *e*	[0.60, 0.618375)
after *c*	[0.61635375, 0.618375)

Figure 3 shows another representation of the coding process of Example 5. The greatest problem of the algorithm is that as the length of the text increases, the required accuracy of the number belonging to it increases, using more and more bits.

Decoding is also easy in this case. We consider the compressed form and the representing number. We divide the starting interval [0, 1) due to the probabilities. The first symbol of the source will be that character whose interval contains the number. Then we take the subinterval of this character and repeat the process until we decode the last character of the source.

In the implementation of the arithmetic coding usually integer arithmetic is used. To avoid overflow the bits that are not necessary for further coding should be sent to the output immediately. A C language implementation of arithmetic coding can be found in [10,70].

It is a well-known fact that arithmetic coding is optimal; i.e., its coding result is arbitrarily close to the entropy. For Huffman coding, the redundancy

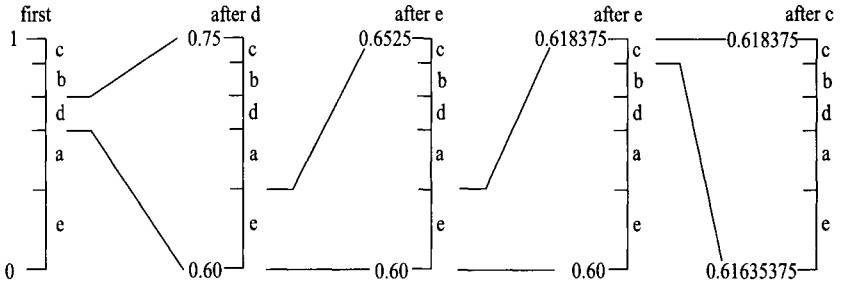


FIGURE 3 Arithmetic coding process for Example 5.

TABLE 2 Compression of Natural Language Alphabets by Huffman and Arithmetic Coding [12]*

Language	Huffman	Arithmetic	Huffman cost (%)
English	4.1854	4.1603	0.6
Finnish	4.0448	4.0134	0.8
French	4.0003	4.0376	0.9
German	4.1475	4.1129	0.8
Hebrew	4.2851	4.249	0.8
Italian	4.0000	3.9725	0.7
Portuguese	4.0100	3.9864	0.6
Spanish	4.0469	4.0230	0.6
Russian	4.4704	4.4425	0.6
English-2	7.4446	7.4158	0.4
Hebrew-2	8.0370	8.0085	0.4

*With permission from Springer-Verlag.

bounds show that this is not always true. So for a given example, it is a natural question as to whether arithmetic coding is superior to Huffman coding. Bookstein and Klein investigated this problem by comparing the two methods from several points of view [12]. They stated the most important advantages and disadvantages of arithmetic coding relative to Huffman codes. The advantages are optimality, efficient encoding if the alphabet or its characteristics are changing over the file, and simple extensibility to infinite alphabet. The disadvantages are slowness, complexity, and only small savings in realistic situations. Some other comparisons were based on the size of the alphabet. For large alphabets it can be stated that in general the probability of the most frequent character is close to 0, so the cost of the Huffman code is close to the entropy and the difference between the two methods is not significant. Bookstein and Klein compared the algorithms for natural language alphabets. Here we present their results in Table 2. The first column contains the name of the language, the next two columns give the average code-word lengths for the two methods, and the fourth gives the increase of the Huffman value over the value for arithmetic coding in percent.

Unfortunately, Huffman coding can work poorly for small alphabets. For example, in the pathological case when we have a binary alphabet with probabilities ε and $1 - \varepsilon$. Then the length of each code word is 1 bit with Huffman coding. Arithmetic coding will reach the entropy, which is $\varepsilon \log_2 \frac{1}{\varepsilon} + (1 - \varepsilon) \log_2 \frac{1}{1-\varepsilon}$. The entropy tends to 0 if $\varepsilon \to 0$. It shows that Huffman coding is a poor choice in such cases. It is also stated in [12] that arithmetic coding gives better compression in case of inaccurate probabilities. Time comparisons present that arithmetic coding consumes much more time than Huffman coding.

Some variants of arithmetic coding appeared in the literature too. A modification was proposed by Teuhola and Raita [63]. They found some disadvantages of arithmetic coding, like nonpartial decodability, vulnerability, i.e., strong effect of small (for example, one-bit) errors, etc. To avoid these problems, they

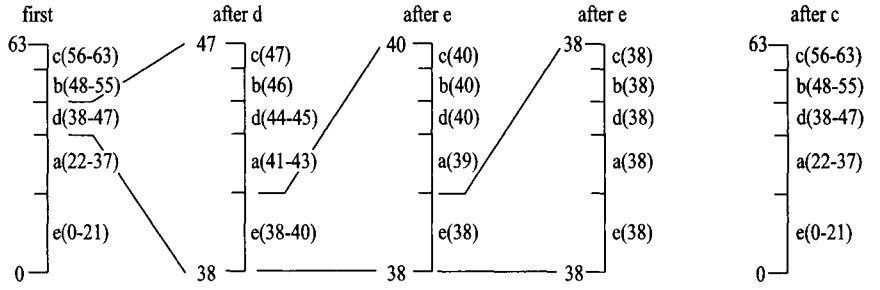


FIGURE 4 Coding steps of Example 3 using the fixed-length arithmetic algorithm.

proposed a modified version of arithmetic coding that produces fixed-length code words. The idea of their algorithm is to apply arithmetic coding repeatedly to some substrings, so that each substring produces a fixed-length code word. The technique is similar to the original arithmetic coding. However instead of the interval [0, 1) the range $[0, 2^n - 1]$ is considered, where n is a fixed integer. A code word will be an integer from this range. The algorithm works the same way as arithmetic coding: first it divides the range $[0, 2^n - 1]$ into some parts in accordance with the symbol probabilities. Then it chooses the subinterval corresponding to the next symbol. This technique is repeated recursively. If the range gets so small that the current symbol does not fall into a unique range, then the code is ready.

EXAMPLE 6. Take again the symbols and probabilities in Example 3. Let $n = 8$, so the code range is [0,63]. Suppose we have the same source word *deec*. Figure 4 shows the coding steps. The corresponding codewords can be 38 and 60.

The main difference between this fixed-length arithmetic coding and the original one is that in this case some collisions happen, i.e., some symbols can fall into the same range after the division, if the original range is narrow. Because this collision information is not utilized, the algorithm would not be optimal. To avoid this problem, the modified version of the algorithm use this information. This means that if a collision happens, both the encoder and the decoder know the symbols that are taking part in the collision. In the next step the algorithm reduces the alphabet to these symbols. This can be done, because in the case of a collision the next symbol is always restricted to one of the symbols of the collisions. The main disadvantage of this algorithm is that the successive code words are not independent and not separately decodable. The experimental results in [63] showed that these variants of arithmetic coding are rather practical and their redundancy is very small for some longer code words.

At the end of this section we refer back to the modeling step. Those models that have been introduced are suitable for supplying probabilities to such types of encoder like the Huffman code or the Arithmetic code. In the later case a coder is able to compress strings in that number of bits that are indicated by the entropy of the model being used, and this is the minimum size possible for the model chosen. Unfortunately, in the case of Huffman coding this lower bound can be achieved only under special conditions. This implies the importance of

modeling and underlines the two-phase process that we need to follow in a compression procedure.

E. Adaptive Techniques

If we consider a generic model for a natural language, or pick up an order-0 statistic for any binary code and we compare these statistics to a particular message we can easily realize that the general models rarely represent these examples exactly. Earlier, we mentioned those adaptive techniques which we can use to fit the model dynamically to the message to be transmitted. The main idea for using the adaptive techniques is to change the symbol frequencies so far in the message. At the beginning the frequencies are equal, and they must be updated as the subsequent symbol arrives to approximate the observed frequencies. Of course both the encoder and the decoder start with the same initial values and they must use the same update algorithm. This ensures us that the model remains in step: getting the next symbol the encoder encodes it and fits the frequencies, and the decoder first identifies the character according to the actual model and then it also updates the model.

It is easy to see that each of the above static models can be changed to use an adaptive technique. When using a particular coding procedure such as the Huffman or Arithmetic coding the most exciting task is always to find the best implementation and to use the most adequate data structure for the technique being used. In the next part of the chapter we will review some known adaptive techniques.

1. Adaptive Huffman Coding

It is evident to proceed with Huffman coding adaptively by recalculating the Huffman tree step by step. However, it is also easy to see that such a procedure cannot be very efficient since updating the frequencies for an alphabet that contains q different characters can take–in the worst case–$q \log q$ operations, and it is very expensive. So, certain approaches for solving this problem more effectively have been investigated:

A. The *incremental-method* was introduced by Faller [26] and Gallager [29] independently. We can succintly describe the method as follows: getting the subsequent character in the message we increment the count of the appropriate node. This can result in a rearrangement of the tree. In the simplest case it is merely an exchange of two nodes in the tree with incrementations of their parents', grandparents', etc. node counts to the root. Unfortunately, sometimes it is not enough, and the update operation may require rearranging further nodes' positions too. This can be implemented effectively by rippling up the tree and/or swapping some nodes as appropriate.

B. The *aging technique* is based on the fact that if we increment a node count, then it may overflow. So, we attempt to keep the counts within an interval, keeping in mind that our statistic must remain relevant to the portion of text being encoded. To avoid the overflow, the simplest technique is to halve all counts, but we can rescale the counts by any constant less than 1 (see [21,43]). The advantage of this method is that the relative frequencies

stay the same, so we do not need to reorganize the Huffman tree. Different aging techniques are considered in [10]. We may have certain problems with this technique:

- The larger the time constant for aging, the slower the adaptation of the model, which may result in a better estimate for slowly varying statistics, but it can be irrelevant for rapidly varying statistics.
- Rescaling can create fractional parts, and the incremental update tree algorithm cannot handle these type of counts. So we need either to round or to truncate them to the nearest integer, which can change the structure of the Huffman tree dramatically.

Improvements and generalizations were considered in [20,43,64]. Finally, Lelewer and Hirschberg [47] summarized the improvements proposed by Vitter [64]. Weyland and Pucket [67] considered an adaptive technique for Huffman-like trees for compression of a Gaussian source of fixed and floating point numbers.

At the end of this subsection we must mention that the key of this technique is always the recreation of the Huffman tree. Since it may be time-consuming the resulting algorithm could be unsatisfactory for on-line coding.

2. Adaptive Arithmetic Coding

Using the Arithmetic coding we have counts for each symbol, and so we can count the total as well. Normalizing the counts we can get the relative frequencies of the symbols. It is easy to see that such a model can be updated simply by incrementing the count of the subsequent character. The only problem is the implementation. We must distinguish techniques developed for binary alphabets from those effective for large alphabets. For binary alphabets the first method was developed by Rissanen and Langton [56]. Efficient implementations for adaptive arithmetic coding both for binary and large alphabets are discussed in [10]. Further implementation can be find in [36,51] for large alphabets. It is interesting that there are experimental results for measuring the efficiency of some implementations; for example in [56] it has been pointed out that the method examined method has a very good efficiency (about 98.5%). In [70] the authors gave an implementation for the adaptive arithmetic coding, and a detailed discussion of its performance was given.

IV. DICTIONARY CODING

The basic idea of textual substitution or dictionary coding is to substitute the subsequent word fragments by a pointer or an index to a given dictionary. A *dictionary* is a finite set of ordered pairs (*source word, code word*), where the source word is a string from a finite alphabet and the code words are used to substitute the corresponding parts of the source text with the code word belonging to the matching source word. The dictionary coding methods can be divided into three groups. These are static, semiadaptive, and adaptive methods.

A. Methods Using a Static Dictionary

A static coding method uses always the same (static) dictionary given in advance, independently from the source text. The disadvantage of this model is based on the fact that if the dictionary does not fit the source text then we can get an extremely wrong compression ratio. On the other hand, if the dictionary contains too many words or word fragments then its size becomes too large, resulting in problems in both storing and searching. Static dictionaries are definitely useful in those cases when we must compress records of a database and some words (or word fragments) occur repeatedly. For example when we would like to compress the records of a library catalog containing as the words authors, titles, ISBN-s, books, etc. Supposing that we use a static dictionary; we can start compression with any record of the database. In the next part of the chapter we deal with these methods in detail.

1. Further Definitions

As was proved by Schuegraf and Heaps [58] compressing a given source string optimally is equivalent to the problem of finding a *shortest path* in a related directed, edge-weighted graph. For a source string $S = s_1 s_2 \ldots s_n$ we define a graph $N = (V, A)$ on the vertex set $V = \{v_0, v_1, \ldots, s_n\}$. The edge $(v_i, v_{i+d}) \in A$ exists iff there exists a pair of codes—(source word, code word)—such that the source word contains d characters that exactly match the original source string in positions $i + 1, \ldots, i + d$. The weight of this edge is the number of bits in the corresponding code word. It is easy to see that a shortest path from v_0 to v_n in the graph N corresponds to an optimal compression of the source string S.

EXAMPLE 7. To illustrate the above graph take the following string and dictionary

$$S = THIS\_IS\_AN\_EXAMPLE!$$

Source word	A	E	H	I	L	M	N	P	S
Code word	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
Weight	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*

Source word	T	X	!	IS	_EX	XAM	THI	MPLE	MPLE!
Code word	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*
Weight	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*

The corresponding graph of Example 7 can be seen in Fig. 5.

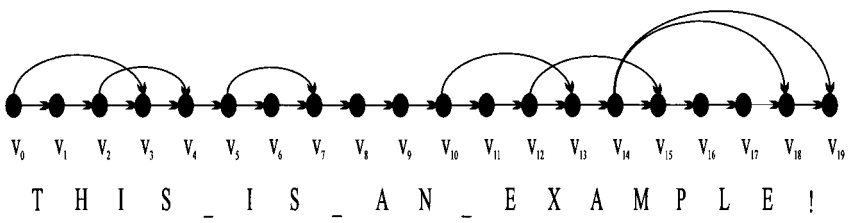


FIGURE 5 The edge-weighted graph of Example 7.

If we suppose that the weights of the edges are equal, then the shortest path from v_0 to v_{19} is the path

$$v_0v_3v_4v_5v_7v_8v_9v_{10}v_{13}v_{14}v_{19}.$$

Using the above model the solution of the problem becomes easy, since we can apply one of the shortest-path algorithms for a directed, weighted graph. These algorithms run always in polynomial time. If the corresponding graph has many *cut vertices* (i.e., vertices that divide the original problem into independent subproblems) and if these subproblems are reasonably small, we can indeed solve the problem efficiently and can compute the *optimal* encoding. Unfortunately, in practice this will not be the case and a shortest-path algorithm cannot be applied since it takes too much time and space to compute an *optimal* solution for very long strings. Similarly, we have difficulties in case of *on-line* compression, where we must compress a source string block by block (where a block is a segment of the given string). Therefore, *heuristics* have been developed to derive near optimal solutions.

The earlier developed heuristics (for example, the *longest fragment first heuristic* (*LFF*) cf. Schuegraf and Heaps [59]) have not been deeply analyzed and only experimental results on their performance have been reported. Later, when the worst-case analysis became more popular these—and other newly created—algorithms were analyzed also from a worst-case point of view. Most of these algorithms are *on-line*. An *on-line data compression algorithm* starts at the source vertex v_0, examines all outgoing edges, and chooses one of them according to some given rule. Then the algorithm continues this procedure from the vertex reached via the chosen edge. There is no possibility of either undoing a decision made at an earlier time or backtracking.

Of course, usually an on-line heuristic will generate only a suboptimal compression. One possibility for measuring the "goodness" of an algorithm is to analyze its *worst-case behavior*. This is generally measured by an *asymptotic worst-case ratio*, which is defined as follows: Let $D = \{(w_i, c_i) : i = 1, \ldots, k\}$ be a static dictionary and consider an arbitrary data compression algorithm A. Let $A(D, S)$, respectively $OPT(D, S)$, denote the compressed string produced by algorithm A, respectively, the optimal encoding for a given source string S. The length of these codings will be denoted by $\|A(D, S)\|$, respectively $\|OPT(D, S)\|$. Then the *asymptotic worst-case* ratio of algorithm A is defined as

$$R_A(D) = \lim_{n\to\infty} \sup \left\{ \frac{\|A(D, S)\|}{\|OPT(D, S)\|} : S \in S(n) \right\},$$

where $S(n)$ is the set of all text strings containing exactly n characters.

The first worst-case analysis for an on-line data compression method was performed by Katajainen and Raita [41]. They analyzed two simple on-line heuristics, the *longest matching* and the *differential greedy* algorithm, which will be defined exactly later.

Four parameters have been used in the literature to investigate the asymptotic worst-case ratios:

$$Bt(S) = \text{length of each symbol of the source string } S \text{ in bits}$$

$$lmax(D) = \max\{|w_i| \mid i = 1, \ldots, k\}$$

$$cmin(D) = \min\{\|c_i\| \, i = 1, \ldots, k\}$$
$$cmax(D) = \max\{\|c_i\| \, i = 1, \ldots, k\},$$

where $|w_i|$ denotes the length of a string w_i in characters and $\|c_i\|$ the length of a code word c_i in bits. If the meaning is clear from the context, we will simply denote the bit length of each input character by Bt and also omit the reference to the dictionary by using *lmax* instead of *lmax(D)*.

Not surprisingly, the worst-case behavior of a heuristic strongly depends on the features of the available dictionary. The following types of dictionaries have been examined in different papers:
A dictionary is called *general* if it contains all of the symbols of the input alphabet as source words (this ensures that every heuristic will in any case reach the sink of the underlying graph and thus will terminate the encoding with a feasible solution). In this paper we will only deal with general dictionaries. A general dictionary is referred to as follows:

1. **code-uniform** dictionary, if all code words are of equal length (i.e., $\|c_i\| = \|c_j\|, 1 \leq i, j \leq k$),
2. **nonlengthening** dictionary, if the length of any code word never exceeds the length of the corresponding source word (i.e., $\|c_i\| \leq |w_i|Bt, 1 \leq i \leq k$),
3. **suffix** dictionary, if with every source word w also all of its proper suffixes are source words (i.e., if $w = \omega_1\omega_2 \ldots \omega_q$ is a source word $\Rightarrow$ $\omega_h\omega_{h+1} \ldots \omega_q$ is a source word for all $2 \leq h \leq q$), and
4. **prefix** dictionary, if with every source word w also all of its proper prefixes are source words i.e., if $w = \omega_1\omega_2 \ldots \omega_q$ is a source word $\Rightarrow$ $\omega_1\omega_2 \ldots \omega_h$ is a source word for all $1 \leq h \leq q - 1$).

2. Optimal and Approximation Algorithms

Consider again now the original problem, i.e., when we want to find an optimal encoding of a source text assuming that we have a static dictionary. There were known results for solving this problem even at the beginning of the 1970s [57,65,66]. As was mentioned before, Schuegraf and Heaps [58] showed that this question is equivalent to the problem of finding a *shortest path* in a related directed edge-weighted graph.

Later Katajainen and Raita [40] presented an optimal algorithm, which can be considered as a refinement of the above methods. Their algorithm is based on the general shortest-path method, but it uses cut vertices to divide the problem into smaller parts. The authors used the graph model to describe the steps of the algorithm. The dictionary is stored in an extended trie. A trie is a multiway tree, where each path from the root to a node represents an element from the dictionary. The extended trie was introduced by Aho and Corasick [3], and it allows fast string matching. In an extended trie each node contains a special pointer—called a failure transition—which points to the node whose associated string is the longest proper suffix of the string associated to the given node. Using these pointers a linked list is ordered to each node, and this list contains the nodes associated to the string itself and to those strings of the dictionary that are proper suffixes of it. Figure 6 illustrates the extended trie of

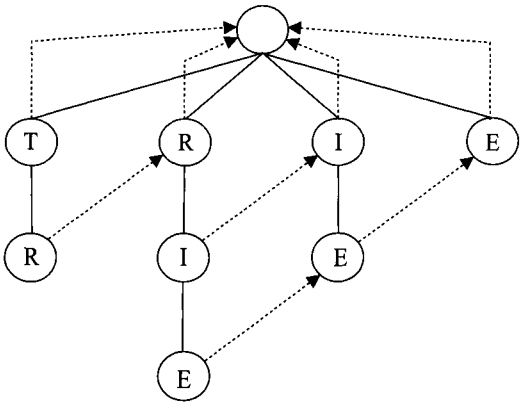


FIGURE 6 An extended trie. The dashed lines represent failure transition pointers.

the dictionary given as follows:

Source word	T	R	I	E	IE	TR	RIE
Code word	*a*	*b*	*c*	*d*	*e*	*f*	*g*

Using the notations given in the Introduction, a formal description of the algorithm is the following:

Let $S = s_1 s_2 \ldots s_n$ be a source string and $D = \{(w_i, c_i) : i = 1, \ldots, k\}$ a static dictionary;
Create an extended trie and add the dictionary strings $w_1, w_2, \ldots, w_k$ to it;

Create an empty output buffer B;
$d(v_0) := 0$; $p(v_0) := 0$; $cp := 0$;
for each character s_j in S **do**
begin
Using the extended trie find the set I of indices which defines those dictionary strings which match with the original text and end at position j;
$d(v_0) := \infty$.;
for each index i in I **do**
begin
$p := |w_i|$; $q := \|c_i\|$;
if $d(v_j) > d(v_{j-p}) + q$ **then** $d(v_j) := d(v_{j-p}) + q$; $p(v_j) := i$;
end
if $j - lmax > cp$ **then**
begin
put $p(v_{j-lmax})$ to B;
if v_{j-lmax} is a cut vertex **then**
begin
Traverse the shortest path in the buffer B
Encode the contents of B
Reset B
$cp := j - lmax$;
end
end

```
for j := lmax − 1 downto 0 do
    put p(v_{n−j}) to B;
Traverse the shortest path in the buffer B
Encode the contents of B
end
```

It is easy to see that the d and p arrays contain the set of the last $lmax$ distance and parent values. The distance value is always the actual value of the shortest path and the parent pointer points to the previous node in the shortest path. The processed parent values are always sent to the output buffer. Unfortunately the necessary size of the output buffer is not known before the algorithm starts. The algorithm checks whether we have a cut vertex. If this occurs, then the contents of the buffer can be encoded. Unfortunately it is possible that there is no cut vertex at all. In this case the size of the output buffer can be arbitrary large. Of course this happens rarely in practice, but theoretically it is possible.

Raita and Katajainen showed that the time complexity of this shortest-path algorithm is $O(n+m)$, while the total time complexity of the extended trie construction and processing is $O(Wc \log_2 c + n \log_2 c + m)$ if it is implemented as a binary tree. Here W denotes the sum of the lengths of the dictionary source words, c is the maximal number of children of a node, n is the number of vertices and m is the number of edges of the graph.

The authors also analyzed the effect of a fixed-length output buffer on the compression ratio. They proved the following theorems, assuming that the length of the output buffer is $|B|$ and this approximation algorithm is denoted by A.

THEOREM 20 [40]. *Let D be a* **general** *dictionary. Then*

$$R_A(D) \leq 1 + \frac{lmax(lmax-1)}{|B|}\frac{cmax}{cmin}.$$

THEOREM 21 [40]. *Let D be a* **code-uniform** *dictionary. Then*

$$R_A(D) \leq 1 + \frac{lmax(lmax-1)}{|B|}.$$

THEOREM 22 [40]. *Let D be a* **nonlengthening** *dictionary. Then*

$$R_A(D) \leq \begin{cases} 1 + \dfrac{lmax\,(lmax-1)}{|B|}\dfrac{Bt}{cmin} & \text{if } cmin \leq Bt < cmax \\ 1 + \dfrac{lmax\,(lmax-1)}{|B|}\dfrac{cmax}{cmin} & \text{if } cmin < cmax \leq Bt. \end{cases}$$

THEOREM 23 [40]. *Let D be a* **suffix** *dictionary. Then*

$$R_A(D) \leq 1 + \frac{lmax}{|B|}\frac{cmax}{cmin}.$$

In Table 3 we summarize the results for this approximation algorithm.

TABLE 3 Summary of the Results on the Behavior of the Approximation Algorithms *A*

⟨Dictionary D⟩			
Suffix	**CU**	**NL**	$R_A(D)$
—	—	—	$1+\frac{lmax\,(lmax-1)}{\lvert B\rvert}\frac{cmax}{cmin}$
x	—	—	$1+\frac{lmax}{\lvert B\rvert}\frac{cmax}{cmin}$
—	x	—	$1+\frac{lmax\,(lmax-1)}{\lvert B\rvert}$
—	—	x	$1+\frac{lmax\,(lmax-1)}{\lvert B\rvert}\frac{\min(Bt,cmax)}{cmin}$
x	x	—	$1+\frac{lmax}{\lvert B\rvert}$
x	—	x	$1+\frac{lmax}{\lvert B\rvert}\frac{cmax}{cmin}$
—	x	x	$1+\frac{lmax(lmax-1)}{\lvert B\rvert}$
x	x	x	$1+\frac{lmax}{\lvert B\rvert}$

3. On-line Heuristic Algorithms

In this section we review the most important on-line heuristics and compare them from worst-case points of view. First of all we give the exact definitions.

The **longest matching** heuristic LM chooses at each vertex of the underlying graph the *longest* outgoing arc, i.e., the arc corresponding to the encoding of the longest substring starting at the current position. Ties can be broken arbitrary.

Katajainen and Raita [41] analyzed the worst–case behavior of LM for dictionaries that are code uniform/nonlengthening/suffix and they derived tight bounds for all eight combinations of these properties.

The *greedy* heuristic, which we will call **differential greedy** DG (introduced by Gonzalez–Smith and Storer [31]) chooses at each position the arc implied by the dictionary entry (w_i, c_i) yielding the maximal "local compression," i.e., the arc maximizing $|w_i|Bt - \|c_i\|$. Ties are broken arbitrarily. The **fractional greedy algorithm** (introduced by Békési *et al.* [7]) takes at any actual position in the source string the fractionally best possible local compression, i.e., if I is the set of indices of the arcs emanating from the current node of the corresponding graph then an arc i_0 will be chosen such that

$$i_0 = \arg\min_{i\in I} \frac{\|c_i\|}{|w_i|Bt}.$$

Obviously, although each heuristic is locally optimal, globally they can give a rather poor result. On the other hand it is possible that some heuristic compresses optimally such inputs, for which another gives the worst possible result and reverse.

It is intuitively clear that in many cases greedy type heuristics will perform better than the LM heuristic, which does not care about code lengths at all. There are also differences between the greedy methods. This is illustrated by the following example (let $a^1 = a, a^{i+1} = aa^i, i \in \mathbb{N}$, for any character a):

EXAMPLE 8. Let us consider the following nonlengthening dictionary with $cmax = 4$, $cmin = 1$ and, as usual for ASCII encodings, $Bt = 8$.

Source word	u	v	uv	$v^{lmax-1}u$
Code word	10	1101	1100	0
Weight	2	4	4	1

Compressing the source string $S_i = u(v^{lmax-1}u)^i$ consisting of $8(lmax.i + 1)$ bits with the LM or the DG algorithm in both cases yields the code string $(1100(1101)^{lmax-2})^i 10$ with $4(lmax - 1)i + 2$ bits. Applying the FG heuristic to the same problem generates the code string $10(0)^i$, which is only $i + 2$ bits long.

Although Example 8 is based on a very special dictionary, it demonstrates the differences between the algorithms. The following theorems give a short summary of the most important results for the different heuristics.

THEOREM 24 [7,41]. *For a* **general** *dictionary D*

$$R_{LM}(D) = R_{FG}(D) = (lmax - 1)\frac{cmax}{cmin},$$

$$R_{DG}(D) = \begin{cases} \dfrac{cmin + (lmax - 1)cmax}{cmin + (lmax - 1)Bt} & \text{if } (lmax - 1)^2 cmax\, Bt \le cmin^2 \\ & \text{and } \left\lfloor \dfrac{cmax - cmin}{Bt} \right\rfloor \ge lmax - 1 \\ \dfrac{(lmax - 1)cmax}{cmin} & \text{otherwise.} \end{cases}$$

The above result presents that from worst-case point of view the three methods work very similar. Only the ratio of the DG algorithm differs a little in a special case.

Another interesting problem may be to analyze the heuristics for different types of dictionaries. First we will investigate suffix dictionaries and present the most important results.

THEOREM 25 [6,7,41]. *Let D be a* **suffix** *dictionary. Then*

$$R_{LM}(D) = \frac{cmax}{cmin},$$

$$R_{DG}(D) = \begin{cases} \dfrac{cmin + (lmax - 1)cmax}{cmin + (lmax - 1)Bt} & \text{if } (lmax - 1)^2 cmax\, Bt < cmin^2 \\ & \text{and} \lfloor (cmax - cmin)/Bt \rfloor \ge lmax - 1 \\ \dfrac{(lmax - 1)cmax}{cmin} & \text{otherwise,} \end{cases}$$

$$R_{FG}(D) \le \frac{cmax(ln(lmax - 1) + 1)}{cmin}$$

and there exists a suffix dictionary D_0, for which

$$\frac{cmax(ln(lmax - 1) + 1 - ln\, 2)}{cmin} < R_{FG}(D_0).$$

What is interesting for suffix dictionaries is that LM works much better than the greedy algorithms. If the dictionary is also nonlengthening, then LM is even optimal. DG can reach the worst possible ratio, while FG is somewhere between LM and DG. Prefix property is similar to the suffix one; therefore one would expect similar results. The following theorems shows that this is not the case.

THEOREM 26 [6,7]. *Let D be a* **prefix** *dictionary. Then*

$$R_{\mathrm{LM}}(D) = R_{\mathrm{FG}}(D) = \frac{(lmax - 1)cmax}{cmin},$$

$$R_{\mathrm{DG}}(D) = \begin{cases} \dfrac{cmin + (lmax - 1)cmax}{cmin + (lmax - 1)Bt} & \text{if } (lmax - 1)^2 cmax\, Bt \leq cmin^2 \\ & \text{and } \left\lfloor \dfrac{cmax - cmin}{Bt} \right\rfloor \geq lmax - 1, \\ \dfrac{(lmax - 1)cmax}{cmin} & \text{otherwise.} \end{cases}$$

We can conclude that prefix property does not help at all in the worst case, because our results are the same as those in the general case. Finally we compare the three heuristics for nonlengthening dictionaries. This case may be interesting, because all heuristics give the same worst-case ratio.

THEOREM 27 [6,7,41]. *Let D be a* **nonlengthening** *dictionary and S be a string. Then $R_{\mathrm{LM}}(D) = R_{\mathrm{DG}}(D) = R_{\mathrm{FG}}(D)$, and*

$$R_{\mathrm{LM}}(D) = \begin{cases} \dfrac{(lmax - 1)cmax}{cmin} & \text{if } cmax \leq Bt \\ \dfrac{(lmax - 2)Bt + cmax}{cmin} & \text{if } Bt < cmax < 2Bt \\ \dfrac{(lmax\ Bt}{cmin} & \text{if } 2Bt \leq cmax. \end{cases}$$

4. Almost On-line Heuristics

More than twenty years ago Shuegraf and Heaps [59] introduced the *longest fragment first (LFF)* heuristic. They supposed that the file we want to compress is divided into records of the same lengths. The idea is the following: the longest word fragment—within the actual record—which matches a word from the dictionary is chosen and encoded. Then the overlapping matches are eliminated and the algorithm works repeatedly.

Later Stauffer and Hirschberg [61] presented the so-called LFF parsing algorithm, which was based on the above idea. However they described the algorithm for a parallel architecture. In this model a processor is assigned to each character of the input string and these processors can work in parallel. First each processor calculates the list of the lengths of matches between the dictionary and the input string beginning at its position. Then the algorithm determines the maximum match length. Starting from this match length, the algorithm goes down to length 1, and in each step it finds the maximum

collection of nonoverlapping matches of the given length. These matches will be selected for coding. Finally the algorithm eliminates all the matches overlapping the selected ones. A formal description of the algorithm is the following:

```
Compute the list of the length of matches at each position of the source
string S.
Compute the maximum match length, L
for l := L downto 1 do
  begin
  Find a maximum collection C of nonoverlapping matches of length l
  Select the elements of C for coding
  Eliminate the elements overlapping the matchings of C from the match
  lists
  end
```

Of course this parallel algorithm can be implemented on a normal computer architecture too. The only problem is that this algorithm is an off-line one in this form, since we must know the total source string to compute the list of match lengths. To avoid this problem Nagumo *et al.*, defined the on-line version of the above algorithm [52], which can be effective for conventional computer architectures. They use a lookahead of length $\frac{1}{2}(lmax-1)(lmax-2)+lmax$. The basis of the algorithm is the same as that of the Stauffer and Hirschberg algorithm, but they use a modified selection and elimination procedure. Their algorithm is the following:

```
p := 1;
while p ≤ Length(S) do
    begin
    for l := 1 to lmax do
        begin
        let d_l be the distance of the closest match of length l to p;
        if d_l > ½(lmax − 1)(lmax − 2) + lmax then d_l := ∞;
        end
    l := lmax;
    t := d_l;
    while d_l > 0 do
        begin
        if d_l + (l − 1) − 1 ≥ t then d_{l−1} := ∞;
        else t := d_{l−1};
        l := l − 1;
        end
    encode substring S_i ... S_{i+l−1};
    p := p + l;
    end
```

The proof of the correctness of the algorithm is given in [52].

A variation of the above-mentioned longest fragment first parsing algorithm is when the remaining part of the record is compressed with an on-line algorithm (the LM heuristic can be chosen, for example). If the file has no record structure, we can consider a buffer, which always contains the actual part of the text (so

we will talk about buffers instead of records). Now, before reading the next part of the file into the buffer, the algorithm always encodes the whole buffer. We will refer to a whole buffer-coding procedure as a *step*. We will denote this algorithm by $\mathrm{LFF_{LM}}$. To avoid double indexing, instead of $R_{\mathrm{LFF_{LM}}}(D)$ we write $R_{\mathrm{LFF}}(\mathrm{LM}, \mathrm{D})$.

As one can see algorithm $\mathrm{LFF_{LM}}$ gives up the strict on-line property, because it looks ahead in the buffer, getting more information about its content. One has a feeling that the more information about the text to be compressed, the better worst-case behavior of a good algorithm. This suggests to us that this algorithm behaves better than the on-line ones. The experimental results showed [59] that the LFF-type algorithms give a compression ratio better than that of the longest matching or the differential greedy heuristics. In the paper [8] we presented some theoretical results on the behavior of the algorithm $\mathrm{LFF_{LM}}$. Here we mention the most important theorems.

THEOREM 28 [8]. *Let D be a* **general** *dictionary. Then*

$$R_{\mathrm{LFF}}(\mathrm{LM}, D) = \frac{(t-1)lmax - (t-3)}{t}\frac{cmax}{cmin}.$$

THEOREM 29 [8]. *Let D be a* **nonlengthening** *dictionary. Then*

$$R_{\mathrm{LFF}}(\mathrm{LM}, D) = \begin{cases} \dfrac{(t-1)lmax - (t-3)}{t}\dfrac{cmax}{cmin} & \text{if } cmax \le Bt \\ T & \text{if } Bt < cmax < 2Bt \\ \dfrac{(t-1)lmax\, Bt + cmax}{t\, cmin} & \text{if } 2Bt \le cmax, \end{cases}$$

where

$$T = \frac{(t-1)lmax\, Bt - t(2Bt - cmax) + 4Bt - cmax}{t\, cmin}.$$

THEOREM 30 [8]. *Let D be a* **prefix** *dictionary. Then*

$$R_{\mathrm{LFF}}(\mathrm{LM}, D) = \frac{(t-1)lmax - (t-3)}{t}\frac{cmax}{cmin}.$$

THEOREM 31 [8]. *Let D be a* **suffix** *dictionary. Then*

$$R_{\mathrm{LFF}}(\mathrm{LM}, D) = \begin{cases} \left(1 + \dfrac{2(lmax-1)}{t}\right)\dfrac{cmax}{cmin} & \text{if } t \ge 3 \\ \left(\dfrac{lmax+1}{2}\right)\dfrac{cmax}{cmin} & \text{if } t = 2. \end{cases}$$

B. Adaptive Methods

1. The LZ77 Algorithm

This algorithm maintains a window and a lookahead buffer. The window contains some characters backwards, i.e., the last coded characters. The lookeahed buffer contains some characters to be coded. The pointers of the LZ77 method point to the occurrences of the substrings from the lookeahed buffer to the occurrences of the same substring in the window. Since it is possible

that only one character match can be found, the output can contain individual characters too. Denote the length of the lookeahed buffer by L_S. The algorithm can work on a buffer of length n, which contains always a part of the source text. The length of the window is $n - L_S$. In the current step we find the longest substring in the windows that match for the lookahead buffer starting from the beginning. The two matching substrings can overlap, but they cannot be the same. This match is coded by a triplet $\langle i, j, a \rangle$, where i is the index of the found substring in the window, j is its length, and a is the first character that has not matched. Then we move the buffer right on the text by $j + 1$ characters and continue the process. Putting the character a to *(i, j)* ensures the working of the algorithm in the case when we have no match at all.

More formally the algorithm works as follows. First we introduce some notations. Let n be the length of the applied buffer, A is the basic alphabet, and S is the source string. As mentioned before denote by L_S the length of the window and by n the length of the buffer, and let $L_C = 1 + \lceil \log(n - L_S) \rceil + \lceil \log(L_S) \rceil$, where the basis of the logarithm is $|A|$. L_C means the fixed length of the codes, which are created from alphabet A too. Let $S(1, j)$ be a real prefix of the string S, and let i, $1 \leq i \leq j$ a given integer. Let $L(i) = \max \{l : S(i, i + l - 1) = S(j + 1, j + l)\}$, and $L(p) = \max_{1 \leq i \leq j} L(i)$. We refer to the $S(j + 1, j + L(p))$ string as a reproducible extension of $S(1, j)$ into S. It can be seen that $S(j + 1, j + L(p))$ is the longest substring among the matching substrings of S beginning in $S(1, j)$.

EXAMPLE 9. Let $S = 01101101$ and $j = 4$. Then $L(1) = 0$, $L(2) = 4$, $L(3) = 1$, $L(4) = 0$. So $S(4 + 1, 4 + 4) = 1101$ is the reproducible extension of $S(1, 4)$ into S with $p = 2$.

The LZ77 algorithm [71] is as follows:

- Let $B_1 = 0^{n-L_S} S(1, L_S)$, where 0^{n-L_S} means the all-zero string of length $n - L_S$, and let $i = 1$.
- Consider the buffer B_i, $i \geq 1$, and let $S_i = B_i(n - L_S + 1, n - L_S + l_i)$, where the length of $l_i - 1$ prefix of S_i is the reproducible extension of $B_i(1, n - L_S)$ into $B_i(1, n - 1)$.
- Let p_i be the index of the above reproducible extension. Then the code word C_i for S_i is $C_i = C_{i1}C_{i2}C_{i3}$, where, C_{i1} and C_{i2} are $|A|$-radix representations of $p_i - 1$, and $l_i - 1$ respectively, while C_{i3} is the last symbol of S_i.
- Modify the contents of B_i that we leave the first l_i, and load the next l_i characters. Increase the value of i by 1, and continue the algorithm with Step 2.

EXAMPLE 10. Consider the following binary series ($|A| = 2$)

$$S = 001010110110111011011100$$

$$L_S = 8, n = 16 \Rightarrow L_C = 1 + \log_2(16 - 8) + \log_2 8 = 6.$$

$B_1 = 00000000\|00101011$	$p_1 = 8, l_1 = 3$	$C_1 = 111\|010\|1$
$B_2 = 00000001\|01011011$	$p_2 = 7, l_2 = 5$	$C_2 = 101\|100\|1$
$B_3 = 00010101\|10110111$	$p_3 = 4, l_3 = 4$	$C_3 = 011\|011\|1$

Decoding of the compressed texts is easy. The algorithm maintains a buffer the same way as during the encoding process. In each step a code word $C_i = C_{i1}C_{i2}C_{i3}$ is read from the input. Then the sequence is specified by C_{i1} and C_{i2}, and the character C_{i3} is sent to the output.

Ziv and Lempel proved [71] that using algorithm LZ77 we get at least as good a result as using another special adaptive dictionary, if the length of the buffer is large enough. The problem is that if we increase the length of the buffer, then searching becomes slow and the compression inefficient. We can overcome this problem by using special data structures, but this may require more storage. Detailed information can be found about this problem in [10].

2. The LZSS Algorithm

The LZSS algorithm is a slight modification of the LZ77. It was proposed by Bell [9]. It eliminates some redundant information, which is created for example in the case if there is no match in the window. The algorithm outputs a pointer if it is not longer than the length of the coded substring. Step 3 of LZ77 modifies as follows, all the other steps being the same:

- Let p_i be the index of the given reproducible extension. Consider C_{i1}, C_{i2}, C_{i3}, where C_{i1} and C_{i2} are $|A|$-radix representations of $p_i - 1$, and $l_i - 1$ respectively, while C_{i3} is the last symbol of S_i. If $|S_i| > |C_{i1}C_{i2}|$ then let $C_i = C_{i1}C_{i2}$: else let $C_i = C_{i3}$ and $l_i = 1$.

3. The LZ78 Algorithm

The LZ78 algorithm is a simple variation of the LZ77 method. It is obvious that the size of the window gives some restriction for finding matchings in the coded part of the text. LZ78 eliminates this problem, since it records all the previous matchings in an explicit dictionary. Of course this dictionary can contain many words, if we compress a large text. Fortunately there is a good data structure for making searching efficient; this is the trie. Each node of the trie contains the number of the represented dictionary item. This number is used for coding. Because of this representation, adding a new word to the dictionary requires only creating a new child from the node representing the longest match. Formally the LZ78 algorithm is as follows [72]:

- Let $B_1 = S(1, n)$, where n is the length of the buffer and let $i = 1$. Construct an empty dictionary D using the trie data structure.
- Consider the buffer $B_i, i \geq 1$, and let $S_i = B_i(1, l_i)$, where the length of $l_i - 1$ prefix of S_i is the longest prefix of $B_i(1, n-1)$ which matches a dictionary item in D. Add S_i to the dictionary D with a new node index.
- Let p_i be the index of the above matching dictionary item, or let p_i be 0 if there is no match at all. Then the code word C_i for S_i is $C_i = C_{i1}C_{i2}$, where C_{i1} is some representation of p_i, while C_{i2} is the last symbol of S_i.
- Modify the contents of B_i that we leave the first l_i, and load the next l_i characters. Increase the value of i by 1, and continue the algorithm with Step 2.

TABLE 4 Dictionary for Example 11

Dictionary	0	01	010	1	10	11	011	101	1011	100
Number	1	2	3	4	5	6	7	8	9	10
Output	0, 0	1, 1	2, 0	0, 1	4, 0	4, 1	2, 1	5, 1	8, 1	5, 0

EXAMPLE 11. Consider again the same binary series as in the case of the LZ77 algorithm

$$S = 00101011011011011011100.$$

Table 4 shows the dictionary generated by LZ77 for S. Figure 7 illustrates the dictionary trie.

Decoding of an LZ78 compressed code starts with an empty dictionary. In each step a pair of codes is read from the input. This code word refers to an existing dictionary string or if it is zero, it contains only one symbol. Then this new string is added to the dictionary in the same way as in the encoding process. This way during the decoding the dictionary changes the same way as in the case of encoding. So we get the proper source string.

4. The LZW Algorithm

LZW is a similar modification of LZ78 as LZSS is of LZ77. It was proposed by Welch [68]. The algorithm modifies as follows:

- Let $B_1 = S(1, n)$, where n is the length of the buffer and let $i = 1$. Construct a dictionary D that contains all the characters of the input alphabet using the trie data structure.
- Consider the buffer B_i, $i \geq 1$, and let $S_i = B_i(1, l_i)$, where the length of $l_i - 1$ prefix of S_i is the longest prefix of $B_i(1, n-1)$ which matches a dictionary item in D. Add S_i to the dictionary D with a new node index.

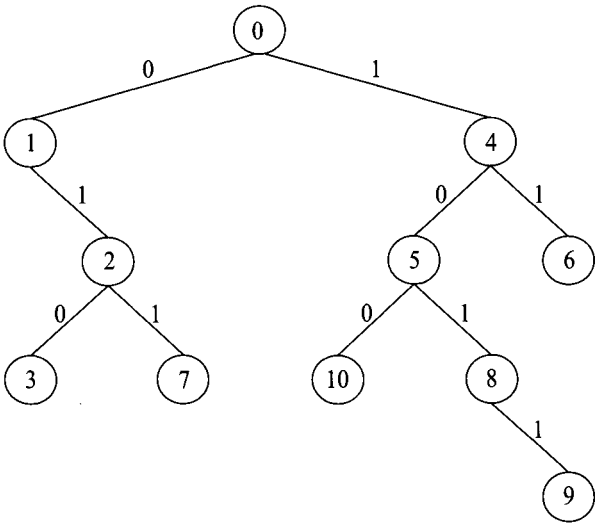


FIGURE 7 Dictionary trie of Example 11.

- Let p_i be the index of the above matching dictionary item. Then the code word C_i for S_i is some representation of p_i.
- Modify the contents of B_i that we leave the first $l_i - 1$, and load the next $l_i - 1$ characters. Increase the value of i by 1, and continue the algorithm with Step 2.

V. UNIVERSAL CODING

The Noiseless Coding Theorem gives a relation between the cost of the optimal prefix code and the entropy of the given source. It is always assumed that the probability distribution is explicitly given. Unfortunately sometimes this distribution is unknown or it is impossible to determine the characteristics of the source. A natural question is whether it is possible to find such a code that is optimal for any probability distribution instead of a particular one. This kind of code is called universal.

Many authors investigated the problem of finding universal codes [24,25,42,69]. Universal codes can be classified into two classes. Some kind of universal coding techniques are similar to statistical coding methods; i.e., they order a code to the characters of the basic alphabet. Other codes are based on dictionary technique. A "statistical" universal code was introduced by Elias [24,25]. The idea of the Elias codes is to represent each source string by integers and then order to each integer a code. This code is given explicitly and can be applied for each message. Calculating the code of an integer x consists of two phases. First we consider the binary representation of x, prefaced by $\lfloor \lg x \rfloor$ zeros. The binary value of x is expressed in the least possible bits, so it begins with 1. Therefore the prefix property is held. In the second phase this code is calculated for the integer $\lfloor \lg x \rfloor + 1$. Then the binary value of x is appended to this code without the leading 1. This way the resulting code word has length $\lfloor \lg x \rfloor + 2\lfloor \lg(1 + \lfloor \lg x \rfloor) \rfloor + 1$. Elias proved the following theorem:

THEOREM 32 [25]. *The Elias code is asymptotically optimal, i.e.,*

$$\lim_{n \to \infty} \frac{E(n)}{H(F)} = 1,$$

where E (n) is the expected code word length for a source word of length n divided by n and H (F) is the entropy of the source.

Table 5 summarizes the Elias codes of the first 8 integers.

EXAMPLE 12. Consider an alphabet that contains four symbols a, b, c, d. Let S be *bbaaadccd*. The Elias code of S can be constructed as

Symbol	Frequency	Rank	Code word
a	3	1	1
b	3	2	0100
c	2	3	0101
d	2	4	01100

The complete coded form of S is

0100010011101100010101010 1100.

TABLE 5 Elias Codes

Number	Code
1	1
2	0100
3	0101
4	01100
5	01101
6	01110
7	01111
8	00100000

A similar universal coding technique was found by Apostolico and Fraenkel [5]. The method is based on the well-known standard Fibonacci numbers, which are defined by the recurrence

$$F_0 = 1,$$
$$F_1 = 1,$$
$$F_k = F_{k-1} + F_{k-2} \text{ for } k \geq 2.$$

The series looks like this: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

It is also known that each positive integer I has a unique Fibonacci representation in the form

$$I = \sum_{i=0}^{k} d_i F_i,$$

where $d_i \in \{0, 1\}, k \leq I$.

Using this representation the Fibonacci code of the integer I can be easily defined. Simply we consider the Fibonacci representation of I with the binary symbols $d_0, d_1, \ldots, d_k$. To get the Fibonacci code of I we reserve the order of the symbols in the representation and append 1 to it. It can be seen that the code is a prefix, because it terminates with two consecutive 1's, which cannot appear anywhere else in a code word because of the property of the Fibonacci representations. Table 6 summarizes the Fibonacci codes of the first 8 integers.

EXAMPLE 13. Consider again the same alphabet and source word as in Example 12. The Fibonacci code of S can be constructed as

Symbol	Frequency	Rank	Codeword
a	3	1	11
b	3	2	011
c	2	3	0011
d	2	4	1011

The complete coded form of S is

01101111111110110011001110 11.

TABLE 6 Fibonacci Codes

Number	Fibonacci representation	Fibonacci code
1	1	11
2	10	011
3	100	0011
4	101	1011
5	1000	00011
6	1001	10011
7	1010	01011
8	10000	000011

It is also interesting to find a very simple code that still has the universal property. One of the simplistic universal code was discovered by Neuhoff and Shields [53]. The idea of the encoding is based on a simple dictionary coding technique. The dictionary is formed where the original source is divided into some blocks of a given length l. The dictionary will contain these l-blocks as source words. The code word of a given dictionary source word is its location in the dictionary using fixed-length encoding. To get universal code all possible block lengths should be investigated and the one that produces the shortest code should be chosen. Denote the length of this shortest code by L_n for a source word of length n. Neuhoff and Shields proved the following theorem.

THEOREM 33 [53]. *For any stationary, ergodic source F with alphabet A and entropy $H(F)$ the encoding rate $\frac{L_n}{n}$ converges to entropy $H(F)$ in both expected value and almost surely as $n \to \infty$.*

This theorem shows that a real simple algorithm can be universal. It is a nice theoretical result. Unfortunately it is complicated to use this algorithm in practice, because finding the optimal block length is very time consuming. Especially for long source strings—when one can get a good approximation of the entropy—many different block lengths should be tried to get the best one.

VI. SPECIAL METHODS

A. Run Length Encoding

Run length encoding is one of the simplest compression methods. It tries out some special properties of the source string. Run length encoding replaces the sequences of repeated characters by their lengths. For a given sequence a character and the length of the sequence is stored as the code. This algorithm is effective, if the data contains long sequences of repeated symbols. For example, some multimedia data like images can have these characteristics. The method is particularly useful when the size of the alphabet is small. It was shown before that in this case classical methods, like Huffman coding, can work poorly, while run length encoding can give good compression, depending on the structure of the data.

EXAMPLE 14. Consider an image whose alphabet contains only 0 and 1 values. 0 means white (w) pixel and 1 means black (b). A part of the image and its run length code would look like this:

Data	Run length code
00000000111111111110000000000001111111111	$w8b11w11b10$

B. Algorithms Based on List Update

The idea of applying list update methods for data compression originates from Bentley *et al.* [11]. They showed how a list update algorithm can be used for compression. List update or self-organizing sequential search problem is widely investigated in combinatorial optimization. We are given a set of items $x_1, \ldots, x_n$. The items are unsorted and stored in a linked list. We are also given a sequence of requests. Each request specifies an item from $x_1, \ldots, x_n$. The algorithm must serve each request by accessing the give item. This can be done with linear search, and in the case of the serving item x_i has a cost of i. After the search the requested item may be moved to another location in the list at no extra cost. This may decrease the cost of the subsequent requests. The goal is to serve the given sequence of requests at the smallest total cost. Different kinds of on-line algorithms exist to solve this problem. A well-known one is the Move to Front (MTF) method. This algorithm moves the requested item to the front of the list. Bentley *et al.* proposed the following idea to use MTF for compressing data. Both the encoder and the decoder have a list of symbols. The encoder is given a source string S of these symbols. For each character of S the encoder searches for the character in its list and outputs the position where the symbol is found. After that the encoder updates its list using MTF. When the decoder receives the compressed message, for each code i it reads the symbol at the position i, outputs it, and update the list using MTF strategy.

EXAMPLE 15. Consider an alphabet that contains four symbols a, b, c, d. Let S be *bbaaadccd*. MTF coding works as follows

Character	MTF code	List
		a, b, c, d
b	2	b, a, c, d
b	1	b, a, c, d
a	2	a, b, c, d
a	1	a, b, c, d
b	2	b, a, c, d
d	4	d, b, c, a
c	3	c, d, b, a
c	1	c, d, b, a
d	2	d, c, b, a

Bentley *et al.* showed that for arbitrary probabilities of symbols, the expected number of bits to encode one symbol of S using MTF coding is linear in the entropy of the source.

Recently Albers and Mitzenmacher [2] presented a new list update algorithm that can be applied for data compression. They called it the Timestamp ($TS(0)$) algorithm. $TS(0)$ inserts the requested item x in front of the first item

in the list that has been requested at most once since the last request to x. If x has not been requested so far, it leaves the position of x unchanged. Albers and Mitzenmacher proved that for $TS(0)$ the expected number of bits to encode one symbol of a source string is also linear in the entropy of the source, and the constant is slightly better than that of the MTF coding.

C. Special Codes and Data Types

Sometimes using some special coding can compress data. For example, English text can be coded where each character is represented by a 7-bit ASCII code. This technique is widely used to save storage. Binary coded decimal (BCD) is a way of storing integers. Using this, four bits represent each digit. This way we can store 100 different numbers in one byte. Some other similar techniques for saving space also exist, see [10] for details.

VII. CONCLUSIONS

In this chapter we gave a review of the most important data compression techniques and some issues of the theories behind them. In the first part of the chapter we presented some classical information theoretical theorems. These results give the basis of many data compression methods, especially for statistical algorithms. The methods in the chapter can be divided into four groups. These are statistical coding, dictionary coding, universal coding, and some special techniques. Statistical coding methods assume *a priori* knowledge of some statistical characteristics of the data to be compressed. This information is usually the relative frequency of the symbols of the data. The most well-known methods and the corresponding theory were presented in the chapter. Dictionary coding uses a completely different idea. In this case fragments of the data are substituted with some code words ordered to these fragments by a given dictionary. The dictionary can be static or can change dynamically during the coding process. We presented many theoretical results for different static dictionary algorithms. Classical methods were also given in the chapter. Universal coding is useful if we do not have any information on the characteristics of the source data. So it is possible that they give worse results than some statistical methods. Some simple universal coding methods were also described. Finally we gave some well-known special methods. These techniques can be efficiently applied for data with special properties.

REFERENCES

1. Abramson, N. *Information Theory and Coding*. McGraw-Hill, New York, 1963.
2. Albers, S., and Mitzenmacher, M. Average case analyses of list update algorithms with application to data compression. *Algorithmica* **21**: 312–329, 1998.
3. Aho, A. V., and Corasick, M. J. Efficient string matching: An aid to bibliographic search. *Commun. ACM* **18**(6): 333–340, 1975.
4. Angluin, D., and Smith, C. H. Inductive inference: Theory and methods. *Comput. Surveys* **15**(3): 237–269, 1983.

5. Apostolico, A., and Fraenkel, A. S. Robust transmission of unbounded strings using Fibonacci representations. Tech Rep. CS85-14, Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, 1985.
6. Békési, J., Galambos, G., Pferschy, U., and Woeginger G. J. Greedy algorithms for on-line data compression. *J. Algorithms* 25: 274–289, 1997.
7. Békési, J., Galambos, G., Pferschy, U., and Woeginger G. J. The fractional greedy algorithm for on-line data compression. *Computing* 56(1): 29–46, 1996.
8. Békési, J., Galambos, G., and Raita, T. Longest fragment first algorithms for data compression. In *New Trends in Mathematical Programming* (Eds.: F. Gianessi, T. Rapcsák. and S. Komlósi, Eds.), pp. 13–28. Kluwer Academic Dordrecht, The Netherlands, 1998.
9. Bell, T. C. Better OPM/L text compression. *IEEE Trans. Commun.* 34(12): 1176–1182, 1986.
10. Bell, T. C., Cleary, G., and Witten, I. H. *Text Compression.* Prentice–Hall, Englewood Cliffs, NJ, 1990.
11. Bentley, J. L., Sleator, D. S., Tarjan, R. E., and Wei, V. K. A locally adaptive data compression scheme. *Commun. ACM* 29: 320–330, 1986.
12. Bookstein, A., and Klein, S. T. Is Huffman coding dead? *Computing* 50(4): 279–296, 1993.
13. Buro, M. On the maximum length of Huffman codes. *Inform. Process. Lett.* 45: 219–223, 1993.
14. Cameron, R. D. Source encoding using syntactic information source models, LCCR Technical Report 86–7. Simon Fraser University, Burnaby, BC, Canada.
15. Capocelli, R. M., Giancarlo, R., and Taneja, I. J. Bounds on the redundancy of Huffman codes. *IEEE Trans. Inform. Theory* 32(6): 854–857, 1986.
16. Capocelli, R. M., and De Santis, A. Tight upper bounds on the redundancy of Huffman codes. *IEEE Trans. Inform. Theory* 35(5): 1084–1091, 1989.
17. Capocelli, R. M., and De Santis, A. New bounds on the redundancy of Huffman codes. *IEEE Trans. Inform. Theory* 37(4): 1095–1104, 1991.
18. Cheng, J.-F., Dolinar, S., Effros, M., and McEliece, R. Data expansion with Huffman codes. In *Proc. of ISIT'95*, pp. 325–332, 1995.
19. Chung, K. Efficient Huffman decoding. *Inform. Process. Lett.* 61: 97–99, 1997.
20. Cormack, G. V., and Horspool, R. N. Algorithms for adaptive Huffman. codes. *Inform. Process. Lett.* 18: 159–166, 1987.
21. Cormack, G. V., and Horspool, R. N. Data compression using dynamic Markov modelling. *Computer J.* 30: 541–550, 1987.
22. Cot, N. *Characterization and Design of Optimal Prefix Code.* Ph. D. Thesis, Computer Science Department, Stanford University, 1977.
23. De Prisco, R., and De Santis, A. A new bound for the data expansion of Huffman codes. *IEEE Trans. Inform. Theory* 43(6): 2028–2032, 1997.
24. Elias, P. Universal code word sets and representations of the integere. *IEEE Trans. Inform. Theory* 2: 194–203, 1975.
25. Elias, P. Interval and recency rank source coding: Two on-line adaptive variable-length schemes. *IEEE Trans. Inform. Theory* 33(1): 3–10, 1987.
26. Faller, N. An adaptive system for data compression. In *Conference Record of Seventh IEEE Asilomar Conference on Circuits and Systems*, pp. 593–597, 1973.
27. Fano, R. *Transmission of Information.* M.I.T. Press, Cambridge, MA, 1949.
28. Gallager, R. *Information Theory and Reliable Communication.* Wiley, New York, 1968.
29. Gallager, R. Variations on a theme by Huffman. *IEEE Trans. Inform. Theory* 24(6): 668–674, 1978.
30. Gilbert, E. N. Codes based on inaccurate source probabilities. *IEEE Trans. Inform. Theory* 17(3): 304–314, 1971.
31. Gonzalez-Smith, M., and Storer, J. Parallel algorithms for data compression. *J. Assoc. Comput. Mach.* 32: 344–373, 1985.
32. Guazzo, M. A general minimum-redundancy source-coding algorithm. *IEEE Trans. Inform. Theory* 26(1): 15–25, 1980.
33. Hamming, R. *Coding and Information Theory.* Prentice–Hall, Englewood Cliffs, NJ, 1980.
34. Hashemian, R. Memory efficient and high-speed search Huffman coding. *IEEE Trans. Commun.* 43: 2576–2581, 1995.

35. Huffman, D. A method for the construction of minimum-redundancy codes. In *Proc. Inst. Electr. Electron. Engineers* **40**(9): 1098–1101, 1952.
36. Jones, D. W. Application of splay trees to data compression. *Commun. ACM* **3**: 280–291, 1988.
37. Johnsen, O. On the redundancy of binary Huffman codes. *IEEE Trans. Inform. Theory* **26**(2): 220–222, 1980.
38. Karp, R. M. Minimum-redundancy coding for the discrete noisless channel. *IRE Trans. Inform. Theory* **7**: 27–39, 1961.
39. Katajainen, J., Pentonnen, N., and Teuhola, J. Syntax-directed compression of program files. *Software-Practice Exper.* **16**(3): 269–276, 1986.
40. Katajainen, J., and Raita, T. An approximation algorithm for space-optimal encoding of a text. *Computer J.* **32**(3): 228–237, 1989.
41. Katajainen, J., and Raita, T. An analysis of the longest matching and the greedy heuristic in text encoding. *J. Assoc. Comput. Mach.* **39**: 281–294, 1992.
42. Kieffer, J. C. A survey of the theory of source coding. *IEEE Trans. Inform. Theory* **39**: 1473–1490, 1993.
43. Knuth, D. E. Dynamic Huffman coding. *J. Algorithms* **6**: 163–180, 1985.
44. Kraft, L. G. A device for quantizing, grouping, and coding amplitude modulated pulses. M. Sc. Thesis, Department of Electrical, Engineering, MIT, Cambridge, MA, 1949.
45. Krause, R. M. Channels which transmit letters of unequal duration. *Inform. Control* **5**: 13–24, 1962.
46. Larmore, L. L., and Hirschberg, D. S. A fast algorithm for optimal length-limited codes, Technical report, Department of Information and Computer Science, University of California, Irvine, CA, 1990.
47. Lelewer, D. A., and Hirschberg, D. S. Data compression. Technical Report, 87-10, Department of Information and Computer Science. University of California, Irvine, CA, 1987.
48. Lovasz, L. Personal communication, 1995.
49. Mehlhorn, K. An efficient algorithm for constructing nearly optimal prefix codes. *IEEE Trans. Inform. Theory* **26**(5): 513–517, 1980.
50. Mukherjee, A., and Bassiouni, M. A. On-the fly Algorithms for data compresion. In *Proc. ACM/IEEE Fall Joint Computer Conference.* 1987.
51. Moffat, A. A data structure for arithmetic encoding on large alphabets. In *Proc. 11th. Australian Computer Science Conference.* Brisbane, Australia, pp. 309–317.
52. Nagumo, H., Lu, M., and Watson, K. On-line longest fragment first parsing algorithm. *Inform. Process. Lett.* **59**: 91–96, 1996.
53. Neuhoff, D. L., and Shields, P. C. Simplistic universal coding. *IEEE Trans. Inform. Theory* **44**(2): 778–781, 1998.
54. Rissanen, J. J. Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Develop.* **20**(3): 198–203, 1976.
55. Rissanen, J. J., and Langdon, G. G. Arithmetic coding. *IBM J. Res. Develop.* **23**(2): 149–162, 1979.
56. Rissanen, J. J., and Langdon, G. G. Universal modeling and coding. *IEEE Trans. Inform. Theory* **27**(1): 12–23, 1981.
57. Rubin, F. Experiments in text file compression. *Commun. ACM* **19**(11): 617–623, 1976.
58. Shuegraf, E. J., and Heaps, H. S. Selection of equifrequent word fragments for information retrieval. *Inform. Storage Retrieval* **9**: 697–711, 1973.
59. Shuegraf, E. J., and Heaps, H. S. A comparision of algorithms for database compression by use of fragments as language elements. *Inform. Storage Retrieval* **10**: 309–319, 1974.
60. Shannon, C. E. A mathematical theory of communication. *Bell System Tech. J.* **27**: 398–403, 1948.
61. Stauffer, L. M., and Hirschberg, D. S. PRAM algorithms for static dictionary compression. In *Proc. 8th International Parallel Processing Symposium,* pp. 344–348, 1994.
62. Teng, S. H. The construction of Huffman-equevalent prefix code in NC. *ACM SIGACT News* **18**: 54–61, 1987.
63. Teuhola, J., and Raita, T. Arithmetic coding into fixed-length code-words. *IEEE Trans. Inform. Theory* **40**(1): 219–223, 1994.

64. Vitter, J. S. Design and analysis of dynamic Huffman coding. *J. Assoc. Comput. Mach.* **34**: 825–845, 1987.
65. Wagner, R. Common phrases and minimum-space text storage. *Commun. ACM* **16**(3): 148–152, 1973.
66. Wagner, R. An algorithm for extracting phrases in a space optimal fashion. *Commun. ACM* **16**(3): 183–185, 1973.
67. Weyland, N., and Puckett, P. Optimal binary models for the Gaussian source of fixed precision numbers. Technical Report, Mitre Corporation, Bedford, MA, 1986.
68. Welch, T. A. A technique for high-performance data compression. *IEEE Computer* **17**(6): 8–19, 1984.
69. Willems, F. M. J. Universal data compression and repetition times. *IEEE Trans. Inform. Theory* **35**: 54–58, 1989.
70. Witten, I. H., Neal, R., and Cleary, J. G. Arithmetic coding for data compression. *Commun. ACM* **30**(6): 520–540, 1987.
71. Ziv, J., and Lempel, A. An universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory* **23**(3): 337–343, 1977.
72. Ziv, J., and Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inform. Theory* **24**(5): 530–536, 1978.

8

GEOMETRIC HASHING AND ITS APPLICATIONS

GILL BAREQUET

The Technion—Israel Institute of Technology, Haifa 32000, Israel

I. INTRODUCTION

The *Geometric Hashing* technique was introduced, about a decade ago, as an efficient method for object recognition in computer vision. Since then it has been applied in many other fields, e.g., in medical imaging, molecular biology, and computer-aided design. The underlying idea of Geometric Hashing is using a database for storing pieces of information (features) of known geometric objects, in such a way that will allow a fast recognition of an unknown query object.

The main advantage of this technique is its ability to perform *partial* matching between geometric objects, e.g., for recognizing objects in an image which are partially occluded or have undergone some transformation. It does not depend on the existence of any particular predefined features in the matched objects. It is usually very easy to implement, and it performs fast and accurately.

The Geometric Hashing technique is an indexing-based approach, where local features of objects are encoded and stored in a database. Such approaches are now widely recognized as the method of choice for implementing reliable recognition systems that handle large model databases.

Database and Data Communication Network Systems, Vol. 1

II. MODEL-BASED OBJECT RECOGNITION

As mentioned above, the Geometric Hashing technique was first introduced in the context of model-based object recognition. Object recognition is an important and extensively studied problem in robotics applications of computer vision, including robot task and motion planning, and automatic image understanding and learning. Given a 2- or 3-dimensional image of a scene, we wish to identify in it certain types of objects (which may be only partially visible), and for each identified object to determine its position and orientation in the scene. Many approaches were developed for object recognition. These include *pose clustering* [27] (also known as *transformation clustering* and *generalized Hough transform* [1,22]), *subgraph isomorphism* [7], *alignment* [18], *iterative closest point* [9], and many *indexing* techniques (including Geometric Hashing). There is an enormously extensive literature on this subject. See, for example, the two comprehensive surveys given by Besl and Jain [8] and by Chin and Dyer [11]. One of the basic approaches to this problem is *model-based* object recognition. In order to identify objects that participate in a given scene, this approach assumes some prior knowledge about them, already stored efficiently in a *model database*. This technique first applies a "learning" process, in which the model objects are analyzed and preprocessed, which enables us to later perform the recognition task on-line, and usually very fast. The typical running time of the recognition step does not depend on the number of stored objects and on their complexities, but only on the complexity of the given scene (under some assumptions about the performance of the database, as detailed below).

The matching between a given image and a known (already processed) model is carried out by comparing their *features*. The database contains for each preprocessed model a set of features encoded by some function, which is *invariant* under the class of transformations by which the model objects are assumed to be placed in the scene. Such typical classes include translations, rigid motions, rigid motions and scalings, and affine or perspective transformations. In order to identify the query model, its features are encoded by the same function and compared to the contents of the database. A database model matches the query model if they have a sufficiently large number of features in common and if these corresponding features match each other under the same transformation. Many recognition systems use encoding functions that are invariant under rotation and translation, since they aim to identify objects subject to rigid motions, but, as just noted, other classes of transformations may also be considered.

The recognition task usually requires the ability to perform only a partial matching between objects. This is either because only portions of the objects may match or, more typically, because the query object may be partially occluded in a composite scene. In addition, the recognition system is usually expected to tolerate some amount of noise, either because the input image is obtained with some inaccuracy or because the objects to be matched are only similar but not identical to those in the model database, or simply because the encoding function is not a one-to-one mapping.

A major contribution to automatic object recognition was made by the *partial curve matching* technique, which was first suggested by Kalvin *et al.* [19]

and by Schwartz and Sharir [26]. This technique, which uses the Geometric Hashing method, originally solved the curve matching problem in the plane, under the restrictive assumption that one curve is a proper subcurve of the other one, namely: Given two curves in the plane, such that one is a (slight deformation of a) proper subcurve of the other, find the translation and rotation of the subcurve that yields the best least-squares fit to the appropriate portion of the longer curve.

This technique was extended (by removing the curve-containment restriction) and used in computer vision for automatic identification of partially obscured objects in two or three dimensions. The Geometric Hashing technique was applied [17,20,21,28] in various ways for identifying partial curve matches between an input scene boundary and a preprocessed set of known object boundaries. This was used for the determination of the objects participating in the scene, and the computation of the position and orientation of each such object.

III. PRINCIPLES OF GEOMETRIC HASHING

Geometric Hashing is a general model-based recognition technique that can be applied in any dimension under different classes of transformations. It identifies efficiently partial matches between objects in a given scene and objects stored in a model library. In a nutshell, the method consists of two steps. In the first off-line step features of some given model objects are analyzed, encoded by some transformation-invariant function, and stored in a database (usually an efficient hashing table). In the second step, a query object is analyzed on-line, and the database is scanned for locating features of the model objects that match (or are similar to) those of the query object. Such matching features suggest partial matches between the query object and the model objects, and also the respective geometric transformation between them. A voting scheme is then applied for identifying candidate partial matches between the query object and some of the model objects (and the respective transformations).

There are three basic aspects to the Geometric Hashing technique:

1. Representation of the object features using transformation invariants, to allow recognition of an object subject to any allowed transformation;
2. Storage of these invariants in a hashing table to allow efficient retrieval, which is (nearly) independent of the complexity of the model database; and
3. Robust matching scheme that guarantees reliable recognition even with relatively small overlap and in the presence of considerable noise.

The original variant of this technique aimed to find partial matches between curves in the plane. We first describe the technique in this context. We assume that some collection of "known" curves is preprocessed and stored in a database, and that the actual task is to find matches between *portions* of a composite query curve and portions of the curves stored in the database, subject to a rigid motion in the plane.

In the preprocessing step, features of all the curves are generated, encoded, and stored in a database. Each curve is scanned and *footprints* are generated at equally spaced points along the curve. Each point is labeled by its sequential number (proportional to the arclength) along the curve. The footprint is chosen so that it is invariant under a rigid motion of the curve. A typical (though certainly not exclusive) choice of a footprint is the second derivative (with respect to arclength) of the curve function; that is, the *change* in the direction of the tangent line to the curve at each point. Each such footprint is used as a key to a hashing table, where we record the curve and the label of the point along the curve at which this footprint was generated. The (expected) time and space complexity of the preprocessing step is linear in the total number of sample points on the curves stored in the database. Since the processing of each curve is independent of the others, the given curves can be processed in parallel. Moreover, adding new curves to the database (or deleting curves from it) can always be performed without recomputing the entire hashing table. The construction of the database is performed off-line before the actual matching.

In the recognition step, the query curve is scanned and footprints are computed at equally spaced points, with the same discretization parameter as for the preprocessed curves. For each such footprint we locate the appropriate entry in the hashing table, and retrieve all the pairs (curve,label) stored in it. Each such pair contributes one vote for the model curve and for the relative shift between this curve and the query curve. The shift is simply the difference between the labels of the matched points. That is, if the footprint of the ith sample point of the query curve is close enough to the footprint of the jth point of model curve c, then we add one vote to the curve c with the relative shift $j - i$. In order to tolerate small deviations in the footprints, we do not fetch from the hashing table only the entry with the same footprints as that of the point along the query curve, but also entries within some small neighborhood of the footprint. A commonly used implementation of this process is by range-searching (see, e.g., [23, p. 69] or [10]). The major assumption on which this voting mechanism relies is that real matches between long portions of curves result in a large number of footprint similarities (and hence votes) between the appropriate model and query curves, with almost identical shifts. By the end of the voting process we identify those (curve,shift) pairs that got most of the votes, and for each such pair we determine the approximate endpoints of the matched portions of the model and the query curves, under this shift. It is then straightforward to compute the rigid transformation between the two curves, with accuracy that increases with the length of the matched portions. The running time of the matching step is, on the average, linear in the number of sample points generated along the query curve. This is based on the assumptions that on the average each access to the hashing table requires constant time, and that the output of the corresponding range-searching queries has constant size. Thus, the expected running time of this step does not depend on the number of curves stored in the database and on the total number of points on the curves.

Many generalizations and applications of the Geometric Hashing technique have appeared in the literature. These include different choices of the allowed transformations, specific domains in which the technique is used (e.g., locating an object in a raster image, registration of medical images, molecule docking),

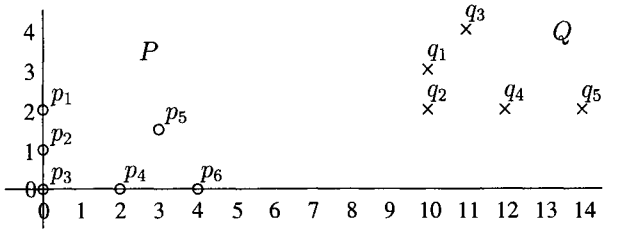


FIGURE 1 Two point sets.

and generalizations to higher dimensions. We note that in most cases the key to success is defining a good footprint system (in the sense detailed above), so that the "correct" solutions manifest themselves by sufficiently many correct votes. In practice, every application of the Geometric Hashing technique has its own special footprint setting, which strongly depends on the nature of the problem in question.

IV. EXAMPLES

Let us illustrate the course of Geometric Hashing by a simple example of matching the two planar point sets shown in Fig. 1. The two sets, denoted by P and Q, contain six and five points, respectively.

In the first matching experiment we allow only translations, and we assume the availability of only the point coordinates. Here every pair of matched points, one of each set, defines uniquely the translation that maps P to Q. (Specifically, if the point $p \in P$ matches the point $q \in Q$, then the sought translation is simply $\vec{q} - \vec{p}$.) Thus the matched feature is a point, and in the absence of any additional information, each pair of points (p, q) (where $p \in P$ and $q \in Q$) contributes one vote for the translation $\vec{q} - \vec{p}$. (In situations where we trust some of the matching pairs, we can weigh the votes.) The resulting voting table, which is also a detailed description of the Minkowski difference between the sets Q and P (denoted as $Q \ominus P$), is shown in Fig. 2. Indeed, the "correct" translation (10,2) received the largest number of votes (4). This is because four pairs of points, one of each set (namely, (p_2, q_1), (p_3, q_2), (p_4, q_4), and (p_6, q_5)) casted votes for this translation. All the other 26 votes were (luckily) spread in the voting table so that no other translation received four or more votes. Geometric Hashing is thus a viable method when the number of "correct" votes (compared to a

4		1		1		1			
3	1		2		1	1			
2	1	1	2		4	1	2		1
1		1		1	2	1	1		1
0					1		1		1
	6	7	8	9	10	11	12	13	14

FIGURE 2 Voting table for $Q \ominus P$.

usually much higher number of "incorrect" votes) is sufficient for identifying the sought transformation, or at least for including it in the first few candidate solutions which receive the largest numbers of votes. The leading candidates are then transferred to a secondary validation process, which applies other methods for determining whether a candidate solution is correct.

In the second matching experiment we allow translations and rotations between the two sets. Formally, the set P is assumed to have undergone a rotation by some angle θ around the origin $(0, 0)$, followed by some translation $t = (t_x, t_y)$, so as to form the set Q. Note that in this experiment the scaling of the Euclidean plane is not allowed. A natural choice of the matched feature is an *ordered pair* of points (in the same set). For each ordered pair of points (p_{i_1}, p_{i_2}) (where $p_{i_1}, p_{i_2} \in P$, $1 \le i_1, i_2 \le 6$, and $i_1 \ne i_2$), and for each ordered pair of points (q_{j_1}, q_{j_2}) (where $q_{j_1}, q_{j_2} \in Q$, $1 \le j_1, j_2 \le 5$, and $j_1 \ne j_2$), we first check whether the lengths of the line segments defined by the two pairs of points are similar. For practical reasons (noisy input data, approximate matching, etc.) we tolerate some deviation between the two segment lengths. Namely, we check whether $||\overline{p_{i_1} p_{i_2}}| - |\overline{q_{j_1} q_{j_2}}|| \le \varepsilon$, where ε is some tuning parameter of the algorithm, usually specified by the user. (This parameter represents the user's *a priori* estimation of the maximum matching error between the sets P and Q.) If the two lengths are not identical (or not approximately the same), then no rotation and translation can map the first pair of points to the second pair. In such case we proceed to checking the next match between two pairs of points. However, in case the two lengths are similar, we compute the unique rotation and translation that realize the match between the two pairs. Denote the point coordinates by $p_{i_1} = (x_{i_1}, y_{i_1})$, $p_{i_2} = (x_{i_2}, y_{i_2})$, $q_{j_1} = (x_{j_1}, y_{j_1})$, and $q_{j_2} = (x_{j_2}, y_{j_2})$. A simple calculation shows that the sought transformation is

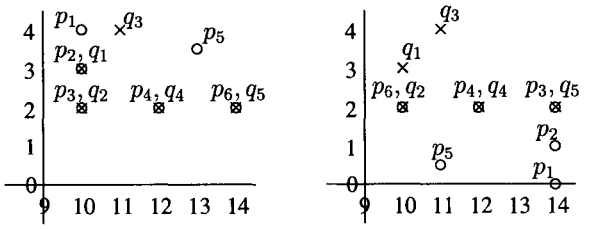
$$\theta = \arctan((y_{j_2} - y_{j_1})/(x_{j_2} - x_{j_1})) - \arctan((y_{i_2} - y_{i_1})/(x_{i_2} - x_{i_1})),$$
$$t_x = x_{j_1} - \cos\theta \cdot x_{i_1} + \sin\theta \cdot y_{i_1}, \quad \text{and}$$
$$t_y = y_{j_1} - \sin\theta \cdot x_{i_1} - \cos\theta \cdot y_{i_1}.$$

Obviously every match between (p_{i_1}, p_{i_2}) and (q_{j_1}, q_{j_2}) is also reflected by the match between (p_{i_2}, p_{i_1}) and (q_{j_2}, q_{j_1}). We avoid this redundancy by requiring that $j_1 < j_2$. In this example there are 30 matches between pairs of points, where each pair consists of two points of the same set. The two leading transformations in the voting table are

1. 6 votes: $\theta = 0°, t = (10, 2)$;
2. 3 votes: $\theta = 180°, t = (14, 2)$.

All the other 21 candidate matches receive only one vote each. Fig. 3 shows the two best solutions by superimposing the transformed set P with the set Q.

So far in our matching experiments the points had no additional information attached to them (except, of course, their coordinates). Assume now that each point is also attributed by some value which we denote as its *footprint*. This value can be a number, a vector of numbers, color, or any other property that is invariant under the mapping that transformed P into Q. In this setting, matching pairs of features (one of each set) should have similar footprints (up to some predefined tolerance), while features that should not match should



(a) $\theta = 0°$, $t = (10, 2)$ (b) $\theta = 180°$, $t = (14, 2)$

FIGURE 3 Matching by rotating and translating.

have significantly different footprints. This assumption allows us to significantly speed up the matching process by storing each set in a database (usually a hashing table) whose keys are the footprints. As noted above, the database implementation should allow not only fetching the entry (or entries) that match a given key, but also fetching all the entries whose keys are close to the given key up to some specified tolerance. This is usually achieved by implementing a data structure that allows range-searching queries. In our example it suffices to store only the set P in a database. For matching under translation only, each point $q \in Q$ (or, rather, its footprint) is used as a key for a range-searching query in the database that contains the set P. Each point $p \in P$ returned by the query then casts a vote for $\vec{q} - \vec{p}$ in the voting table, and the matching proceeds as described above.

Refer again to the matching experiment in which we allowed rotations and translations. In this experiment the matched feature was a *pair* of points. Assuming that each set contains n points, each set has $O(n^2)$ pairs and we now potentially have $O(n^4)$ candidate matches. An alternative used in many Geometric Hashing works (e.g., [14]) is to use every pair of points as a *basis* of a coordinate system, to specify the coordinates of all the other points of the same set with respect to that system, and to store all the points in the database in a redundant manner, so that each point is specified in terms of all the bases in which it does not take part. The matching feature in this variant is a single point: points are matched according to their basis-defined coordinates. The information given by the match (namely, the two bases—one of each set, and the translation between the two matched points) is enough for casting a vote for a unique rotation and a unique translation. Asymptotically we do here the same amount of work. Each point appears (redundantly) $O(n)$ times in the database, so the number of stored points is now $O(n^2)$. Then the matching step considers every pair of points, one of each database, giving us a total of $O(n^4)$ work.

The same ideas work for higher dimensions. In a 3-dimensional space, for example, we can consider triples of points. For every pair of congruent triangles, one defined by three points of the first set and the other defined by three points of the other set, we cast a vote for the transformation that maps the first triple to the second triple. Alternatively, we can have every three noncollinear points define a basis for a coordinate system and redundantly represent each point by its coordinates with respect to all the bases. In the matching step each pair of points, together with their respective coordinates according to some basis,

casts a vote for the appropriate transformation. In both methods we may use footprints that depend on the application for pruning the matched pairs of features.

V. IMPLEMENTATION ISSUES

A. Footprint Quality and Matching Parameter

The success of the Geometric Hashing technique crucially relies on the "descriptiveness" of the footprint system. That is, we expect features that should match to have similar footprints, and expect features that should not match to have footprints that differ significantly enough. In practice, the amount of incorrect votes usually dominates the amount of correct votes, but when the distribution of the incorrect votes does not have too-high random peaks, the correct votes still exhibit an "accumulation point" in the voting table, which suggests the correct solution for the matching problem.

As mentioned earlier, the running time of the matching step is, on average, linear in the total number of features in the two sets. This is based on the assumptions that on average each access to the hashing table requires constant time, and that the output of the corresponding queries has constant size. This is due to the nature of hashing and does not assume anything about the input to the algorithm. Nevertheless, it requires a reasonable choice of the proximity parameter ε, which should yield on average a constant number of output points for each range-searching query. Improper choice of ε, say, equal to the size of the entire set, will result in a running time that is quadratic in the complexity of the input.

B. Rehashing

Even when the footprint quality is satisfactory, it is desirable to have the distribution of footprints (in the space of invariants) as uniform as possible. This is for optimizing the performance of the hashing table. A highly nonuniform distribution interferes with the balance of the hashing bins that store the footprints, while the most occupied bin determines the worst-case performance of the hashing table. When the probability density function of the footprints is known, one can transform the input point coordinates so as to make the *expected* distribution of footprints uniform (see Figs. 5a and 6a of [29, p. 16]. If the rehashing function is chosen carefully, the new hashing table can have the same number of bins as that of the original table.

VI. APPLICATIONS

As noted earlier, the first application of Geometric Hashing was for partial curve matching in the plane [19,26]. This application was later extended in [17,20, 21,28] for identifying objects participating in a scene. (In fact the silhouettes of the objects and of the scene were used to solve a 2-dimensional matching

problem.) In addition to these object-recognition problems, Geometric Hashing was used in several other domains, some of which are detailed below.

A. Molecular Biology

Geometric Hashing was used to solve surface and volume matching problems that arise in molecular biology. One such typical problem is to find a "docking" of two molecules (or subunits of the same molecule). Here one seeks a rigid motion of one molecule relative to the other, which creates a good geometric fit between large portions of the molecule boundaries, so that the molecules themselves remain disjoint; that is, one seeks surface matching and volume complementarity. (In practice, the docking of molecules may depend also on a good chemical/electrical fit between the atoms participating in the docking. The purely geometric statement of the problem is thus only an approximation of the problem, but it is appropriate in many cases.) Another typical problem is to detect similar 3-dimensional structural motifs in macromolecules. Here one also seeks a rigid motion that creates a good fit between large portions of the molecule surfaces, but now one wants their volumes to overlap near this fit. The standard representation of a molecule is just a list of its atoms and their spatial positions. Consequently, a major difficulty in studying molecule docking and structural motifs is in the definition and computation of the molecule boundary. We do not elaborate here on available techniques for overcoming this difficulty.

First attempts to solve the molecule docking problem, which are based on energy minimization, were only partially successful. Geometric approaches were much more successful, but (at least the earlier ones) were not reliable enough, and suffered from unacceptedly long computation time. Some geometric methods use clique-search algorithms in graphs; other methods perform a brute-force search over all the discretized 3-dimensional rotations, while using a secondary method for identifying the appropriate translation. Traditional methods for detecting structural motifs in proteins usually employ string-matching algorithms. A survey of these methods, most of which are based on dynamic programming, is found in [25].

A major contribution to the problems of detecting structural motifs and of molecule docking was achieved by applying Geometric Hashing. In this application, the method proceeds by assigning footprints to the molecule atoms, then by matching the footprints and by voting for the relative transformation (rigid motion) of one molecule relative to the other. For the motif detection, Nussinov and Wolfson [24] define the footprint of each atom as its coordinates in systems defined by any three noncollinear atoms (thus each atom has $O(n^3)$ footprints, where n is the number of atoms in the molecule). Similar ideas are presented in [13]. Fischer *et al.* [14] take a similar approach for the molecule docking problem. In one variant, each pair of atoms defines a basis (whose length is the distance between the two atoms), and the footprint of every atom is defined as the distances from the atom to the endpoints of every basis, coupled with the length of the basis (thus each atom has $O(n^2)$ footprints). In another variant, the angles between the normal to the surface (at the candidate atom) and the normals at the endpoints of the basis, as well as a knob/hole label of

the atom (obtained in a preprocessing step), are also considered. In all cases, the footprints are stored in a hashing table, which makes it possible to retrieve entries with some tolerance. Here this is needed not just because of the noisy footprints, but also because of the conformational changes that might occur in the molecule structures during the reaction between them.

B. Medical Imaging

The topic of medical image matching has attracted a lot of attention in the medical literature. The problem arises when complementary information about some organ is obtained by several imaging techniques, such as CT (computed tomography) and MRI (magnetic resonance imaging). The goal is to match (register) the various models of the same organ obtained by these methods, in order to obtain a single improved and more accurate model. Such a registration is needed because the orientations of the organ usually differ from one model to another.

Many methods, which are similar to the methods for object recognition, were proposed for the solution of this organ registration problem. These include, among many others, approximated least-squares fit between a small number of markers, singular-value decomposition for matching point pairs, high-order polynomials for a least-squares fit, "thin-plate spline" for registering intrinsic landmarks or extrinsic markers, parametric correspondence, chamfer maps, partial contour matching, moments and principal axes matching, and correlation functions. Detailed reviews of image-registration techniques are given in [6,12]. Geometric Hashing was also exploited for alignment of medical data by Barequet and Sharir [5] and by Guéziec *et al.* [16]. The former work matched 3-dimensional point sets (voxels), while the latter work registered 3-dimensional curves extracted from the data.

C. Other Applications

Geometric Hashing has been applied to other matching problems as well. Germain *et al.* [15] use this technique for matching real (human) fingerprints for noncriminal identification applications. Barequet and Sharir [2,3] apply Geometric Hashing to solve a computer-aided design problem, namely, for detecting and repairing defects in the boundary of a polyhedral object. These defects, usually caused by problems in CAD software, consist of small gaps bounded by edges that are incident to only one face of the model. Barequet and Sharir [4] apply a similar technique to the reconstruction of a three-dimensional surface (bounding a human organ) from a series of polygonal cross sections.

REFERENCES

1. Ballard, D. H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognit.* 13 (2): 111–122, 1981.
2. Barequet, G. Using geometric hashing to repair CAD objects. *IEEE Comput. Sci. Engrg.* 4 (4): 22–28, 1997.

3. Barequet, G. and Sharir, M. Filling gaps in the boundary of a polyhedron. *Comput. Aided Geom. Design* **12** (2): 207–229, 1995.
4. Barequet, G. and Sharir, M. Piecewise-linear interpolation between polygonal slices. *Comput. Vision Image Understanding*, **63** (2): 251–272, 1996.
5. Barequet, G. and Sharir, M. Partial surface and volume matching in three dimensions. *IEEE Trans. Pattern Anal. Mach. Intell.* **19** (9): 929–948, 1997.
6. Brown, L. G. A survey of image registration techniques. *ACM Comput. Surveys* **24**: 325–376, 1992.
7. Bolles, R. C. and Cain, R. A. Recognizing and locating partially visible objects: The local-feature-focus method. *Int. J. Robot. Res.* 1 (3): 637–643, 1982.
8. Besl, P. J. and Jain, R. C. Three-dimensional object recognition. *ACM Comput. Surveys* **17** (1): 75–154, 1985.
9. Besl, P. J. and McKay, N. D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **14** (2): 239–256, 1992.
10. Chazelle, B. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.* **17** (3): 427–462, 1988.
11. Chin, R. T. and Dyer, C.R. Model-based recognition in robot vision. *ACM Comput. Surveys* **18** (1): 67–108, 1986.
12. van der Elsen, P. A. Pol, E. J. D., and Viergever, M. A. Medical image matching—A review with classification. *IEEE Engrg. Med. Biol.* **12** (1): 26–39, 1993.
13. Fischer, D., Bachar, O., Nussinov, R., and Wolfson, H. J. An efficient computer vision based technique for detection of three dimensional structural motifs in proteins. *J. Biomolec. Structure Dynam.* 9: 769–789, 1992.
14. Fischer, D., Norel, R., Nussinov, R., and Wolfson, H. J. 3-D docking of protein molecules. In *Proc. 4th Symp. on Combinatorial Pattern Matching,* Lecture Notes in Computer Science 684, pp. 20–34. Springer-Verlag, Berlin, 1993.
15. Germain, R. S., Califano, A., and Colville, S. Fingerprint matching using transformation parameter clustering. *IEEE Comput. Sci. Engrg.* **4** (4): 42–49, 1997.
16. Guéziec, A. P., Pennec, X., and Ayache, N. Medical image registration using geometric hashing. *IEEE Comput. Sci. Engrg.* **4** (4): 29–41, 1997.
17. Hong, J. and Wolfson, H. J. An improved model-based matching method using footprints. In *Proc. 9th Int. Conf. on Pattern Recognition,* Rome, Italy, November 1988, pp. 72–78.
18. Huttenlocher, D. P. and Ullman, S. Recognizing solid objects by alignment with an image. *Int. J. Comput. Vision* 5 (2): 195–212, 1990.
19. Kalvin, A., Schonberg, E., Schwartz, J. T., and Sharir, M. Two-dimensional, model based, boundary matching using footprints. *Int. J. Robot. Res.* **5** (4): 38–55, 1986.
20. Kishon, E., Hastie, T., and Wolfson, H. 3-D curve matching using splines. *J. Robot. Systems* 8 (6): 723–743, 1991.
21. Lamdan, Y., Schwartz, J. T., and Wolfson, H. J. Affine invariant model-based object recognition. *IEEE Trans. Robot. Automat.* **6** (5): 578–589, 1991.
22. Linnainmaa, S., Harwood, D., and Davis, L. S. Pose determination of a three-dimensional object using triangle pairs. *IEEE Trans. Pattern Anal. Mach. Intell.* **10** (5): 634–647, 1988.
23. Mehlhorn, K. *Data Structures and Algorithms 3: Multi-Dimensional Searching and Computational Geometry* (Brauer, W., Rozenberg, G., and Salomaa, A. Eds.). Springer-Verlag, Berlin, 1984.
24. Nussinov, R. and Wolfson, H. J. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *In Proc. Natl. Acad. Sci. USA* **88**: 10,495–10,499, 1991.
25. Sankoff, D. and Kruskal, J. B. *Time Warps, String Edits and Macromolecules.* Addison-Wesley, Reading, MA, 1983.
26. Schwartz, J.T. and Sharir, M. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *Int. J. Robot. Res.* 6 (2): 29–44, 1987.
27. Stockman, G. Object recognition and localization via pose clustering. *Comput. Vision Graphics Image Process.* 40 (3): 361–387, 1987.
28. Wolfson, H. J. On curve matching. *IEEE Trans. Pattern Anal. Mach. Intell.* **12** (5): 483–489, 1990.
29. Wolfson, H. J. and Rigoutsos, I. Geometric hashing: An overview. *IEEE Comput. Sci. Engrg.* 4 (4): 10–21, 1997.

This Page Intentionally Left Blank

9

INTELLIGENT AND HEURISTIC APPROACHES AND TOOLS FOR THE TOPOLOGICAL DESIGN OF DATA COMMUNICATION NETWORKS

SAMUEL PIERRE

Mobile Computing and Networking Research Laboratory (LARIM), and Department of Computer Engineering, École Polytechnique de Montréal, Montréal, Quebec, Canada H3C 3A7

I. INTRODUCTION

A typical data communication network essentially consists of a set of nodes representing workstations, switches, routers, and so on, linked to each other by means of communication links. As shown in Fig. 1, such a network is generally

Database and Data Communication Network Systems, Vol. 1

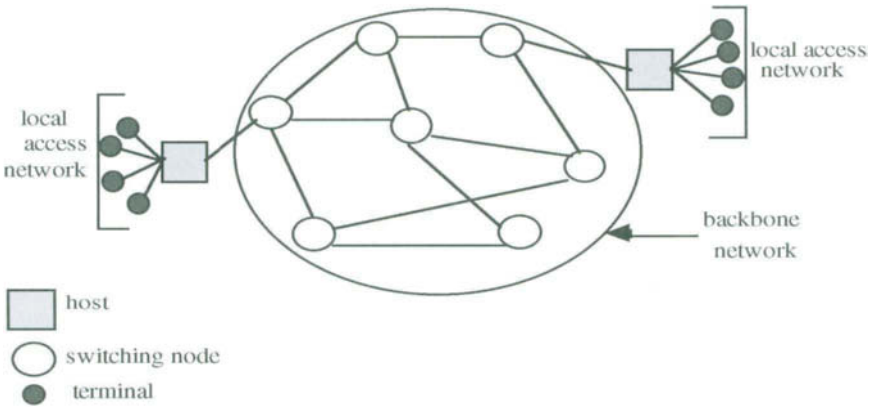


FIGURE 1 Network hierarchy.

considered as a hierarchical structure integrating two levels: the backbone network at the first level, and the local access networks at the second level. The backbone network is dedicated to the delivery of information from source to destination. The local access networks are typically centralized systems that essentially allow users to access hosts or local servers. In this chapter, the focus is on the backbone network design considered as a distributed network.

The topological design of data communication networks consists essentially of finding a network topology that satisfies at the lowest possible cost some constraints related to quality of service and reliability [6,9,12,19,28,29,46]. Traditionally formulated as an integer programming problem and for various reasons discussed and reported elsewhere [3,9–12,39,45], it is considered to be a very difficult optimization problem [20]. In fact, if n indicates the number of nodes, the maximum number of links is given by $n(n-1)/2$, and therefore the maximum number of topological configurations of n nodes is $2^{n(n-1)/2}$. For instance, if $n = 11$, the number of topological configurations that can be exhaustively explored is 3.603×10^{13}; at the generation speed of 10^6 configurations per second, the overall CPU time required for such an exploration is 1,142.46 years. Even by taking into account only the configuration aspect of this problem, the risk of combinatorial explosion is already obvious.

In order to facilitate its resolution, the topological design problem is usually divided into three subproblems: (1) topological configuration taking into account reliability aspects, (2) routing or flow assignment, and (3) capacity assignment [43]. Nevertheless, this division does not enable one to solve the overall problem in a reasonable CPU time. As a result, this problem is realistically solved by means of heuristic methods that attempt to reduce the search space of candidate topologies, even if that possibly leads to suboptimal solutions. Clearly, determining the optimal size of such a search space would be a compromise between the exploration time and the quality of the solution. This observation led many researchers to develop heuristic approaches leading to "good" solutions, instead of optimal solutions [35,37,40,45].

The range and the nature of the heuristics vary widely [17]. Some of them are inspired by formal optimization approaches [10,11]. Linear programming

methods (without constraints involving integer variables) can take advantage of convexity; unfortunately this is not the case with integer programming methods, which are characterized by nonconvex solution spaces. The exploration of such spaces by local search methods leads to minima that may only be local. For that reason, over the past ten years, many researchers have opted for meta-heuristics such as simulated annealing, genetic algorithm, tabu search, and artificial neural networks, in order to solve network design problems by generalizing and improving conventional local search methods [23,24,32–35,38]. On the other hand, several other researchers, looking for solution efficiency, transparency, and user-friendliness have adopted artificial intelligence (AI) approaches, particularly knowledge-based systems [9,36,37,40]. Dutta and Mitra [9] have proposed a hybrid method that integrates both the algorithmic approach and a heuristic knowledge system. This method builds a solution by subdividing the network design problem into modules that can be individually solved by applying either optimization models or heuristic methods. Furthermore, it integrates the partial solutions to obtain a global solution of the design problem.

This chapter presents and analyzes some intelligent and heuristic approaches and tools that have been used for designing distributed data communication networks. Section II presents basic concepts and discusses related work and background. Section III analyzes network characterization and representation issues considered as fundamental aspects of network planning. Section IV studies some design approaches based on the artificial intelligence concepts of knowledge-based system and machine learning. Section V presents heuristic design appraoches, with a particular emphasis on tabu search and some comparisons with other meta-heuristics such as simulated annealing and genetic algorithms.

II. BASIC CONCEPTS AND BACKGROUND

The topological design problem of computer networks can be considered as part of the overall network planning. It consists of finding a network topology configuration that minimizes the total communication cost, taking into account some constraints such as delay and reliability. In this section, we first present some basic definition needed to formulate the problem, then discuss other approaches and related work.

A. Acronyms, Notation, and Basic Assumptions

Acronyms/Abbreviations

AI	Artificial intelligence
BXC	Branch X-change
CBE	Concave branch elimination
CS	Cut saturation
FD	Flow deviation
GA	Genetic algorithm
MENTOR	Mesh network topology optimization and routing

SA	Simulated annealing
TS	Tabu search
WAN	Wide-area network
MAN	Metropolitan-area network
LAN	Local-area network

Notation

G	A graph
N	Set of nodes of a graph
A	Set of edges of a graph
n	Number of nodes of a network
m	Number of links of a network
$m_{\max}$	Maximum number of links
R	Diameter of the network, that is, the length of the longest of the shortest paths over all node pairs in the network
K	Connectivity degree of a network
C_k	Capacity of link k, that is, the maximum data rate in bits per second (bps) carried by this link
f_k	Flow of link k, that is, the effective data rate in bps on this link
U_k	Utilization of link k, that is, the ratio f_k/C_k
L_k	Length of the link k
$L_{\max}$	Maximum Euclidean distance between any node pair of the network
$d(i)$	Incidence degree of a node i, that is, the number of links connected to it
d_G	Degree of a graph G, that is, the degree of the node having the smallest degree among all the nodes
$C = (C_k)$	link capacity vector
$f = (f_k)$	Link flow vector
γ_{ij}	Traffic (number of packets per second exchanged) between nodes i and j
γ	Total traffic in a network
$r_{\text{aver}} = \gamma/m_{\max}$	Average traffic between all the node pairs
$I_{ij} = \gamma_{ij}/r_{\text{aver}}$	Index traffic of link (i, j)
Γ	Traffic matrix
T	Average packet delay in a network
$T_{\max}$	Maximum acceptable delay
$T_n = T/T_{\max}$	Normalized average delay
$d_k(C_k)$	Cost capacity function of link k
Variable unit cost	Cost in \$/month/km for a given link capacity
Fixed cost	Cost in \$/month for a given link capacity
CMD_{ij}	The most economical path between two nodes i and j

Assumptions

- Poisson distribution of the traffic between node pairs
- Exponential distribution of packet size with a mean of $1/\mu$ bits/packet

- Infinite nodal memory
- Independence and identical distribution of interarrival times
- Independence and identical distribution of transmission times on each link.

B. Definitions and Problem Statement

Quality of service in a data communication network is usually measured by the average packet delay T. Based on the set of assumptions of the previous section and according to Little's rules, the average packet delay can be expressed as [12,13]

$$T = \frac{1}{\gamma} \sum_{k=1}^{m} \frac{f_k}{C_k - f_k}. \tag{1}$$

The average delay T given by (1) must be less than or equal to the maximum acceptable delay $T_{\max}$.

Equation (1) does not take into account the propagation delay and the nodal processing time. These factors play a more important role in high-speed networks where it is unrealistic to neglect them. Furthermore, the validity of the previous assumptions (Section A) has been tested by simulation studies on a variety of applications [12]. Results confirm the robustness of the model. Thus, the average packet delay obtained under these assumptions is realistic for medium-speed packet-switched networks. However, such assumptions can be unrealistic if one is interested in estimating the delay of a particular packet or the delay distribution rather than just the average value.

The cost d_k of the link k is generally a function of the link length and capacity. Therefore, the total link cost D is given by

$$D = \sum_{k=1}^{m} d_k(C_k). \tag{2}$$

The techniques used to solve the problem of the capacity assignment essentially depends on the nature of the cost capacity functions $d_k(C_k)$, which can be linear, concave, or discrete [12]. In practice, the cost of the link k includes two components: a fixed part and a variable part which depends on the physical length L_k of this link:

$$d_k(C_k) = (\text{Variable unit cost})_k L_k + (\text{Fixed cost})_k. \tag{3}$$

Variable unit cost represents the price structure for leased communications links; the fixed cost refers to a constant cost associated with a given link and represents the cost of a modem, interface, or other piece of equipment used to connect this link to its end nodes.

The flow assignment consists of determining the average number of packets λ_k on each link k [43]. For this purpose, a routing policy that determines the route to be taken by packets between each source–destination pair is needed. Routing strategies are generally classified into fixed, adaptative, and optimal routing. In this chapter, for sake of simplicity, we adopt the fixed routing based on the Euclidean distance between nodes.

Given:

- the number of nodes n and their location (X_i, Y_i), with i=1, 2, ..., n
- the traffic requirements $\Gamma = (\gamma_{ij})$, with i, j=1, 2, ..., n and $i \neq j$
- the capacity options and their costs
- the maximum acceptable delay T_{max} in (ms)
- the desired level of reliability (K-node-connectivity)

Minimize

the network cost D

Subject to:

- Network delay constraint: $T \leq T_{max}$
- Reliability constraint : K-node-connectivity
- Capacity constraint : $f \leq C$

Design variables:

- topological configuration
- routing strategy
- capacity assignment.

FIGURE 2 General formulation of the topological design problem.

The reliability of a data communication network depends on the availability and the reliability of their components. For this reason, it is necessary to evaluate the overall network reliability by taking into account in the design phase the possibility of link or node failures [18]. Many reliability measures have been proposed for computer communication networks. The most popular among these is the concept of *K*-connectivity, which integrates both arc-connectivity and node-connectivity [41]. A graph is said to be *K*-arc-connected if and only if all pairs of nodes are connected by at least *K* arc-disjoint paths. Similarly, a graph is said to be *K*-node-connected if and only if any node pair is connected by at least *K* node-disjoint paths. If C_a denotes the arc-connectivity degree and C_n the node-connectivity degree, it is obvious that $C_n \leq C_a \leq d_G$. As a result, for a strong topological design, node-connectivity appears to be more relevant than arc-connectivity for measuring network reliability as well as for providing a certain level of network survivability and fault-tolerance. For large networks with high failure rates, a high level of node-connectivity is required (3-connectivity and more) in order to ensure an adequate level of reliability. For smaller networks, 2-connectivity has been suggested as a reliable measure [12].

Various formulations of the topological design problem can be found in the literature [9–13,21,18]. Generally, they correspond to different choices of performance measures, design variable, and constraints. Common formulations search either to minimize the average packet delay given the network cost or to maximize the network throughput given the network cost and the admissible average delay [10]. This chapter adopts the general formulation shown in Fig. 2.

III. CHARACTERIZATION AND REPRESENTATION OF DATA COMMUNICATION NETWORKS

The network design problem raises three kinds of questions: What is the usefulness of designing a model for representing data communication networks? How

could these models be used to improve the network performance? What degree of manoeuvrability must these models have? This section identifies the main characteristics of a typical data communication network, then analyzes some issues related to its representation, and finally presents some existing network representation or design tools.

A. Network Characterization

A data communication network is characterized by different structures, types, topologies, components, performance indexes, and management strategies. Regarding structures, one can distinguish between the distributed and centralized networks. In centralized networks, terminals or workstations are related to a single data source called a *server* through a variety of communication links. Conversely, distributed networks are characterized by the multiplicity of routes that link each source to a given destination.

Generally, three types of network should be distinguished: local-area network, metropolitan-area network, and wide-area network. A LAN is a local communication system connecting several computers, servers, and other network components; it makes possible high-speed data transfer (1 to 100 Mbps) through short distances, in small areas such as organizations, campuses, and firms. A MAN essentially serves a large city; it also regroups several computers with more or less reduced data throughput and makes it possible to link several enterprises in one city. WANs are used to connect several cities located at some tens of kilometers apart. Usually, the data throughput of a WAN is less than 100 Mbps.

The topology of a network informs on its configuration, that is, the way in which its nodes are linked to each other; it also indicates the capacity of each link in the network. If communication is established between two nodes through a direct link, one can speak of a point-to-point link; every network consisting of a point-to-point link is called a *point-to-point network.* Conversely, with a *multipoint link,* communication is rather broadcasted from one node to several nodes; every network consisting of multipoint links is a *multipoint network* or *general broadcasting network.*

Point-to-point networks can have a star, tree, ring, or mesh topology. In a star topology, all nodes are related by a point-to-point link to a common central node called the *star center.* All communications placed in this type of network must go through this node. In a tree topology, the network has a directory structure that is hierarchically structured. The principal node of this topology through which all applications pass is called a *tree root.* In this structure, the common link takes the form of a cable (with several branches) to which one or several stations are attached. In a ring topology, all the nodes are related to form a closed ring, which, in its turn, takes a point-to-point form. A mesh topology is formed by a number of links such that each node pair of the network is linked by more than one path. Generally, a mesh topology is used with WAN.

In multipoint networks, there are two types of topologies: the bus topologies and the ring topologies. In a bus topology, each network node is set linearly on a cable which constitutes a common physical link. The information is transmitted by any node through the entire bus in order to reach the other nodes of the network. In a ring configuration, all the nodes are set on a closed circuit

formed by a series of point-to-point links. These nodes form a ring; the information within the ring is transmitted in one sense.

The main components of a network are nodes and links. The nodes represent more or less advanced units, which could be terminals, servers, computers, multiplexers, concentrators, switches, bridges, routers, or repeaters. Each of these units has its own attributes. In the framework of topological design, the most relevant factors are cost, capacity, availability, compatibility, and reliability [22].

The cost of a node includes purchasing and maintenance, as well as the cost of related software. The capacity of a node refers to the speed of its processor, the size of both programs and available memories. Availability is defined by the percentage of time during which a node is usable, or else, as the probability that a node could be available in a given instant. The compatibility of a node can be defined as the concordance between the types of traffic which the node manages and the types of links to which that node could be attached.

The reliability of a node is considered as the probability that it correctly functions under given conditions. This means that the node is not a subject of repair nor of any intervention other than that predicted in the technical manuals.

A link or a transmission support refers to a set of physical means put in place in order to propagate electromagnetic signals that correspond to messages exchanged between an emitter and a receiver. There exist several types of transmission support. Each type has its distinct physical features, the way it carries data, and its realm of use. Most known and used transmission supports are twisted pairs, coaxial cables, electromagnetic waves, fiber-optic, and satellite links.

Like the nodes, each type of link has its own characteristics, which are attributes that could be taken into account during the topological design process. The most important attributes are the following: length, capacity, cost, flow, delay, and utilization. These attributes have been defined in Section A (Notation) of Section II.

The most usual indexes for measuring the performance of a network are the following: response time, data throughput, stability, easiness of extension, information security, reliability, availability, and cost. Response time can be defined as the time delay between the emission of a message by a node and the receipt of an answer. An efficient data throughput is the useful average quality of information processed by a unit of time and evaluated under given conditions and time period.

The stability of a network refers to its capacity to absorb a realistic traffic pattern. It can be defined as the number of tasks that the system can perform and the time needed to process each of these tasks in different instants, while the network is functioning. For a given network, the easiness of extension represents the capacity of this network to accept new users, or its capacity to be modified without structural changes. Information security includes both the information protection that restricts users' access to a part of the system, and the notion of information integrity, which depends on the capacity of the system to alter or lose information.

The reliability of a network can be defined as its ability to continue functioning when some of its nodes or links fail. It is often linked to the notion of network connectivity [31,34]. The availability of a network refers to the

probability that this network is usable during a period of time, given redundancies, breakdown detection, and repair procedures, as well as reconfiguration mechanisms.

Networks could assume several functions which could differ by their importance and use. Among these functions, one can mention routing, flow control, congestion control, configuration management, performance management, error and breakdown handling, security, and accounting information management. Figure 3 synthesizes the characteristics of data communication networks.

B. Network Representation

The purpose of the network representation is to find the most efficient way to model networks in order to make easier their design and analysis. To achieve this objective, the representation must be based on the use of data structures being sufficiently efficient (i.e., neither congested or complex) for minimizing the execution times or optimizing the data storage process. A good representation facilitates the implementation of both analysis and design algorithms [22].

There exist two types of representation: external and internal. The external representation of a network deals with aspects related to data display. In a design and analysis tool, external representation plays the important role of taking into account the data supplied by the user, controlling their integrity and organization, and storing in a manner that facilitates their use and update. At the level of this representation, the user specifies the number of nodes, the location of each node, the choice of links, and related characteristics as well as the traffic requirements. Internal representation has the purpose of facilitating the implementation of design and analysis algorithms. It also makes the latter efficient in terms of memory consumption and execution times.

Ag Rhissa *et al.* [1] describe two types of network representation: the inheritance tree and the content tree. The inheritance tree defines the hierarchical organization of the network's components and the administrative element included in these components. Such a representation can be transformed into an object structure whose route is a superclass that contains information regarding all the objects.

The content or instance tree defines the physical or logical relationships between the different elements of the network; it allows us to identify, in an unequivocal manner, an element from its positioning in the tree. This tree can also be transformed into an object structure and makes it possible to model the relationships between different objects in the form of an information management tree. The content tree essentially supplies information regarding the management of configurations [42].

Users still suffer from difficulties while they manipulate the design and representation models available in the marketplace. Their work is often constrained by the lack of flexibility of such models. Hence, it is necessary to provide for a generic representation model. By *generic model,* we refer to the capacity of representing a variety of types of networks and aspects related to their performance and management. Therefore, the following questions can be raised: What are the functions a representation model could offer to the user? What data structure could one use in order to be able to take into account all

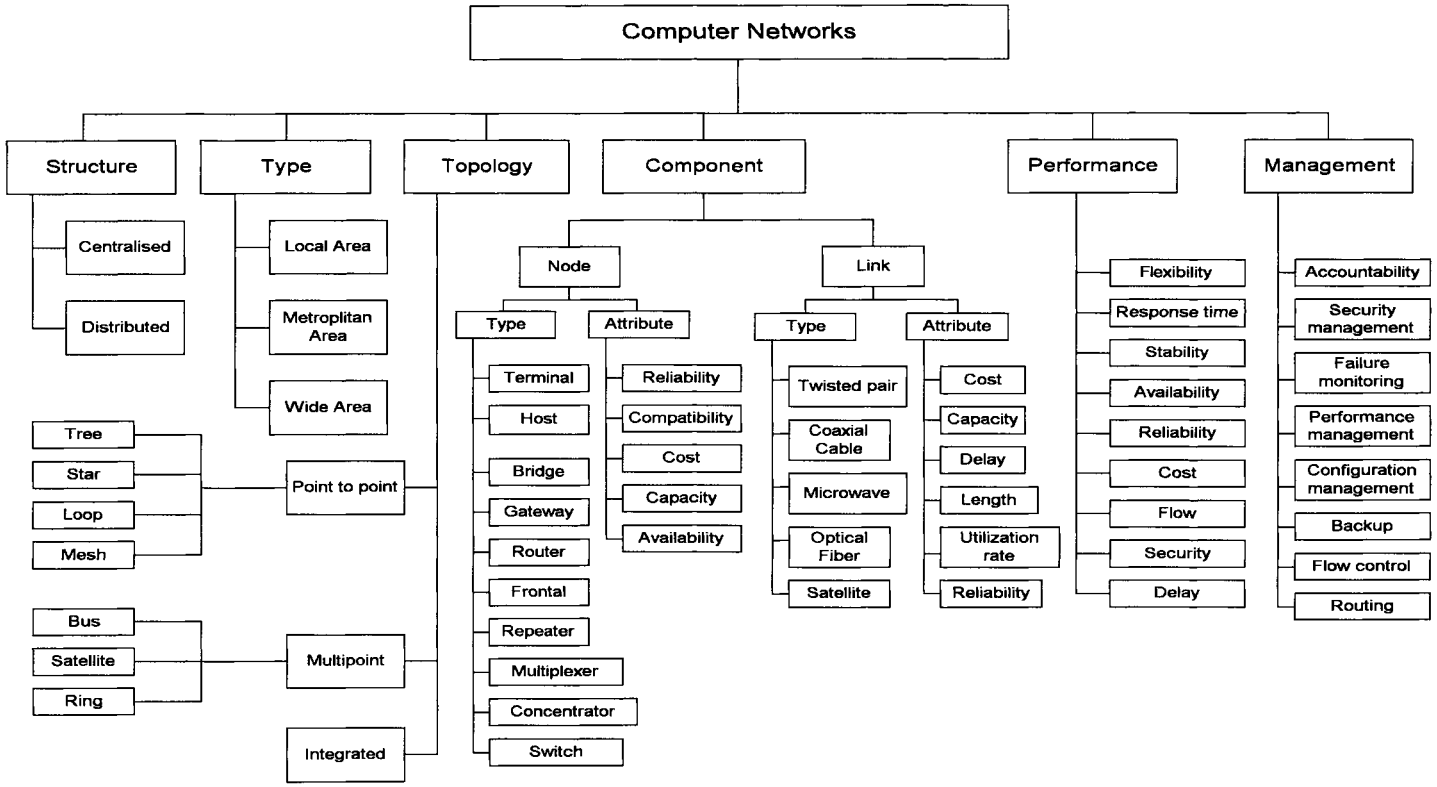


FIGURE 3 Characterization of data communication networks.

the aspects of a data communication network under design? How does one implement and validate such a model by taking into account the diversity of topologies, structures, and management mechanisms associated with the concept of a network?

The representation of networks is considered an important aspect of topological design. Most models and tools were rather devoted to the aspects of routing, evaluation, and optimization of performance. In these models, a network representation is often limited to capturing incomplete data and displaying the network in a graphical mode. Tools and models presented in this section are not contradictory to these observations.

Dutta and Mitra [9] have designed a hybrid tool that integrates formal models for specifying problems (Lagrangian formulation), and heuristic models for evaluating and optimizing network performance. This leads to the most needed flexibility in both the processing of data and the application of constraints. For this purpose, this tool first divides the network into a core part (Backbone) and several other access networks. After this procedure, the designer limits their work to the topological design of the backbone only. They consider as data the following items: the location of the nodes, traffic between each node pair, allowed maximum delay, reliability, and cost requirements. From this information, the tool decides the topological configuration, routing, flow, and capacity assignment. All this is done by taking into account the constraints of the problem.

In this approach, the network representation has been practically neglected for the benefit of other topological design aspects. This is explained by the fact that the objective of the designers was essentially the optimization of network performance. Therefore, the resulting tool lacks both universality and interactive capabilities.

AUTONET [2] is a tool for analyzing the performance of a WAN that interconnects local networks. It is among the few tools that pay special attention to network representation. In fact, AUTONET provides the users with sophisticated means to design networks and perform modifications and adjustments required at any moment. It also offers a user-friendly interface suitable for various levels of users. One of its inconveniences remains the fact that it is dedicated exclusively to the modeling and representation of WAN, without taking into account other types of networks. Thus, the local networks, which could be linked to WAN, are considered as forming a single entity, and therefore their specific characteristics are neglected. The other inconvenience is that the capture of data is essentially oriented in performance evaluation, which is not necessarily compatible with the requirements of network representation. As a result, these data are incomplete and insufficient for adequately representing networks.

COMNET III [7] is the latest version of a series of tools produced by the CACI Products Company. This tool simulates and analyzes the performance of telecommunication networks, and offers the advantage of modeling and representing all types of network. COMNET III also integrates sophisticated graphical display options, allowing users to represent networks with a greater flexibility. It is available to users of all levels, from beginners to experts. In terms of universality, COMNET III is certainly better than the other tools previously mentioned. However, like AUTONET, it has some inconveniences.

For instance, it could be improved by both introducing other representation modes than graphical representation, and by adding data control procedures and user-customized representations.

OPNET (Optimized Network Engineering Tools), produced by MIL3 [26], is a simulation and analysis tool that allows, among other things, the simulation of large networks and detailed modeling of related protocols. It integrates several characteristics such as the specification of graphical models, dynamic simulation based on events, integrated data analysis, and hierarchical and object modeling. The specification domain offers several editors that allow the user to specify graphically the different components of a network. It also supplies an editor for programming finite states and specification parameters. The simulation domain is a library of procedures that facilitates the processing of packets to be transmitted. Finally, the analysis domain supplies the users with several tools including those that collect, graphically represent, and analyze data.

C. DESNET: An Example of a Design Tool

DESNET (*des*ign of *net*work) is a design tool developed by Pierre and Gharbi [30] for personal computers in a Windows environment. It offers the users the possibility of processing several design aspects such as routing, flow and capacity assignment, and calculation of network's cost and delay. It also integrates data manipulation interfaces that are understandable and accessible by any user who has a minimum knowledge of topological design.

1. Description of Functions

DESNET provides the user with a large number of functions. Its flexibility and simplicity facilitate adding several functions without affecting the robustness and reliability of the system. For organization reasons, we have decided to regroup the functions offered by DESNET in three classes according to their type, extent, and domains of application. We have defined a class for the definition of networks, another to represent them, and another for their handling.

Network Definition

This class of functions regroups all the applications that act as a single entity. The functions allow the user to select the network to be processed and permit the introduction of a new network in the system, or the deletion of an existing network.

Network Representation

The class of functions that describe network representation has the role of representing the network and its components in several manners, according to the user's needs. The user has the choice among a graphical representation, a matrix, or a list.

Network Management

This is the most important class of functions offered by DESNET. As shown in Fig. 4, this class regroups the most specific functions of the tool. Certain functions of this class are accessible at any time, when a network is open or

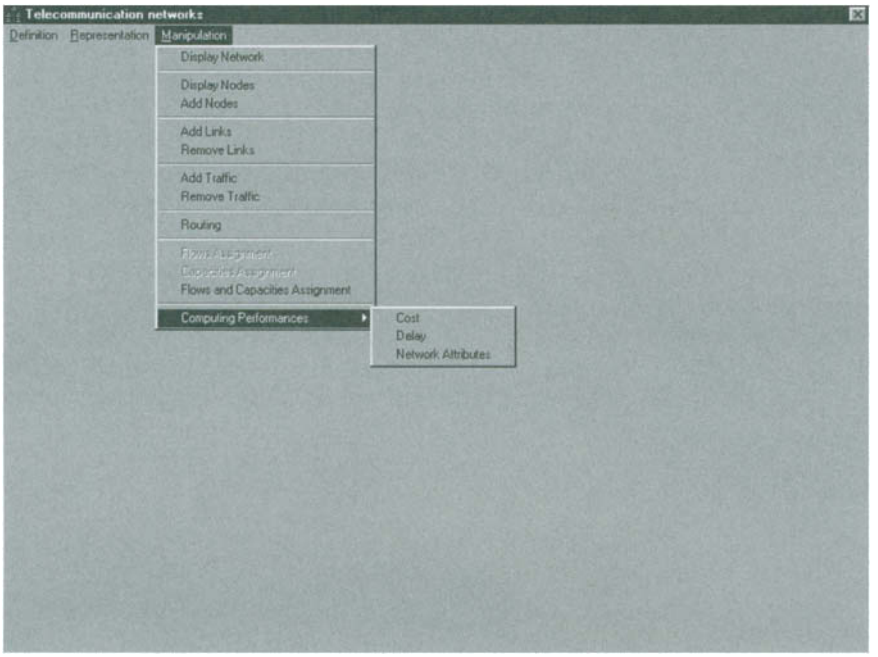


FIGURE 4 Network manipulation.

captured: this is the case of those related to network, nodes, links, and traffic. There are also certain other functions that are accessible once the routing is calculated: this is the case of flow and capacity assignment, and calculation of network performance.

2. Example of DESNET's Working

Let us consider a network that has 9 nodes representing 8 American cities and 12 links [22]. The coordinates of the nodes are represented in Table 1; they have been fixed according to a reference point on the upper left part of the screen. As shown in Fig. 5, this network is a WAN with a mesh topology. After the capture of data related to the network and its topological configuration, we have started to capture the network's nodes. Figure 6 illustrates the capture of the first node (NYK); one should note the presence of the *principal node* domain, reserved to a network configured as a star or a tree. In both cases, it is important for the system to know the main node of the star and the root of the tree. Because our network has a mesh configuration, this field is automatically disactivated by the system.

TABLE 1 Cartesian Coordinates of the 8 Nodes [22]

NODE	NYK	LSA	CHI	DAL	BAL	SFO	MIA	DEN
Abscissa	92	12	64	56	86	2	99	36
Ordinate	12	98	26	90	25	90	89	61

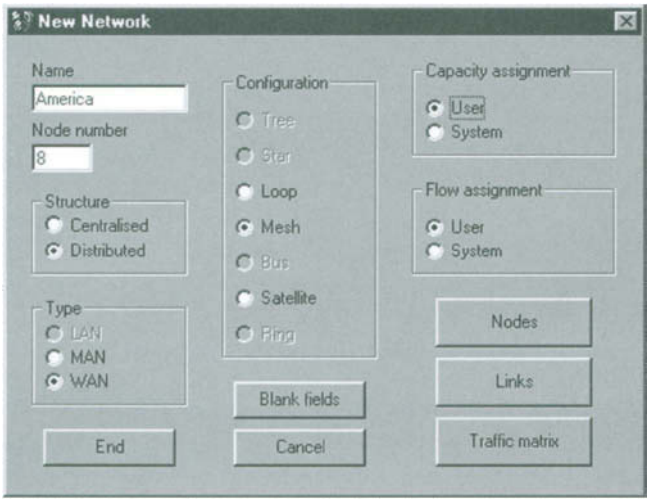


FIGURE 5 Data capture of network features and topological configuration.

Figure 7 represents the screen that captures a link. One should note on this screen that the domain *Cost* is not activated, because the cost of a link is calculated automatically and should be captured by the user. Similarly, for the nodes, DESNET does allow the user to capture the same link twice. After the capture of nodes and links, the user has the following choices: to capture the traffic, to confirm the capture, or to quit the system. For this example, we have chosen the capture of the traffic matrix. We have supposed that the traffic matrix is uniform with a constant value of 5 packets/s; the average size of packets is equal to 1000 bits. Figure 8 represents the capture of traffic between the nodes NYK and BAL.

Having confirmed the capture of these data, we can then access the data processing part. Figure 9 shows the graphical representation of this network. The numbers that appear beside the links represent their length in kilometers.

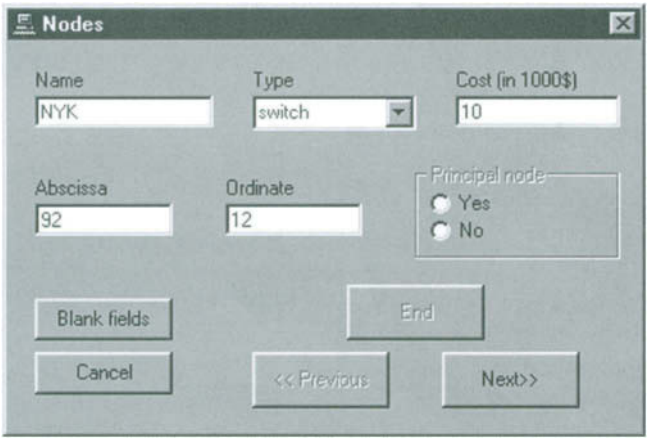


FIGURE 6 Data capture of the first node.

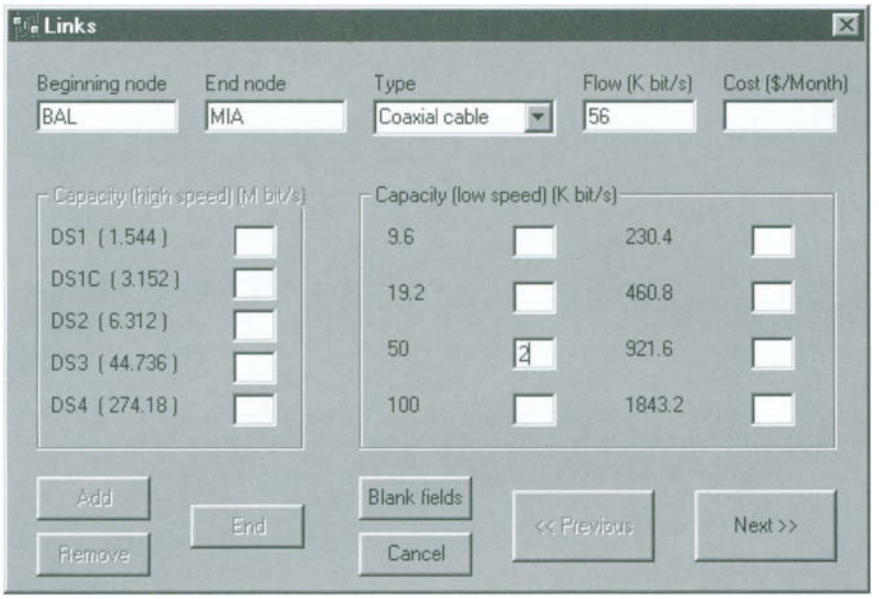


FIGURE 7 Data capture of the link BAL–MIA.

Furthermore, we have added to this network two new links called SEA and CANADA5. The node SEA has 5 and 20 as coordinates, while the coordinates of CANADA5 are 45 and 15. During the addition of one or several nodes, DESNET allows the user to choose the network's new name and to determine exactly the number of nodes needing to be added. The addition of a node is realized exactly in the same manner to that of the capture a node.

To link two new nodes added to an existing network, we have decided to add four links, which are in this case the links between SEA and SFO, SEA and CHI, SEA and CANADA5, and CANADA5 and NYK. As shown in Fig. 10, the addition of a link is considered a data capture. In effect, all control operations regarding the network's connectivity and integrity of data used by DESNET during the capture of links are also applicable upon the addition. The user has

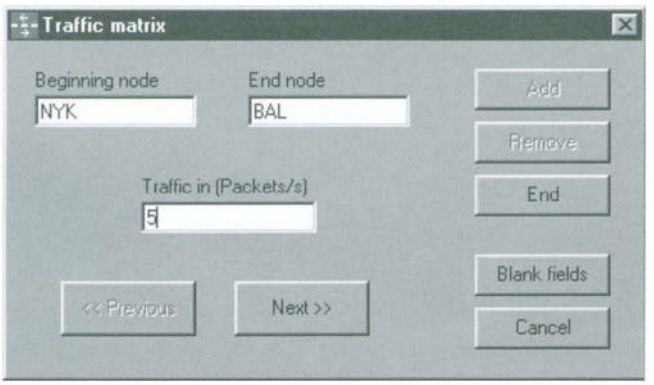


FIGURE 8 Data capture of the traffic between NYK and BAL.

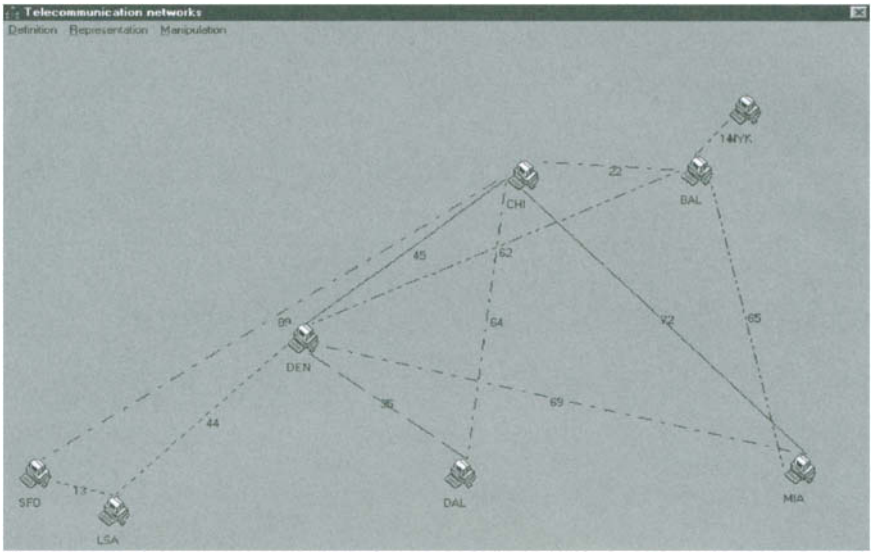


FIGURE 9 Graphical representation of the network.

the right to add as many links as she/he likes, provided that the maximum number of links $m_{\max}$ is not exceeded. For the current example, there are a maximum of 45 links.

After adding the nodes, it is convenient to add the traffic between these new nodes and the remaining network nodes. In order to uniformly maintain the traffic matrix, the traffic between each new node pair is maintained at 5 packets/s; the average size of packets is still 1000 bits. Figure 11 represents the network obtained after the addition of SEA and CANADA5 nodes. The new network contains 10 nodes and 16 links.

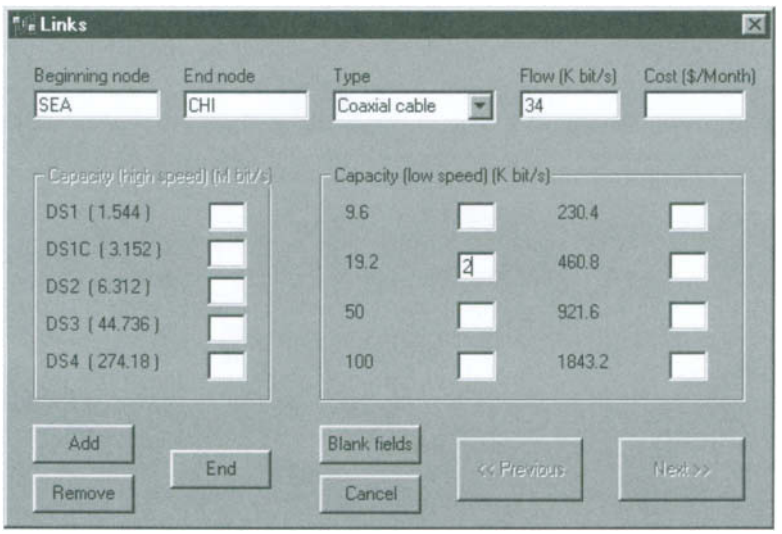


FIGURE 10 Addition of a link between SEA and CHI.

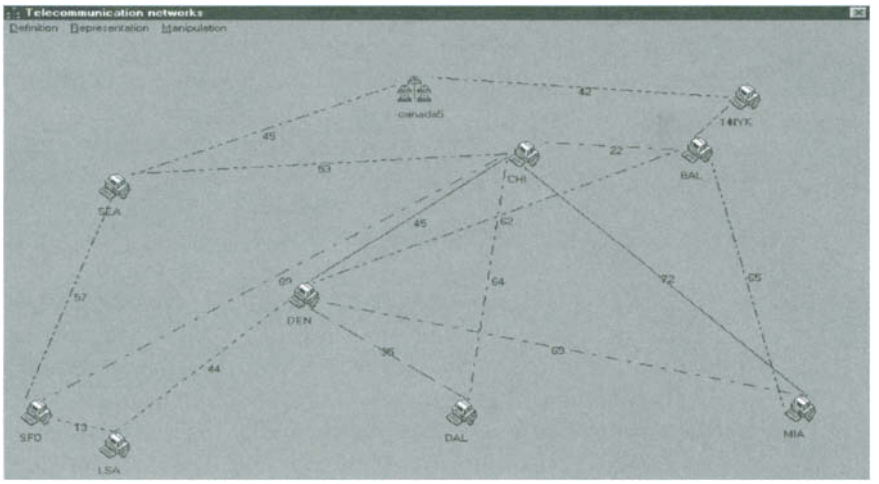


FIGURE 11 New network graphical representation.

IV. INTELLIGENT AND HYBRID APPROACHES

A new tendency emerges in the use of AI approaches for designing data communication network topologies. Among others, Pierre [37] proposed the system called SIDRO, which starts with an initial topology according to user specifications. This initial topology is then submitted to a set of rules that determine the choice of perturbation types in order to reduce the total link cost or the average delay. According to this choice, SIDRO applies perturbation rules for generating examples that are kept in an example base and used elsewhere by a learning module to generate new rules.

A. Basic Principle of the AI-Based Approach

The approach presented in this section consists of perturbating a starting topology by applying sequences of rules and metarules, handled by an example generator, in order to reduce the total cost of the links and/or to improve the average delay of this network topology [36,37]. This *example generator* consists of a rule base, an inference engine, and an example base. For efficiency and performance purposes, the rule base component is subdivided into three hierarchical levels: (1) perturbation selecting rules (PSR), (2) perturbation rules (PR), and finally (3) positive or negative example defining rules (PNR). A perturbation selecting rule is a metarule that indicates the type of perturbation rule to be applied, depending on the specific context described by the premises of this rule. It is also a generic concept employed to point out a link addition rule, a link deletion rule, or a link substitution rule.

A perturbation cycle can be defined as the process by which positive and negative examples are generated by applying all the perturbation rules corresponding to a given perturbation selecting rule to a specific starting topology. After each perturbation cycle, positive or negative example defining rules are then used for labeling the examples generated by perturbation: positive

examples (e^+) are feasible topologies in the sense they satisfy all the specified constraints, whereas negative examples (e^-) refer to topologies where only the delay constraint is not satisfied. The solution of the current problem is then the least-cost positive example, which has been generated up to the last perturbation cycle. In this way, perturbation cycles that aim at improving the current solutions can be viewed as a method for generalizing a local search process. After a certain number of perturbation cycles, feasible solutions or improvements to current solutions cannot be obtained by starting new cycles. As a result, refining current perturbation rules and discovering new ones from examples that have been previously generated constitute the only ways for improving solutions. This can be done by means of machine learning mechanisms [4,5,8,27]. The integration of an inductive learning module into the basic knowledge-based system has been considered as an extension of the example generator. In this chapter, we are interested in the specific problem of inferring new design rules that could reduce the cost of the network, as well as the message delay below some acceptable threshold.

B. Basic Concepts and Background

A network topology can be characterized by three types of descriptors: local, global, and structural [31]. These descriptors will be used either to state the topological design problem or to describe the machine learning module integrated into SIDRO.

1. Local Descriptors

A *local descriptor* essentially characterizes one node or one link of a network. This category of descriptors includes the length of a link, the incidence degree of a node, the traffic index of a link, the flow of a link, the capacity of a link, the utilization ratio of a link, and the excess cost index of a link. Except for the last one, all these concepts have already been defined in Section A of Section II.

The excess cost of a link k can be defined as $E_k = d_k(C_k - f_k)/C_k$. This relation can be rewritten as $E_k = d_k(1 - U_k)$. The average excess cost E of a network can be defined as $E = D_m(1 - U)$, where D_m is the average cost of the links, and U a mean utilization ratio. The *excess cost index* P_k can now be defined, for each link k of a given network topology, as

$$P_k = E_k/E = [d_k(1 - U_k)]/[D\,(1 - U)].$$

Clearly, if the excess cost index P_k of a link is greater than 1, the excess cost of that link is greater than the average excess cost of the network.

2. Some Global and Structural Descriptors

Global descriptors are features whose scope covers the whole network. For instance, the following parameters are global descriptors: number of nodes, number of links, total cost of the links, connectivity degree, normalized average delay, diameter, incidence degree of the network, and average traffic of the network.

Structural (or semantic) descriptors essentially consist of classic concepts derived from graph theory or conventional network design techniques [12,13].

Stated at an intermediary level of characterization between local and global descriptors, the structural descriptors refer to features shared by a subset of nodes or links. This category of descriptors includes a subgraph of a given degree, a subgraph of a given diameter, a saturated cut set, the least loaded path between two nodes, the fastest path between two nodes, and the most economical path between two nodes.

The structural descriptor *subgraph with a given degree* is obtained by associating the descriptors' *set of links incident on a node* and *subgraph* with the global descriptor *incidence degree of a graph*. The structural descriptor *subgraph of diameter* R_S is obtained by combining the descriptors *subgraph* and *diameter;* it is very relevant in a process of average delay reduction. A *saturated cut* is the minimal set of highly utilized links ($U_k > 0.9$) such that the removal of these links will split the network into two connected components.

The residual capacity of a link k is defined as the difference between the flow and the capacity of this link ($C_k - f_k$). It follows that the residual capacity of a path corresponds to the minimum residual capacity measured over the links of this path. Therefore, the structural descriptor *least loaded path between two nodes* can be defined as the path having the highest residual capacity between these two nodes. On the other hand, each link k of a path introduces a specific mean delay. The delay of a path is obtained by summing the mean delay values of its links. Thus, the *fastest path between two nodes* corresponds to the path having the lowest delay between these nodes. This structural descriptor can be computed using a shortest path search procedure, which takes the mean link delay as metric. Finally, the cost of a path is simply the total cost of the links belonging to this path. Therefore, the structural descriptor *most economical path between two nodes* can be computed using a shortest path search procedure, by taking the link cost as metric.

C. The Inductive Learning Module

Inductive learning is defined as the acquisition of knowledge by means of inductive inference performed from facts that are provided by a teacher or by the environment. It is often considered as a heuristic search through a space of symbolic descriptions called inductive assertions [8,27]. The inductive learning module developed in this section aims at inferring new perturbation rules from examples stored in the example base of SIDRO [37].

1. Syntax of the Perturbation Rules

Perturbation rules to be inferred are structured according to one of the following two formats [37]:

Format 1. Cost reduction objective:

If (1) $P_1(x_1)$
(2) $P_2(x_2)$
(3) $P_3(x_3)$
(4) $P_4(x_4)$
(5) $P_5(x_5)$
Then $Q_i(h_{ij}, O)$ can lead to a cost reduction (CV: α).

Format 2. Delay constraint restoration objective:

If (1) $P_1(x_1)$
(2) $P_2(x_2)$
(3) $P_3(x_3)$
(4) $P_4(x_4)$
(5) $P_5(x_5)$
Then $Q_i(h_{ij}, O)$ can lead to the delay constraint restoration (CV: α).

$P_i(x_i)$ denotes a premise built with some descriptors, and $Q_i(h_{ij}, O) = p_k$ a perturbation operator built with a hypothesis h_{ij}, where $i = 1$ for a link addition, $i = 2$ for a link deletion, and $i = 3$ for a link substitution operator. The index j specifies a hypothesis, that is, a basic criterion or a combination of basic criteria that can be used for selecting potential links to be perturbed. Clearly, because of the great number of possible ways to perturb a network topology, it is unrealistic to adopt an exhaustive approach.

Once a perturbation operator $Q_i(h_{ij}, O)$ is applied to an example in a learning process, the objective indicator O takes the value 1 if the aimed objective is reached, and 0 otherwise. This operation is successively applied to a great deal of examples. The likelihood factor CV is then computed as the success rate of applying this perturbation operator to all the considered examples. For a CV greater than or equal to 0.85, a new rule is inferred and built with the specific operator $Q_i(h_{ij}, O)$.

For the same i in h_{ij}, it can happen that more than one hypothesis leads to the searched objective: cost reduction or delay constraint restoration. As a result, a hypothesis preference criterion is needed for selecting only one hypothesis from the appropriate space indexed by i. In the context of topological design, the hypothesis preference criterion (HPC) chosen is the *best cost-performance ratio;* it implies choosing the hypothesis leading to the greatest total link cost reduction, while restoring or maintaining the delay constraint. According to this HPC, with a positive example as seed (cost reduction objective), the only hypothesis h_{ij} that can be selected is the one that enables the best cost reduction and preserves delay constraint. Conversely, with a negative example as seed (delay constraint restoration objective), HPC allows only the hypothesis enabling the restoration of the delay constraint and the highest cost reduction.

2. Notation Used for Describing the Learning Algorithm

Here is the list of notations used to describe the learning algorithm, which intrinsically attempts to infer new perturbation rules leading to cost reduction or delay constraint restoration:

- E Current example base related to a given problem being solved; it is also called *short-term example base.*
- X Example base related to all problems already solved by the example generator; it is also called *long-term example base.*
- X_c Compressed long-term example base resulting from removing superfluous or redundant examples from X.

e_g Example (positive or negative) extracted from E and used as the starting example for the learning process; it is also called a *seed*.
R_b Current rule base.
r_t Tentative rule inferred (not validated yet).

This learning algorithm is inspired by the well-known Star methodology [27] and described by the following steps:

Step 0. Initialization:

$$X = X \cup E$$
$$X_c = \phi.$$

Step 1. Select at random in E a seed e_g.

Step 2. From the number of nodes n, the degree of connectivity K, the normalized average delay T_n, and the diameter R of the seed e_g, obtain the premises $P_1(x_1)$, $P_2(x_2)$, $P_3(x_3)$, $P_4(x_4)$.

Step 3. Determine (i, j), with $i = 1, 2, 3$ and $j = 1, 2, \ldots, q_i$ (q_i = number of hypotheses of type i) such that applying the operator $Q_i(h_{ij}, O)$ to e_g results in a cost reduction if e_g is a positive example, or in the restoration of the delay constraint if e_g is a negative example.

Step 4. In the case of more than one acceptable operator $Q_i(h_{ij}, O)$, turn to the hypothesis preference criterion (the best cost-performance ratio) to select the best conclusion for a new rule. Then infer from this the value of $x_5(x_5 = i)$ specifying the premise $P_5(x_5)$, together with the tentative rule $r_t = (P_1, P_2, P_3, P_4, P_5, i, j, Q_i(h_{ij}, O), 1)$.

Step 5. Apply r_t to examples stored in X in order to validate this tentative rule, that is, to verify that its application effectively results in a cost reduction or in the delay constraint restoration, depending on the features of the considered seed e_g. Then compute the value of the likelihood factor CV as the ratio υ_s/υ_t, where υ_s denotes the number of successful applications of r_t, and υ_t the total number of applications of this tentative rule to the suitable examples taken from X.

Step 6. If r_t covers at least 85% of examples considered in X (CV ≥ 0.85), then it becomes a valid discovered rule to be inserted in the rule base R_b, and go to Step 8.

Step 7. Reduce the short-term example base E to the only examples not covered by the tentative rule r_t and return to Step 1.

Step 8. Update the long-term example base by keeping only nonredundant examples which constitute X_c ($X = X_c$).

3. Illustrative Example

Once the seed e_g is selected at Step 1, a premise-building procedure is used at Step 2 for extracting the number of nodes n, the connectivity degree K, the normalized average delay T_n, and the diameter R. For instance, for a seed e_g characterized by the values of parameters $n = 20$, $K = 5$, $T_n = 0.7$, $R = 0.95L_{max}$, this procedure builds the following premises:

(P_1) The topology has a number of nodes greater than 7 ($K \leq n-1 \Rightarrow n > K+2 = 7$)
(P_2) The degree of connectivity is equal to 5 ($K = 5$)
(P_3) The normalized average delay is less than or equal to 1 ($T_n = 0.7 < 1$)
(P_4) The diameter of the topology is less than or equal to L_{max} ($R = 0.95L_{max} < L_{max}$).

Since the normalized average delay T_n is less than 1, the seed e_g is a positive example: only cost reduction constitutes a relevant objective. Specifically, Step 3 consists of searching heuristicly through the hypothesis spaces H_i for the hypothesis h_{ij}, with $i = 1, 2, 3$ and $j = 1, 2, \ldots, q_i$, such that applying operator $Q_i(h_{ij})$ to the seed e_g leads to a cost reduction. Assume that the preferred hypothesis is

$$h_{29} = (k \in \mathrm{CMD}_{ij}) \wedge ([k = (i, j)] \wedge [d(i) > K] \wedge [d(j) > K]).$$

In accordance with Step 4, the result is $i = 2$ and $j = 9$. It follows that $x_5 = 2$, $r_t = (P_1, P_2, P_3, P_4, P_5, 2, 9, Q_2(h_{29}, 0), 1)$. Since $i = 2$, (P_5) can be paraphrased as follows: "The envisioned perturbation is a link deletion."

At Step 5, the tentative rule r_t is applied to examples stored in X in order to compute the likelihood factor CV. In accordance with Step 6, if the success rate is at least equal to 85%, the tentative rule is considered as a valid rule discovered by the inductive learning module. Otherwise, the short-term example base is updated by removing all examples covered by r_t.

In a postprocessing phase, the rule r_t is paraphrased as follows:

(r_t) IF (1) THE TOPOLOGY HAS A NUMBER OF NODES GREATER THAN 7
(2) THE DEGREE OF CONNECTIVITY IS EQUAL TO 5
(3) THE NORMALIZED AVERAGE DELAY OF THE TOPOLOGY IS LESS THAN OR EQUAL TO 1
(4) THE DIAMETER OF THE TOPOLOGY IS LESS THAN OR EQUAL TO L_{max}
(5) THE ENVISIONED PERTURBATION IS A LINK DELETION
THEN THE DELETION OF LINKS $k = (i, j)$ BELONGING TO THE LEAST LOADED PATH BETWEEN i AND j, SUCH THAT $d(i)$ AND $d(j)$ REMAIN GREATER THAN OR EQUAL TO 5 AFTER THE DELETION, CAN LEAD TO A COST REDUCTION (CV:0.85)

Table 2 summarizes and paraphrases a sample of rules discovered by the inductive learning module, according to a process detailed in [36].

D. Numerical Applications with SIDRO

Consider the problem of finding a 3-connected topology which guarantees a maximum allowable delay of 50 ms per packet, starting from the 20-node initial topology shown in Fig. 12. The traffic between each node pair is equal

TABLE 2 Sample of Inferred Rules

(r_5)	IF	(1) THE TOPOLOGY HAS A NUMBER OF NODES GREATER THAN 6 (2) THE DEGREE OF CONNECTIVITY IS EQUAL TO 4 (3) THE NORMALIZED AVERAGE DELAY IS LESS THAN OR EQUAL TO 1 (4) THE DIAMETER OF THE TOPOLOGY IS LESS THAN OR EQUAL TO L_{max} (5) THE ENVISIONED PERTURBATION IS A LINK DELETION
	THEN	THE DELETION OF LINKS $k = (i, j)$ SUCH THAT $L(k) > 0.6^* L_{max}$, k BEING IN A SUBGRAPH OF WHICH THE DEGREE IS GREATER THAN 3, $d(i)$ AND $d(j)$ REMAINING GREATER OR EQUAL TO 3 AFTER THE LINK DELETION, CAN LEAD TO A COST REDUCTION (CV: 0.85).
(r_{11})	IF	(1) THE TOPOLOGY HAS A NUMBER OF NODES GREATER THAN 5 (2) THE DEGREE OF CONNECTIVITY IS EQUAL TO 3 (3) THE NORMALIZED AVERAGE DELAY IS GREATER THAN 1 (4) THE DIAMETER OF THE TOPOLOGY IS LESS THAN OR EQUAL TO L_{max} (5) THE ENVISIONED PERTURBATION IS A LINK ADDITION
	THEN	THE ADDITION OF LINKS $k = (i, j)$ INCIDENT TO A NODE BELONGING TO THE SATURATED CUT-SET, AND SUCH THAT ITS TRAFFIC INDEX IS GREATER THAN 1, CAN LEAD TO THE DELAY CONSTRAINT RESTORATION (CV: 0.88).
(r_{18})	IF	(1) THE TOPOLOGY HAS A NUMBER OF NODES GREATER THAN 7 (2) THE DEGREE OF CONNECTIVITY IS EQUAL TO 5 (3) THE NORMALIZED AVERAGE DELAY IS GREATER THAN 1 (4) THE DIAMETER OF THE TOPOLOGY IS GREATER THAN L_{max} (5) THE ENVISIONED PERTURBATION IS A LINK SUBSTITUTION
	THEN	THE REPLACEMENT OF LINKS $k = (i, j)$ SUCH THAT $L(k) > 0.7^* L_{max}$, BY TWO OTHER LINKS $v = (i, p)$ AND $w = (j, q)$ BOTH INCIDENT TO NODES BELONGING TO THE FASTEST PATH BETWEEN i AND j, WITH $L(v)$ AND $L(w)$ BOTH LESS THAN $0.8^* L(k)$, CAN LEAD TO THE DELAY CONSTRAINT RESTORATION (CV:0.92).
(r_{19})	IF	(1) THE TOPOLOGY HAS A NUMBER OF NODES GREATER THAN 5 (2) THE DEGREE OF CONNECTIVITY IS EQUAL TO 3 (3) THE NORMALIZED AVERAGE DELAY IS LESS THAN OR EQUAL TO 1 (4) THE DIAMETER OF THE TOPOLOGY IS LESS THAN OR EQUAL TO L_{max} (5) THE ENVISIONED PERTURBATION IS A LINK SUBSTITUTION
	THEN	THE REPLACEMENT OF LINKS $k = (i, j)$ OF WHICH UTILIZATION RATIO IS LESS THAN 0.8, BY TWO OTHER LINKS $v = (i, p)$ AND $w = (j, q)$, THE SHORTEST POSSIBLE, BOTH HAVING THEIR TRAFFIC INDICES GREATER THAN 1 CAN LEAD TO A COST REDUCTION (CV: 0.87).
(r_{23})	IF	(1) THE TOPOLOGY HAS A NUMBER OF NODES GREATER THAN 7 (2) THE DEGREE OF CONNECTIVITY IS EQUAL TO 5 (3) THE NORMALIZED AVERAGE DELAY IS LESS THAN OR EQUAL TO 1 (4) THE DIAMETER OF THE TOPOLOGY IS LESS THAN OR EQUAL TO L_{max} (5) THE ENVISIONED PERTURBATION IS A LINK DELETION
	THEN	THE DELETION OF LINKS $k = (i, j)$ SUCH THAT $L(k) > 0.7^* L_{max}$, $d(i)$ AND $d(j)$ REMAINING GREATER OR EQUAL TO 5 AFTER THE LINK DELETION, AND THE EXCESS COST INDEX OF k IS LESS THAN 1, CAN LEAD TO A COST REDUCTION (CV: 0.87).

to 10 packets/s, and the average packet length has been taken to be 1000 bits. Table 3 gives the Cartesian coordinates of the 20 nodes. Table 4 shows the costs associated with the link capacity options. The total link cost D of this topology is equal to \$197,776/month. Figure 13 shows the solution resulting from the application of some addition and deletion rules of Table 2 during three perturbation cycles. We observe a cost reduction of 9.2% per month, whereas the delay constraint has been preserved.

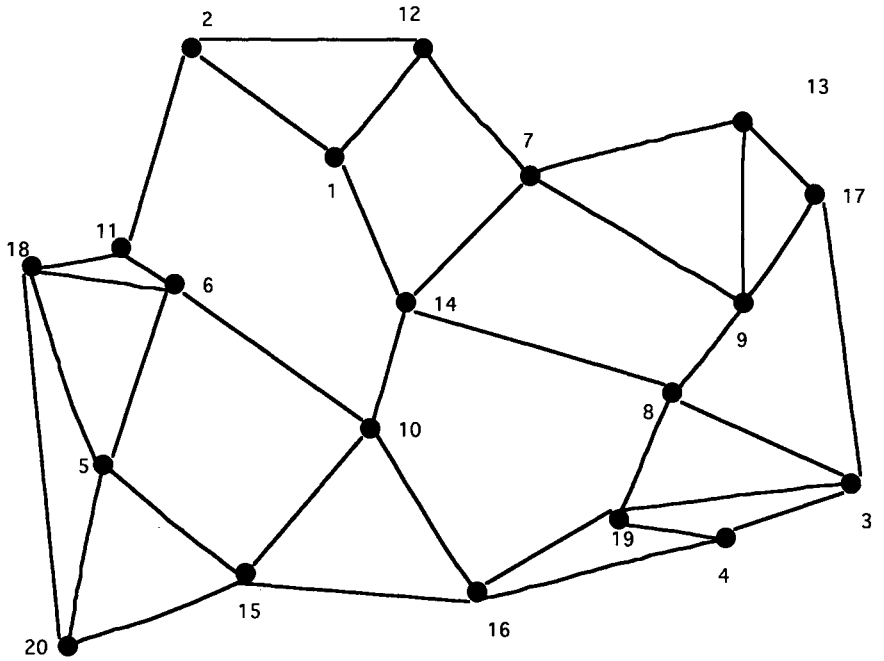


FIGURE 12 A 20-node initial topology (D = \$197,776/month and T = 42.4 ms).

V. HEURISTIC APPROACHES

Various heuristic methods for solving the topological design problem, in the context of packet-switched networks, have been proposed. These methods are generally incremental in the sense that they start with an initial topology and perturb it repeatedly until they produce suboptimal solutions.

A. Conventional Heuristics and Meta-heuristics

BXC starts from an arbitrary topological configuration generated by a user or a design program to reach a local minimum by means of local transformations.

TABLE 3 Cartesian Coordinates of the Nodes (X_i, Y_i)

i	X_i	Y_i	i	X_i	Y_i
1	250	360	2	165	420
3	600	100	4	480	55
5	90	130	6	150	250
7	400	325	8	435	185
9	530	250	10	275	165
11	100	300	12	350	420
13	550	360	14	320	240
15	205	85	16	365	40
17	595	310	18	15	260
19	415	80	20	50	25

TABLE 4 Costs Associated with the Link Capacity Options

Capacity (Kbps)	Variable cost (($/month)/Km)	Fixed cost ($/month)
9.6	3.0	10.0
19.2	5.0	12.0
56.0	10.0	15.0
100.0	15.0	20.0
200.0	25.0	25.0
560.0	90.0	60.0

A local transformation often called *Branch X-change* consists of the elimination of one or more old links and the insertion of one or more new links to preserve the 2-connectivity. This technique requires an exhaustive exploration of all local topological exchanges and tends to be time consuming when applied to networks with more than 20 or 30 nodes [12].

CBE starts with a fully connected topology using concave costs and applies the FD algorithm until it reaches the local minimum [12]. The FD algorithm eliminates uneconomical links and strongly reduces the topology. This algorithm is terminated whenever the next link removal violates the constraint of 2-connectivity: the last 2-connected solution is then assumed to be the local minimum. CBE can efficiently eliminate uneconomical links, but does not allow for the insertion of new links. In order to overcome such limitations, the

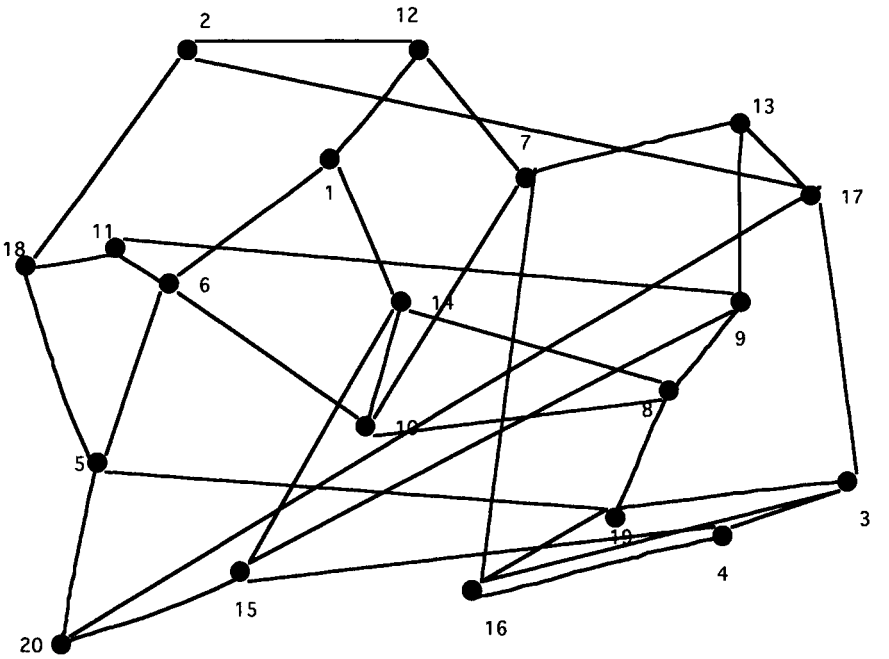


FIGURE 13 New topology generated by SIDRO ($D = \$179{,}620$/month and $T = 44.76$ ms).

CS method has been proposed [13]. This method is iterative and consists of three main steps:

1. Find the saturated cut, that is, assess the minimal set of the most utilized links that, if removed, leaves the network disconnected.
2. Add new links across the cut in order to connect the two components.
3. Allow the removal of the least utilized links.

CS can be considered as an extension of BXC, in the sense that, rather than exhaustively performing all possible branch exchanges, it selects only those exchanges that are likely to improve throughput and cost.

MENTOR has been proposed by Kershenbaum *et al.* [21]. It essentially tries to find a distributed network with all the following characteristics: (i) traffic requirements are routed on relatively direct paths; (ii) links have a reasonable utilization; (iii) relatively high-capacity links are used, thereby allowing us to benefit from the economy of scale generally present in the relationship between capacity and cost. These three objectives are, to some extent, contradictory. Nevertheless, the MENTOR algorithm trades them off against one another to create low-cost networks.

Another solving approach consists of using combinatorial optimization meta-heuristic methods, such as SA and GA. The SA method starts by choosing an arbitrary initial solution, then searches, in the set of neighbor solutions, a new solution that, hopefully, improves the cost.

SA is a process whereby candidate solutions to a problem are repeatedly evaluated according to some objective function and incrementally changed to achieve better solutions [23,24,44]. The nature of each individual change is probabilistic in the sense that there is some probability it worsens the solution. In addition, an annealing schedule is followed whereby the probability of allowing a change that worsens the solution is gradually reduced to 0 [38]. If a better solution is found, then it becomes the current solution; if not, the method stops and at this step a local optima is reached [24].

This method has been applied to many combinatorial optimization problems [6]. Pierre *et al.* [35] adapted this method to solve the problem of topological design of packet-switched networks. This adaption consists of starting with an initial topology that satisfies the reliability constraint, then applying the SA algorithm with an initial high value of the temperature parameter, in order to obtain a new configuration that minimizes the total link cost or improves the mean delay.

The Genetic Algorithms (GA) have been introduced by Holland [17]. They are inspired by the Darwin's Model and based on the survival of the fittest species. Just as in nature where specimens reproduce themselves, in genetic algorithms specimens also reproduce themselves. GAs are essentially characterized by the coding of the problem parameters, the solution space, the evaluation function, and the way of choosing chromosomes to be perturbed. In practice, from a generation to another, chromosomes that form the population have a very high aptitude value. GAs start generally with a population generated randomly. To undertake an efficient search of performing structures, genetic operators are applied to this initial population in order to produce, within a time limit, successive high-quality populations. We distinguish generally four main genetic operators: reproduction, crossover, mutation, and inversion [17].

Pierre and Legault [32,33] adapted GAs for configuring economical packet-switched computer networks that satisfy some constraints related to quality of service. The adaptation results show that the GA method can produce good solutions by being applied to networks of 15 nodes and more.

An hybrid method has been introduced by Dutta and Mitra [9]. This method integrates both the algorithmic approach and a knowledge-based system. It builds a solution by subdividing the topological design problem into modules that can be individually solved by applying optimization models or heuristic methods. Furthermore, it integrates the partial solutions to obtain a global solution to the design problem. The system is made up of independent modules that share a common blackboard structure. It usually provides good solutions with minimal costs.

B. Implementation of the Tabu Search Approach

TS is an iterative improvement procedure that starts from an initial feasible solution and attempts to determine a better solution in the manner of an ordinary (descent) local method, until a local optimum is reached [14,15]. This method can be used to guide any process that employs a set of moves for transforming one solution into another; thus it provides an evaluation function for measuring the attractiveness of these moves [14,16].

1. Basic Principles

For a given combinatorial optimization problem, a solution space S may be defined as the set of all feasible solutions. TS associates with each feasible solution a numerical value that may be considered as the cost of the solution obtained by optimizing the cost function. For each solution s_i in the space solution S, there is a subset of S, say $N(s_i)$, considered as a *neigborhood* of s_i. This subset contains a set of feasible solutions that may be reached from s_i in one move. TS starts from an initial feasible solution s_i, and then moves to a solution s_j, which is also an element of $N(s_i)$. This process is repeated iteratively, and solutions that yield cost values lower than those previously encountered are recorded. The final cost recorded, when the search is interrupted, constitutes the overall optimum solution. Thus, TS can be viewed as a variable neighborhood method: each step redefines the neighborhood for which the next solution will be drawn.

A move from s_i to s_j is made on the basis that s_j ($s_i \neq s_j$) has the minimum cost among all the allowable solution in $N(s_i)$. Allowability is managed by a mechanism that involves historical information about moves made while the procedure progresses. TS is a high-level procedure that can be used for solving optimization problems; it escapes the trap of local optimality by using short-term memory recording of the most recently visited solutions [16]. The short-term memory constitutes a form of aggressive exploration that seeks to make the best move possible satisfying certain constraints. These constraints are designed to prevent repetition of some moves considered as forbidden (tabu). Such attributes are maintained in a *tabu list* (LT).

TS permits backtracking to previous solutions which may ultimately lead, via a different direction, to better solutions. This flexibility element is implemented through a mechanism called *aspiration criteria* [16]. The goal of aspiration criteria is to increase the flexibility of the algorithm while preserving the

Given

X : *Space of feasible solutions*

F : *Objective function,* $X \rightarrow R$, *defined on* X

N(s) : *Neighbourhood of* $s \in X$

$|LT_i|$: *Length of each tabu list* LT_i

f* : *Lower bound of objective function*

nbmax : *Maximum number of iterations between two improvements of* s^*

Initialization

Choose by any heuristic an initial solution $s \in X$

$s^* := s$ (s^* *best solution obtained*)

nbiter := 0 (*Iteration counter*)

bestiter := 0 (*iteration given the last* s^*)

Initialize LT_i for each i

$A_j(s, m) = +\infty$

While (f(s) > f*) and (nbiter - bestiter < nbmax) **DO**

nbiter := nbiter +1

generate a sample $V^* \subseteq N(s)$ of neighbour solutions

if [*one of the tabu conditions is violated,* i.e. $t_i(s,m) \in LT_i$] or [*at least one of aspiration criteria conditions is verified,* i.e. $a_j(s,m) < A_j(s,m)$] then

Choose the best $s' \in V^*$ minimizing f on V^* (*by a heuristic*)

If f(s') < f(s*) **Then** s* := s'

bestiter := nbiter

Update of tabu lists LT_i

Update of threshold values $A_j(s,m)$

s := s'

Endwhile

Result s* : best solution found after the execution of the procedure

FIGURE 14 General version of TS.

basic features that allow the algorithm to escape local optima and avoid cyclic behavior. Figure 14 presents a general version of TS.

2. Definition of the Moves

Adaptation of TS to a specific problem essentially relies on the definition of moves that describe the neighborhood to be explored. In the case of topological design of communication networks, some moves or local transformations called perturbations are applied to a starting topology in order to reduce its total link cost and/or to improve its average packet delay. These perturbations deal with *addition, removal, and substitution* of links.

The main goal of removal moves is to reduce the total link cost. If a starting topology contains the minimum number of links to preserve the K-connectivity, then it is impossible to apply to it a removal move because the connectivity constraint would be automatically violated. In our implementation, we defined three types of removal moves:

M_1^R the removal of links $k = (i, j)$ such that $L(k) > \alpha^* L_{max}$, with $0 < \alpha \leq 1$, $d(i)$ and $d(j)$ remaining at least equal to the desired connectivity degree after the link removal;

M_2^R the removal of links $k = (i, j)$ such that the excess index cost P_k is less than 1, $d(i)$ and $d(j)$ remaining at least equal to the desired connectivity degree after the link removal;

M_3^R the removal of links $k = (i, j)$ such that the utilization rate U_k is less than α, with $0 < \alpha \leq 1$, $d(i)$ and $d(j)$ remaining at least equal to the desired connectivity degree after the link removal.

The addition moves cannot violate the connectivity constraint, but generally have negative effects on the cost. They are defined as follows:

M_1^A to each topology that has a normalized delay (T_n) greater than 1, add the links $k = (i, j)$ that have an index traffic greater than 1;

M_2^A to each topology that has a normalized delay (T_n) less than 1, add the links $k = (i, j)$ such that $L(k) < \alpha^* R$, with $0 < \alpha \leq 1$.

The substitution moves correspond to a sequence of removals and additions of links. These moves, which cannot guarantee the preservation of the connectivity degree, are defined as follows:

M_1^S the substitution of all links $k = (i, j)$ such that $L(k) > \alpha^* L_{max}$, with $0 < \alpha \leq 1$, by two other links $v = (i, p)$ and $w = (j, q)$, with $L(v) \leq \alpha_1 L(k)$ and $L(w) \leq \alpha_2 L(k)$, $0 < \alpha_1 \leq 1$ and $0 < \alpha_2 \leq 1$;

M_2^S the substitution of all link $k = (i, j)$ such that $U_k > \alpha$, with $0 < \alpha \leq 1$, by two other links $v = (i, p)$ and $w = (j, q)$ with $I_{ip} > 1$ and $I_{iq} > 1$.

The neighborhood of a current topology s is a finite set $N(s)$ of topologies that are said to be feasible. In our implementation, this neighborhood cannot contain more than 6 topologies, each of which is obtained from the current topology by applying one move. The length of the tabu list is fixed to 7. The average packet delay is calculated using Eq. (1), and the total link cost is computed using Eqs. (2) and (3).

C. Numerical Applications with Heuristic Methods

In order to evaluate the effectiveness and efficiency of TS approach, we have considered a set of 20 nodes defined by the Cartesian coordinates given in Table 5. The node coordinates give an Euclidean representation of the network and are used to determine the distance between each pair of nodes or the length

TABLE 5 Node Coordinates

Node	1	2	3	4	5	6	7	8	9	10
Abscissa	63	22	41	33	32	40	52	80	19	15
Ordinate	8	72	45	10	84	78	52	33	35	81
Node	11	12	13	14	15	16	17	18	19	20
Abscissa	27	27	70	48	4	10	56	82	9	95
Ordinate	3	16	96	4	73	71	27	47	54	61

TABLE 6 Capacity Options and Costs

Capacity (Kbps)	Fixed cost ($/month)	Variable cost ($/month/Km)
9.60	650.00	0.40
19.20	850.00	2.50
50.00	850.00	7.50
100.00	1700.00	10.00
230.40	2350.00	30.00
460.80	4700.00	60.00
921.60	9400.00	120.00
1843.20	18800.00	240.00

of each link; the selection of moves to be performed at each step is based on the link length, among others.

The capacity options are given in Table 6, and the traffic matrix is provided in Table 7. The maximum acceptable delay is $T_{\max} = 250$ ms, the degree of connectivity is equal to 3, and the average packet length is 1000 bits.

Starting with the above specification data, step 1 of this method generates an initial topology with connectivity degree at least equal to 3. In step 1, the operation that has a higher cost in terms of CPU time and memory requirements remains the computation of link lengths. The computational complexity of this step is $O(n^2)$. Figure 15 gives the initial topology, while Table 4 provides the link attribute values of this topology.

Table 8 shows that the links of the initial topology are among the shortest possible links. The total link cost of this configuration is $D =$ $124,222.98/month and its average delay is evaluated at 81.84 ms. To this

TABLE 7 Traffic Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	4	7	8	6	9	5	5	9	3	9	3	6	4	3	4	1	7	7	10
2	4	0	9	3	2	8	3	1	2	9	8	10	6	10	3	6	10	2	6	3
3	7	9	0	7	10	10	6	5	5	10	6	8	8	9	1	2	4	4	10	9
4	8	3	7	0	9	6	5	4	7	3	7	6	8	5	9	4	2	10	3	5
5	6	2	10	9	0	8	1	3	8	6	6	7	6	3	3	1	6	5	4	2
6	9	8	10	6	8	0	10	9	6	7	1	5	1	5	9	6	10	5	4	2
7	5	3	6	5	1	10	0	9	7	4	7	6	2	8	5	3	8	8	3	4
8	5	1	5	4	3	9	9	0	9	4	4	4	1	5	10	9	7	7	10	3
9	9	2	5	7	8	6	7	9	0	6	7	1	4	5	10	6	6	8	8	3
10	3	9	10	3	6	7	4	4	6	0	10	2	6	5	7	2	4	8	7	8
11	9	8	6	7	6	1	7	4	7	10	0	5	4	6	10	9	8	5	6	2
12	3	10	8	6	7	5	6	4	1	2	5	0	3	6	5	6	6	3	7	7
13	6	6	8	8	6	1	2	1	4	6	4	3	0	3	9	3	4	7	1	9
14	4	10	9	5	3	5	8	5	5	5	6	6	3	0	3	4	3	4	2	4
15	3	3	1	9	3	9	5	10	10	7	10	5	9	3	0	8	5	1	9	9
16	4	6	2	4	1	6	3	9	6	2	9	6	3	4	8	0	2	1	4	7
17	1	10	4	2	6	10	8	7	6	4	8	6	4	3	5	2	0	10	10	9
18	7	2	4	10	5	5	8	7	8	8	5	3	7	4	1	1	10	0	10	6
19	7	6	10	3	4	4	3	10	8	7	6	7	1	2	9	4	10	10	0	8
20	10	3	9	5	2	2	4	3	3	8	2	7	9	4	9	7	9	6	8	0

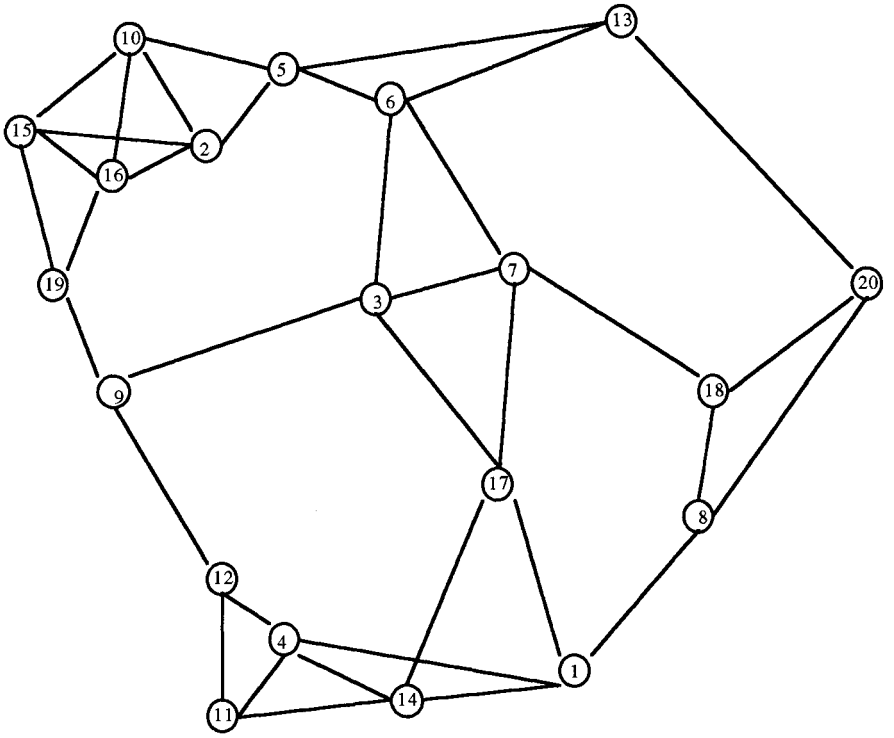


FIGURE 15 Initial topological configuration ($D = \$124{,}222.98$/month, $T = 81.84$ ms).

initial topology, the application of the third removal move links M_3^R during the first iteration cycle has generated, by deletion of links 1–14, 2–5, and 10–15, the topology specified by Fig. 16 and Table 9. The cost of this topology was checked against the cost of each topology generated from the initial topology during the current iteration cycle; this cost is found to be the lowest. Thus, it becomes the solution of the first perturbation cycle.

Therefore, iteration cycle 1 did not lead to a solution less expensive than the current initial topology. By contrast, we observe a diminution of the average delay T. In fact, this one has changed from 81.84 to 73.78 ms, that is, a diminution of 9.48%. For iteration cycle 2, this solution is kept as a new starting topology to which different moves are applied. The link removal move M_2^R led to a new topology with a total link cost $D = \$111{,}420.70$/month and an average delay $T = 95$ ms.

In iteration cycle 3, the application of the substitution move M_1^s to the starting topology shown in Fig. 15 led to the new topology specified by Fig. 17 and Table 10. The improvement percentage in terms of cost is 17.6%. By contrast, the average delay is changed from 81.84 to 133.68 ms, with an increase of 51.84 ms. This delay did not pass the maximum acceptable delay T_{max}, which is fixed in this application at 250 ms. Since we are looking for a solution that minimizes the total link cost, taking into consideration some performance constraints, this solution therefore becomes the best solution found during the overall execution of this method.

TABLE 8 Link Attributes of the Initial Topology

Link number	Link	Flow (Kbps)	Capacity (Kbps)	Utilization
3	1–4	134	230.40	0.58
7	1–8	170	230.40	0.73
13	1–14	74	100.00	0.74
16	1–17	70	100.00	0.70
22	2–5	216	230.40	0.93
27	2–10	18	19.20	0.93
32	2–15	6	9.60	0.62
33	2–16	210	230.40	0.91
40	3–6	302	460.80	0.65
41	3–7	174	230.40	0.75
43	3–9	388	460.80	0.84
51	3–17	348	460.80	0.75
61	4–11	14	19.20	0.72
62	4–12	302	460.80	0.65
64	4–14	96	100.00	0.96
71	5–6	296	460.80	0.64
75	5–10	192	230.40	0.83
78	5–13	120	230.40	0.52
86	6–7	134	230.40	0.58
92	6–13	78	100.00	0.78
110	7–18	204	230.40	0.88
121	8–17	138	230.40	0.59
122	8–18	144	230.40	0.62
124	8–20	94	100.00	0.94
127	9–12	500	921.60	0.54
134	9–19	550	921.60	0.59
140	10–15	86	100.00	0.86
141	10–16	70	100.00	0.70
146	11–12	158	230.40	0.68
148	11–14	68	100.00	0.68
169	13–20	108	230.40	0.46
172	14–17	94	100.00	0.94
176	15–16	16	19.20	0.83
179	15–19	130	230.40	0.28
183	16–19	322	460.80	0.69
189	18–20	74	100.00	0.74

Our results in terms of cost and mean delay are compared with results provided by other heuristics. First, TS is compared with CS. For both methods, the experience was based on the following choices: uniform traffic of 5 packets per second between each pair of nodes, 2-connected topologies (the CS method handles only 2-connected topologies), and the average length of packets being 1000 bits. We have used the capacity options and costs of Table 5. The Cartesian coordinates of the nodes are given in Table 11. Table 12 shows the comparative results provided by both methods. The last column of this table provides the improvement rate obtained by TS versus CS. In all cases, TS offers better solutions in terms of cost than CS. However, delays provided by CS are generally better.

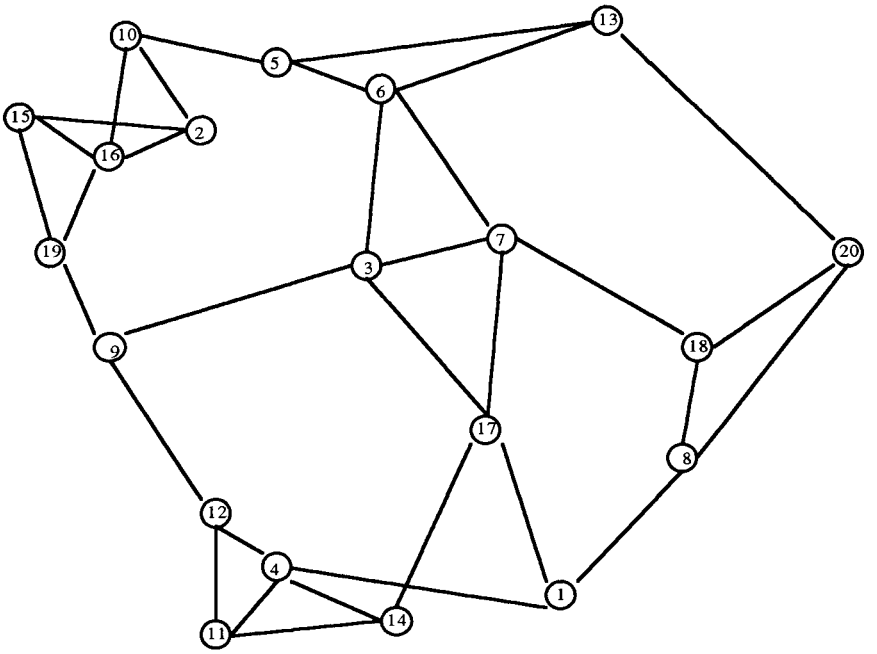


FIGURE 16 Best topology found during the first iteration cycle ($D = \$127{,}890.26$/month, $T = 73.78$ ms).

The TS approach has been also compared to the GA approach, on networks having a connectivity degree ranging from 3 to 5. The traffic between all pairs of nodes was uniform, equal to 5 packets per second, and the average length of packets was 1000 bits. We used the capacity options and costs given in Table 5. The Cartesian coordinates are similar to those shown in Table 11. Comparative results provided by both methods are given in Table 13. In almost cases

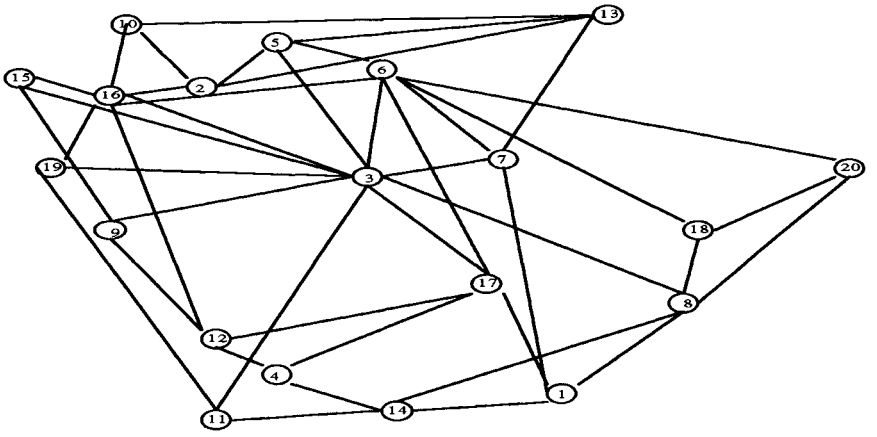


FIGURE 17 Topological configuration of the (last) final solution ($D = \$102{,}419.91$/month, $T = 133.68$ ms).

TABLE 9 Link Attributes of the Solution Obtained during the First Iteration Cycle

Link number	Link	Flow (Kbps)	Capacity (Kbps)	Utilization
3	1–4	182	230.40	0.78
7	1–8	144	230.40	0.62
16	1–17	70	100.00	0.70
27	2–10	106	230.40	0.46
32	2–15	6	9.60	0.62
33	2–16	98	100.00	0.98
40	3–6	294	460.80	0.63
41	3–7	194	230.40	0.84
43	3–9	388	460.80	0.84
51	3–17	340	460.80	0.73
61	4–11	54	100.00	0.54
62	4–12	310	460.80	0.67
64	4–14	96	100.00	0.96
71	5–6	288	460.80	0.45
75	5–10	400	460.80	0.86
78	5–13	120	230.40	0.52
86	6–7	134	230.40	0.58
92	6–13	78	100.00	0.78
110	7–18	204	230.40	0.88
121	8–17	164	230.40	0.71
122	8–18	144	230.40	0.62
124	8–20	94	100.00	0.94
127	9–12	508	921.60	0.55
134	9–19	558	921.60	0.60
141	10–16	276	460.80	0.59
146	11–12	158	230.40	0.68
148	11–14	28	50.00	0.56
169	13–20	108	230.40	0.46
172	14–17	128	230.40	0.55
176	15–16	102	230.40	0.44
179	15–19	130	230.40	0.56
183	16–19	330	460.80	0.71
189	18–20	74	100.00	0.74

(13 out of 14), costs provided by TS are better than those obtained by GA. However, delays obtained by TS are relatively less than those obtained by GA. In general, TS provides better solutions in terms of cost than GA.

We have also compared our results to those provided by SA. Our experience used the same data previously mentioned. Table 14 gives a summary of the results obtained by the two methods. These results confirm once again that TS offers better solutions than SA in terms of cost.

TABLE 10 Link Attributes of the Final Solution

Link number	Link	Flow (Kbps)	Capacity (Kbps)	Utilization
6	1–7	44	50.00	0.88
7	1–8	96	100.00	0.96
13	1–14	144	230.40	0.62
16	1–17	196	230.40	0.85
22	2–5	112	230.40	0.48
27	2–10	76	100.00	0.76
30	2–13	38	50.00	0.76
33	2–16	184	230.40	0.79
39	3–5	62	100.00	0.62
40	3–6	44	50.00	0.88
41	3–7	214	230.40	0.92
42	3–8	206	230.40	0.89
43	3–9	186	230.40	0.80
45	3–11	48	50.00	0.96
49	3–15	48	50.00	0.96
50	3–16	136	230.40	0.59
51	3–17	220	230.40	0.95
53	3–19	116	230.40	0.50
62	4–12	212	230.40	0.92
64	4–14	220	230.40	0.95
67	4–17	74	100.00	0.74
71	5–6	200	230.40	0.86
78	5–13	46	50.00	0.92
86	6–7	28	50.00	0.56
95	6–16	90	100.00	0.90
96	6–17	108	230.40	0.46
97	6–18	84	100.00	0.84
99	6–20	96	100.00	0.96
105	7–13	86	100.00	0.86
118	8–14	114	230.40	0.49
122	8–18	178	230.40	0.77
124	8–20	112	230.40	0.48
127	9–12	150	230.40	0.65
130	9–15	82	100.00	0.82
138	10–13	12	19.20	0.62
141	10–16	134	230.40	0.58
148	11–14	110	230.40	0.47
153	11–19	90	100.00	0.90
158	12–16	114	230.40	0.49
159	12–17	12	19.20	0.62
176	15–16	164	230.40	0.71
183	16–19	180	230.40	0.78
189	18–20	12	19.20	0.62

TABLE 11 Node Coordinates

Node	1	2	3	4	5	6	7	8	9	10
Abscissa	63	22	41	33	32	40	52	80	19	15
Ordinate	8	72	45	10	84	78	52	33	35	81
Node	11	12	13	14	15	16	17	18	19	20
Abscissa	27	27	70	48	4	10	56	82	9	95
Ordinate	3	16	96	4	73	71	27	47	54	61
Node	21	22	23							
Abscissa	67	56	54							
Ordinate	8	54	23							

TABLE 12 Comparison of Results Provided by TS and CS

			Cut Saturation			Tabu Search		
No.	N	T_{max}	D ($/month)	T (ms)	Cap. (Kbps)	D ($/month)	T (ms)	%D
1	6	80	19972	32.73	50	8480	80	58
2	10	100	44262	37.28	100	25272	69.56	43
3	12	120	56231	109.29	100	43338	77.48	23
4	15	150	160984	17.69	230.40	65161	141.86	60
5	20	150	381278	7.16	460.80	144131	103.73	63
6	23	80	394455	10.80	460.80	210070	58.12	45

TABLE 13 Comparison of Results Provided by TS and GA

				GA		TS			
No.	N	K	T_{max}	D ($/month)	T (ms)	D ($/month)	T (ms)	%T	%D
1	6	3	100	10697	80.62	9867	80.89	—	8
2	6	4	100	12938	91.35	11338	78.50	15	13
3	10	3	150	27534	115.08	24322	97.12	16	12
4	10	4	150	26860	94.10	22252	94.82	—	18
5	10	5	150	30189	91.57	28407	87.20	5	6
6	15	3	200	65142	95.06	60907	78.95	17	7
7	15	4	200	63838	83.10	60013	93.10	—	6
8	15	5	200	46774	94.63	41626	90.62	5	12
9	20	3	250	113714	96.25	105495	112	—	8
10	20	4	250	110815	103.06	103524	88.02	15	7
11	20	5	250	116123	76.98	106122	83.14	—	9
12	23	3	190	162627	107.18	148340	90.67	16	9
13	23	4	190	163129	90.14	168118	113	—	−3
14	23	5	190	149522	87.13	139154	86.14	2	7

TABLE 14 Comparison of Results Provided by TS and SA

				SA		TS			
No.	N	K	T_{max}	D ($/month)	T (ms)	D ($/month)	T (ms)	%T	%D
1	6	3	100	11490	90.50	9867	80.89	11	15
2	6	4	100	16939	88.15	11338	78.50	11	33
3	10	3	150	30433	103.06	24322	97.12	6	20
4	10	4	150	28450	100.02	22252	94.82	5	22
5	10	5	150	33982	98.23	28407	87.20	12	17
6	15	3	200	65200	110	60907	78.95	29	7
7	15	4	200	70434	96.12	60013	93.10	4	15
8	15	5	200	54810	105	41626	90.62	14	24
9	20	3	250	132872	123.06	105495	112	10	21
10	20	4	250	151918	99.09	103524	88.02	12	32
11	20	5	250	143341	114.03	106122	83.14	28	26
12	23	3	190	174696	97.45	148340	90.67	7	15
13	23	4	190	186871	115.08	168118	113	2	11
14	23	5	190	164123	93.67	139154	86.14	9	16

REFERENCES

1. Ag Rhissa, A., Jiang, S., Ray Barman, J., and Siboni, D. Network generic modeling for fault management expert system. In *International Symposium on Information, Computer and Network Control,* Beijing, China, Feb. 1994.
2. AUTONET / *Performance-3,* Network Design and Analysis Corporation, VA, USA, 1995.
3. Boorstyn, R. R., and Frank, H. Large-scale network topological optimization. *IEEE Trans. Commun.* 25(1): 29–47, 1977.
4. Buchanan, B. G. Some approaches to knowledge acquisition. In *Machine Learning: A Guide to Current Research* (Mitchell, T. M., Carbonell, J. G., and Michalski, R. S. Eds.), pp. 19–24. Kluwer Academic, Dordrecht, 1986.
5. Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. An overview of machine learning. In *Machine Learning: An Artificial Intelligence Approach* (Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. Eds.), pp. 3–23. Tioga, Portola Valley, CA, 1983.
6. Coan, B. A., Leland, W. E., Vechi, M. P., and Weinrib, A. Using distributed topology update an preplanned configurations to achieve trunk network survivability. *IEEE Trans. Reliability* 40: 404–416, 1991.
7. COMNET III. *A Quick Look at COMNET III, Planning for Network Managers.* CACI Products Company, La Jolla, CA, 1995.
8. Dietterich, T., and Michalski, R. A comparative review of selected methods for learning from examples. In *Machine Learning: An Artificial Intelligence Approach* (Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. Eds.), pp. 41–82.Tioga, Portola Valley, CA, 1983.
9. Dutta, A., and Mitra, S. Integrating heuristic knowledge and optimization models for communication network design. *IEEE Trans. Knowledge and Data Engrg.* 5: 999–1017, 1993.
10. Gavish, B. Topological design of computer networks — The overall design problem. *Eur. J. Oper. Res.* 58: 149–172, 1992.
11. Gavish, B., and Neuman, I. A system for routing and capacity assignment in computer communication networks. *IEEE Trans. Commun.* 37(4): 360–366, 1989.
12. Gerla, M., and Kleinrock, L. On the topological design of distributed computer networks. *IEEE Trans. Commun.* 25: 48–60, 1977.
13. Gerla, M., Frank, H., Chou, H. M., and Eckl, J. A cut saturation algorithm for topological design of packet switched communication networks. In *Proc. of National Telecommunication Conference,* Dec. 1974, pp. 1074–1085.
14. Glover, F. Tabu search: Improved solution alternatives for real world problems. In *Mathematical Programming: State of the Art* (J. R. Birge and K. G. Murty, Eds.), pp. 64–92 Univ. of Michigan Press, Ann Arbor, MI, 1994.
15. Glover, F. Tabu thresholding: improved search by nonmonotonic trajectories. *INFORMS J. Comput.* 7: 426–442, 1995.
16. Glover, F., and Laguna, M. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems* (C. Reeves, Ed.), pp. 70–141. Blackwell Scientific, Oxford, 1993.
17. Holland, J. H. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, Ann Arbor, MI, 1975.
18. Jan, R. H., Hwang, F. J., and Cheng, S. T. Topological optimization of a communication network subject to reliability constraints. *IEEE Trans. Reliability* 42: 63–70, 1993.
19. Kamimura, K., and Nishino, H. An efficient method for determining economical configurations of elementary packet-switched networks. *IEEE Trans. Commun.* 39(2): 278–288, 1991.
20. Karp, R. M. Combinatorics, complexity, and randomness. *Commun. ACM* 29(2): 97–109, 1986.
21. Kershenbaum, A., Kermani, P., and Grover, G. A. MENTOR: An algorithm for mesh network topological optimization and routing. *IEEE Trans. Commun.* 39: 503–513, 1991.
22. Kershenbaum, A. *Telecommunication Network Design Algorithms.* IBM Thomas J. Watson Research Center, USA, 1993.
23. Kirkpatrick, S. Optimization by simulated annealing: Quantitative studies. *J. Stat. Phys.* 34: 975–986, 1984.
24. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science* 220: 671–680, 1983.

25. Kleinrock, L. *Queueing Systems: Vol. II, Computer Applications.* Wiley–Interscience, New York, 1976.
26. MIL3. *The OPNET Modeler Simulation Environment,* 1997. Available at http://www.mil3.com/products/modeler/home.html.
27. Michalski, R. S. A theory and methodology of inductive learning. In *Machine Learning: An Artificial Intelligence Approach* (Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. Eds.), pp. 83–134. Tioga, Portola Valley, CA, 1983.
28. Monma, C. L., and Sheng, D. D. Backbone network design and performance analysis: A methodology for packet switching networks. *IEEE J. Selected Areas in Commun.* 4(6): 946–965, 1986.
29. Newport, K. T., and Varshney, P. K. Design of survivable communications networks under performance constraints. *IEEE Trans. Reliability* 40(4): 433–440, 1991.
30. Pierre, S., and Gharbi, I. A generic object-oriented model for representing computer network topologies, *Adv. Eng. Software* 32(2): 95–110, 2001.
31. Pierre, S. Inferring new design rules by machine learning: A case study of topological optimization. *IEEE Trans. Man Systems Cybernet.* 28A (5): 575–585, 1998.
32. Pierre, S., and Legault, G. A genetic algorithm for designing distributed computer network topologies. *IEEE Trans. Man Systems Cybernet.* 28(2): 249–258, 1998.
33. Pierre, S., and Legault, G. An evolutionary approach for configuring economical packet-switched computer networks. *Artif. Intell. Engrg.* 10: 127–134, 1996.
34. Pierre, S., and Elgibaoui, A. A tabu search approach for designing computer network topologies with unreliable components. *IEEE Trans. Reliability* 46(3): 350–359, 1997.
35. Pierre, S., Hyppolite, M.-A., Bourjolly, J.-M., and Dioume, O. Topological design of computer communications networks using simulated annealing. *Engrg. Appl. Artif. Intell.* 8: 61–69, 1995.
36. Pierre, S. A new methodology for generating rules in topological design of computer networks. *Engrg. Appl. Artif. Intell.* 8(3): 333–344, 1995.
37. Pierre, S. Application of artificial intelligence techniques to computer network topologies. *Eng. Appl. Artif. Intell.* 6: 465–472, 1993.
38. Rose, C. Low mean internodal distance network topologies and simulated annealing. *IEEE Trans. Commun.* 40: 1319–1326, 1992.
39. Saksena, V. R. Topological analysis of packet networks. *IEEE J. Selected Areas Commun.* 7: 1243–1252, 1989.
40. Samoylenko, S. I. Application of heuristic problem-solving methods in computer communication networks. *Mach. Intell.* 3(4): 197–210, 1985.
41. Schumacher, U. An algorithm for construction of a K-connected graph with minimum number of edges and quasiminimal diameter. *Networks* 14: 63–74, 1984.
42. Siboni, D., Ag Rhissa, A., and Jiang, S. Une fonction intelligente de gestion globale des incidents pour un hyperviseur de réseaux hétérogènes. In *Actes des quatorzièmes journées internationales d'Avignon,* AI' 94, Paris, 30 Mai–3 Juin 1994, pp. 379–387.
43. Suk-Gwon, C. Fair integration of routing and flow control in communications networks. *IEEE Trans. Commun.* 40(4): 821–834, 1992.
44. Van Laarhoven, P. J. M., and Aarts, E. H. L. *Simulated Annealing: Theory and Applications.* Reidel, D., Dordrecht, Holland, 1987.
45. Wong, R. T. Probabilistic analysis of a network design problem heuristic. *Networks* 15: 347–362, 1985.
46. Yokohira, T., Sugano, M., Nishida, T., and Miyahara, H. Fault-tolerant packet-switched network design and its sensitivity. *IEEE Trans. Reliability* 40(4): 452–460, 1991.

DATABASE AND DATA COMMUNICATION NETWORK SYSTEMS

Techniques and Applications

VOLUME 2

This Page Intentionally Left Blank

DATABASE AND DATA COMMUNICATION NETWORK SYSTEMS

Techniques and Applications

VOLUME 2

Edited by

Cornelius T. Leondes

Professor Emeritus
University of California
Los Angeles, California

ACADEMIC PRESS
An imprint of Elsevier Science

Amsterdam Boston London New York Oxford Paris
San Diego San Francisco Singapore Sydney Tokyo

Front cover: Digital Image © 2002 PhotoDisc.

This book is printed on acid-free paper. ♾

Academic Press
An imprint of Elsevier Science
525 B Street, Suite 1900, San Diego, California 92101-4495, USA
http://www.academicpress.com

Academic Press
84 Theobalds Road, London WC1X 8RR, UK
http://www.academicpress.com

Library of Congress Catalog Card Number: 20016576

International Standard Book Number: 0-12-443895-4 (set)
International Standard Book Number: 0-12-443896-2 (volume 1)
International Standard Book Number: 0-12-443897-0 (volume 2)
International Standard Book Number: 0-12-443898-9 (volume 3)

PRINTED IN THE UNITED STATES OF AMERICA
02 03 04 05 06 07 MM 9 8 7 6 5 4 3 2 1

CONTENTS

CONTENTS OF VOLUME 2

CONTENTS OF VOLUME I

CONTENTS OF VOLUME 3

CONTRIBUTORS

Numbers in parentheses indicate the pages on which the authors' contributions begin.

C. Apostolas (859) Network Communications Laboratory, Department of Informatics, University of Athens, Panepistimioupolis, Athens 15784, Greece

Bernardino Arcay (825) Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Gill Barequet (277) The Technion—Israel Institute of Technology, Haifa 32000, Israel

József Békési (233) Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

Elisa Bertino (175) Dipartimento di Scienze dell'Informazione, Università di Milano, 20135 Milano, Italy

Hsin-Horng Chen (203) Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

Shyi-Ming Chen (203) Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan, Republic of China

Jian Chen (663) Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Carlos Dafonte (825) Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Timon C. Du (1) Department of Industrial Engineering, Chung Yuan Christian University, Chung Li, Taiwan 32023; and Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Shatin, NT Hong Kong

Elena Ferrari (175) Dipartimento di Chimica, Fisica e Matematica, Università dell'Insubria – Como, Italy

Gábor Galambos (233) Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

Jacob Gan (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

Mario Gerla (693) Computer Science Department, University of California—Los Angeles, Los Angeles, California 90095

Stathes Hadjiefthymiades (635) Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

M. Ahsan Akhtar Hasin[1] (417) Industrial Systems Engineering, Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

Li Hong-Yi (901) Advanced Wireless Networks, Nortel Research, Nepean, Ontario, Canada K2G 6J8

Hiroshi Ishikawa (77) Department of Electronics and Information Engineering, Tokyo Metropolitan University, Tokyo 192-0397, Japan

Hiroo Iwata (549) Institute of Engineering Mechanics and Systems, University of Tsukuba, Tsukuba 305-8573, Japan

Rong-Hong Jan (719) Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

Chua Chee Kai (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

Myoung Ho Kim (41) Department of Computer Science, Korea Advanced Institute of Science and Technology, Taejon 305-701, Korea

Wataru Kishimoto (783) Department of Information and Image Sciences, Chiba University, Chiba 263-8522, Japan

Shmuel Tomi Klein (573) Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel

Marco Lazzari[2] (751) ISMES, Via Pastrengo 9, 24068 Seriate BG, Italy

Doheon Lee (41) Department of BioSystems, Korea Advanced Institute of Science and Technology, Daejon, Republic of Korea

[1] Current address: Industrial and Production Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh

[2] Current address: Dipartimento de Scienze della Formazione, Università di Bergamo, Bergamo 24029, Italy

Chiang Lee (123) Institute of Information Engineering, National Cheng-Kung University, Tainan, Taiwan, Republic of China

Mong Li Lee (497) School of Computing, National University of Singapore, Singapore 117543

Kunwoo Lee (517) School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea

Ngoh Lek-Heng (901) SingAREN, Kent Ridge Digital Labs, Singapore 119613

Jason Lowder (663) School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

Drakoulis Martakos (635) Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

Saverio Mascolo (693) Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 70125 Bari, Italy

Tong Mei (367) Gintic Institute of Manufacturing Technology, Singapore 638075

Troy Milner (663) School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

P. C. Pandey (417) Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

Sangkun Park (517) Institute of Advanced Machinery and Design, Seoul National University, Seoul 151-742, Korea

Samuel Pierre (289) Mobile Computing and Networking Research Laboratory (LARIM); and Department of Computer Engineering, École Polytechnique de Montréal, Montréal, Quebec, Canada H3C 3A7

Sameer Pradhan (663) Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Paolo Salvaneschi (751) ISMES, Via Pastrengo 9, 24068 Seriate BG, Italy

G. Sfikas[3] (859) Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

Timothy K. Shih (327) Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 25137, Republic of China

José A. Taboada (825) Department of Electronics and Computer Science, Universidade de Santiago de Compostela, 15782, Santiago de Compostela (A Coruña), Spain

R Tafazolli (859) Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

I-Fei Tsai (719) Wistron Corporation, Taipei 221, Taiwan

[3]Current address: Lucent Technologies, Optimus, Windmill Hill Business Park, Swindon, Wiltshire SN5 6PP, England

Irene M. Y. Woon (497) School of Computing, National University of Singapore, Singapore 117543

Xindong Wu (663) Department of Computer Science, University of Vermont, Burlington, Vermont 05405

Du Zhaohui (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

10

MULTIMEDIA DATABASE SYSTEMS IN EDUCATION, TRAINING, AND PRODUCT DEMONSTRATION

TIMOTHY K. SHIH

Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 25137, Republic of China

Multimedia computing and networking changes the style of interaction between computer and human. With the growth of the Internet, multimedia applications such as educational software, electronic commerce applications, and video games have brought a great impact to the way humans think of, use, and rely on computers/networks. One of the most important technologies to support these applications is the distributed multimedia database management system (MDBMS). This chapter summarizes research issues and state-of-the-art technologies of MDBMSs from the perspective of multimedia presentations. Multimedia presentations are used widely in different forms from instruction delivery to advertisement and electronic commerce, and in different software architectures from a stand alone computer, to local area networked computers and World Wide Web servers. These different varieties of architectures result in different organization of MDBMSs. The chapter discusses MDBMS architectures and examples that were developed at our university.

Database and Data Communication Network Systems, Vol. 2

I. INTRODUCTION

Multimedia computing and networking changes the way people interact with computers. In line with the new multimedia hardware technologies, as well as well-engineered multimedia software, multimedia computers with the assistance of the Internet have changed our society to a distanceless and colorful global community. Yet, despite the fantasy gradually being realized, there still exist many technique problems to be solved. This chapter summarizes state-of-the-art research topics in multimedia computing and networking, with emphasis on database technologies for multimedia presentations. The chapter addresses many problems from the perspective of multimedia applications. Theoretical details are dropped from the discussion not because of the lack of their importance but due to the avoiding of tediousness. A list of carefully selected references serves as suggested readings for those who are participating in this research area.

In this section, the discussion starts with the preliminary concepts of multimedia presentations widely used in education, training, and demonstrations. The supporting multimedia database management systems (MDBMSs) and related techniques in terms of reusability, distribution, and real-time considerations are then presented. In Section II and Section III, two MDBMSs are discussed. The first is an intelligent multimedia presentation design system for training and product demonstration. The second is a World Wide Web documentation database used in a distance learning environment. These systems are also illustrated in the Appendices. Finally, some suggestions for future extensions of MDBMSs are presented in Section IV.

A. Multimedia Presentation

One of the key reasons for making multimedia computers become attractive and successful is the availability of many varieties of multimedia presentation software, including many CD-ROM titles carrying educational software, entertainment, product demonstrations, training programs, and tutoring packages. Most presentations are hypertext like multimedia documentation [1,8,26]. These documents, while retrieved, involve a navigation topology, which consists of hyperlinks jumping from hot spots to other presentation windows. The underlying implementation mechanism of this type of documentation traversal may rely on a message passing system and an event-based synchronization scheme. Discussions of these mechanisms are found in [6,16,18,19,37,39,40].

Other new areas related to multimedia are intelligent tutoring [32] and intelligent interface [4,5,11,12,36]. The incorporation of Expert System technology has caused multimedia presentations to diversify. An intelligent multimedia presentation can learn from its audiences through interactions. The feedback is asserted into the knowledge base of the presentation. Therefore, after different audiences of different backgrounds interact with a tutorial or a training program, the multimedia workstation may act according to the individuals and give different appropriate guidance.

B. Multimedia Database Management System

In order to support the production of multimedia presentations, the management of multimedia resources (e.g., video clips, pictures, sound files) is important. Also, presentations can be designed as building blocks to be reused. To facilitate presentation design, many articles indicate the need for a multimedia database [27,29,30,33,40,41]. A multimedia database is different from a traditional relational database in that the former is object-oriented while the latter uses an entity relation diagram. Moreover, a multimedia database needs to support binary resource types of large and variable sizes. Due to the amount of binary information that needs to be processed, the performance requirement of a multimedia database is high. Clustering and indexing mechanisms supporting multimedia databases are found in [28,43]. In general, there are four ways to maintain multimedia resources:

- use a primitive file system,
- use a relational database to store multimedia resources,
- use an object-oriented database to organize multimedia documents, and
- build a multimedia database from scratch.

The first approach is too limited in that it is relatively hard to share resources among multimedia documents. Also it is hard for a presentation designer to keep track of various versions of a resource. Using a relation database will partially solve this problem. However, since multimedia documentation is object-oriented, using a relational database will introduce some overhead. Therefore, many researchers suggest using an existing object-oriented database system for building multimedia databases [9,12,13,29,41]. However, there are still some problems yet to be solved:

- *Quality of service.* Multimedia resources require a guarantee of the presentation quality. It is possible to use a specific file structure and program to control and guarantee the quality of service (QoS). A traditional OODBMS does not support QoS multimedia objects.
- *Synchronization.* Synchronization of multiple resource streams plays an important role in a multimedia presentation, especially when the presentation is running across a network. In our projects, we are not currently dealing with this interstream synchronization problem. We use an event-based synchronization mechanism in the current systems. Interstream and other synchronization issues are left for future development. However, in the future we plan to break multimedia resources into small pieces in order to implement interstream synchronization, which is especially important in a distributed environment.
- *Networking.* A distributed database requires a locking mechanism for concurrent access controls. Using our own architecture design, it is easy for us to control some implementation issues, such as a two-phase locking mechanism of the database, database administration, and control of traffic on the network.

In order to ensure the synchronization of a multimedia document, a multimedia system needs to have full control of data access activity and its timing. It is sometimes difficult to have a total control of the performance of an

object-oriented database, especially when the multimedia database is distributed. Therefore, if needed, a multimedia database must be built using only primitive multimedia programming facilities.

Building a multimedia database requires three layers of architecture to be considered.

- the interface layer,
- the object composition layer, and
- the storage management layer.

The tasks that we must deal with in the interface level include object browsing, query processing, and the interaction of object composition/decomposition. Object browsing [8] allows the user to find multimedia resource entities to be reused. Through queries, either text-based or visualized, the user specifies a number of conditions to the properties of resource and retrieves a list of candidate objects. Suitable objects are then reused. Multimedia resources, unlike text or numerical information, cannot be effectively located using a text-based query language. Even natural language presented in a text form is hard to precisely retrieve a picture or a video with certain content. Content-based information retrieval research [21,49] focuses on the mechanism that allows the user to effectively find reusable multimedia objects, including pictures, sound, video, and the combined forms. After a successful retrieval, the database interface should help the user to compose/decompose multimedia documents. The second layer works in conjunction with the interface layer to manage objects. Typically, object composition requires a number of links, such as association links, similarity links, and inheritance links to specify different relations among objects. These links are specified either via the database graphical user interface, or via a number of application program interface (API) functions. The last layer, the storage management layer, needs to consider two performance issues: clustering and indexing. Clustering means to organize multimedia information physically on a hard disk (or an optical storage) such that, when retrieved, the system is able to access the large binary data efficiently. Usually, the performance of retrieval needs to guarantee some sort of QoS [44]. Indexing means that a fast-locating mechanism is essential for finding the physical address of a multimedia object [28]. Sometimes, the scheme involves a complex data or file structure. Media synchronization should be considered in both issues.

There are other issues in multimedia database research, including transaction processing, object locking mechanisms and concurrent access, persistency, versioning, security, and referential integrity. Most of these are also issues of traditional database research. The research of multimedia databases has become an important issue in the community of multimedia computing.

C. Multimedia Synchronization

Multimedia information is processed in real-time. Therefore, the temporal aspect of multimedia information coherence–synchronization is one of the most interesting research topics in multimedia computing [3,7,14,15,17,19,22, 23,25,31,42,45,47,48]. Synchronization mechanisms are, in general, divided

into the intra- and the interstream-based schemes. The former focuses on the simultaneous demonstration of one or more sources of information in one multimedia resource stream (e.g., a video file contains both motion pictures and sound data). The later discusses the simultaneous process of multiple streams holding different resources (e.g., an animation video is synchronized with a MIDI music file and a sound record).

Among a number of synchronization computation models, timed Petri net [20,31] seems to be a powerful model for describing the behavior of real-time systems. A Petri net is a bipartite directed graph with two types of nodes: places and transitions. A place node holds tokens, which are passed to transitions. A place can also represent a multimedia resource, which is demonstrated for a period of time before its token is passed to a transition. A transition controls the synchronization of those places adjacent to the transition. Timed Petri net is found to be one of the most suitable models for multimedia synchronization controls. Another theoretical computation of synchronization is based on temporal interval relations [2]. There are 13 types of relations between a pair of temporal intervals. Temporal interval relations can be used to calculate the schedule of a multimedia presentation [38]. Synchronization information can also be embedded into multimedia resources. Synchronization can also be controlled by event and message passing. The research of multimedia synchronization becomes a challenge if the mechanism is to be implemented on a distributed environment.

D. Multimedia Networking

In line with the success of the Internet, the needs of multimedia communication has brought the attention of many software developers and researchers [10,24,34,35,46]. Two important issues are network transmission protocols and network infrastructure. The former includes mechanisms enabling transmissions of a guaranteed quality and security. The later focuses on the structure of communication system so that reliability and traffic management is feasible. At the top of multimedia network systems (either intra- or inter-networking), many applications are available. Multimedia client-server based applications include the World Wide Web (WWW), multimedia documentation on demand (e.g., video-on-demand and news-on-demand), electronic catalog ordering, computer-supported cooperative work (CSCW), video conferencing and distance learning, and multimedia electronic mail. The combination of multimedia and networking technologies changes the way people think of, use, and rely on computers.

E. Reusability

Multimedia presentations are software, which need a specification to describe their functions. Nevertheless, the well-defined specification does not realize an improved functionality of a system. From the perspective of software development, reusability is one of the most important factors in improving the efficacy of multimedia presentation designs and multimedia database access. Many

articles discuss reusability—one of them [50] analyzes nine commonly believed software reuse myths, and points out why reusability has not played a major role in improving software productivity. One article [51] points out that there are three common approaches to achieve software reuse: subroutine libraries, software generators, and object-oriented languages. However, each individual approach raises problems. Therefore, the author takes the best features from each of these and develops an object-oriented language, called Meld. Another reuse mechanism via building blocks is proposed in [52]. The authors apply the concept to systems programming and achieve a great success. The authors also indicate that there are three important aspects of reusability: the abstraction level, customization methods, and reusability conditions. In [53], reusability criteria are proposed for the Ada language. The author suggests that, to be reusable, a program must be transportable, be context independent, and be independent of the runtime environment. According to the discussion given in [54], there are two levels of reuse: the reuse of knowledge and the reuse of software components. In order to support the fast retrieval of reusable objects, a classification scheme for grouping together objects sharing a common characteristic is required. A classification scheme organizing a software reuse library is proposed in [55]. Instead of using a centralized repository, the author proposes a generic design that can be applied to different environments. Another article [56] proposes two concepts, domain and theme, to allow software component classification by the services that they offer and by application domain.

The value of object reuse needs to be evaluated. A quantitative study of reuse through inheritance is found in [57]. The author studies several software systems containing a large number of classes. The results show reuse through inheritance is far less frequent than expected. Another article [58] measures cohesion versus reuse through inheritance, and determines that reuse through inheritance results in a lower cohesion. Fortunately, many articles propose solutions to software reuse. The RSL (Reusable Software Library) prototype [59] is a software system incorporating interactive design tools and a passive database of reusable software components. Each component is associated with some attributes to be used in their automatic retrieval. Another author [60] claims that reusable components are not enough. Software engineers need an intelligent support system to help them learn and understand how to reuse components in a software environment. A set of tool kits is implemented to support object reuse. NUT [61,62] is an object-oriented programming language/environment that supports class and object reuse. In NUT, classes are prototypes; when they are reused, initial values (could be nil values) are assigned according to the class specification. Objects can also be implicitly generated. For instance, program objects can be generated when equations are processed. Knowledge in the NUT environment can be reused in the following ways:

- As a super-class,
- When a new object is created via the "new" operator,
- As a prototype for specifying components of another class,
- As the specification of a predicate, and
- As a specification of problem conditions in problem statements.

Only the first two are typical at conventional object-oriented systems.

The above discussions consider reusability from the perspective of software development. A multimedia presentation contains a collection of presentation components as does a software system. To achieve object reuse in a multimedia database, we consider four tasks:

- ***Declare building blocks.*** Presentation sections are the building blocks of a multimedia presentation. These sections are objects to be reused.
- ***Retrieve building blocks.*** A presentation system should provide a tool for assisting the user in retrieving appropriate building blocks.
- ***Modify building blocks.*** The building blocks to be reused may be modified to fit the need of a new presentation.
- ***Reorganize database storage.*** Database storage should be reorganized so that information sharing is feasible and disk space is used efficiently.

Building block declaration is achieved by means of an object grouping mechanism at different levels of our database. These reusable objects can be retrieved using a browser. After finding the reusable presentation components, the presentation designer establishes links and possibly modifies the component. Finally, we have a storage management mechanism to allow information sharing on the disk.

F. Other Considerations

There are other challenges in developing a multimedia DBMS. For instance, the design of a visual query language for content-based retrieval of multimedia information is a difficult problem yet to be solved. Since multimedia information is not stored as text or numbers, traditional database query commands are no longer suitable. Quick retrieval of information based on the semantics of a multimedia resource is an important research topic. In our system, we did not address content-based retrieval. However, we developed a visual database browser that facilitates selection of resources. Another important issue relates to the quality of service [44]. Within the limits of hardware and operating system performance, we must organize multimedia information on the disk and across the network in such a way that we can guarantee to exceed a minimum level of presentation quality. In our system, we have a sophisticated storage management and indexing mechanism for the efficient retrieval of large multimedia resources. These concepts are further discussed with the two MDBMSs in the following sections.

II. DATABASE APPLICATIONS IN TRAINING—THE IMMPS PROJECT

As multimedia personal computers become widely available with reasonable prices, multimedia PCs built with sound card, CD-ROM, and high-resolution graphics display unit are used broadly by business persons, engineers, and others. Due to the impressive sound effects, graphics animation, and video play, the importance and business opportunity of multimedia presentation systems

are realized by researchers and commercial software developers. Presentation design software packages are also available at affordable prices. However, most presentations generated by these systems communicate with the addressee in a single direction manner. That is, the presentation software does not listen to the listeners' response. For this reason, we have developed an Intelligent Multimedia Presentation System (IMMPS) [36,39] to overcome the shortage.

IMMPS is one of the database applications that we have created. In the development of this project, our research focuses on the following issues:

- Canonical representation of knowledge,
- Intelligent presentation of specification language,
- Specification and learning of addressee characteristics, and
- Multimedia DBMS with reuse controls.

Presentation intelligence is represented by a canonical rule-based format. The knowledge not only includes the addressee's background (i.e., common sense of the person who watches the presentation) but allows human reactions to be learned by the presentation program. A database management system is also designed for the CD ROM title designers for organizing and storing multimedia resources and presentations. This database is associated with a presentation reuse control model. An intelligent specification language is designed. The language provides facilities for hypermedia access and rule-based statements for knowledge representation. Our system supports personalization. Not only can the graphical user interface of the generated presentation be fully customized, but the underlying knowledge of the addressee can be easily updated. The system also provides a learning subsystem to be included in the generated software title, which allows an addressee's interaction to be asserted into the knowledge base. This learning environment, the presentation inference engine, and other components can be included in the runtime environment of a CD-ROM title. Figure 1 illustrates the software architecture of the IMMPS system.

We suggest that a presentation can be designed from two different perspectives: the navigation view and the representation view. Figure 2 shows a presentation navigation subsystem and a knowledge inference subsystem facilitating these two views. A window I/O control subsystem collects the audience's navigation messages and performs multimedia operations. The navigation subsystem communicates with the inference subsystem via messages that assert or retract knowledge so that the audience's behaviors will be learned by the computer. The knowledge inference result can also produce a side effect that sends a message to the navigation subsystem for some actions. Similarly, via message passing, the end user's navigation information is passed from the window I/O control system to the navigation system. Also the content of a presentation, such as a video play, is presented by the media control interface (MCI) functions provided by Microsoft.

From the navigation view, a presentation is a graph with nodes as presentation windows and edges as navigation links. From the representation view, information that can be shared among windows includes background of the audiences (e.g., the addressee's name or knowledge of the presentation topic), multimedia resources (e.g., a text document file or a video file showing a mechanical operation), or other knowledge useful in the presentation. A property

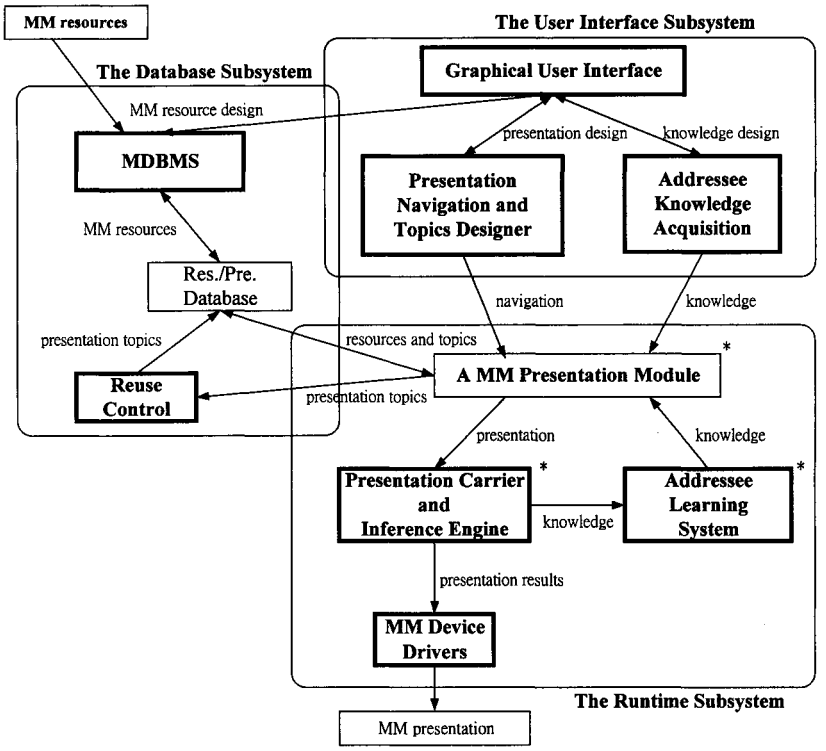


FIGURE 1 The software architecture of IMMPS.

inheritance structure such as a tree or a DAG (directed acyclic graph) is suitable for our knowledge representation architecture. Figure 3 illustrates these two structures. In the figure, a presentation window (i.e., PWin) is a composed object that represents a topic that a presenter wants to discuss. A presentation window may contain push buttons, one or more multimedia resources to be

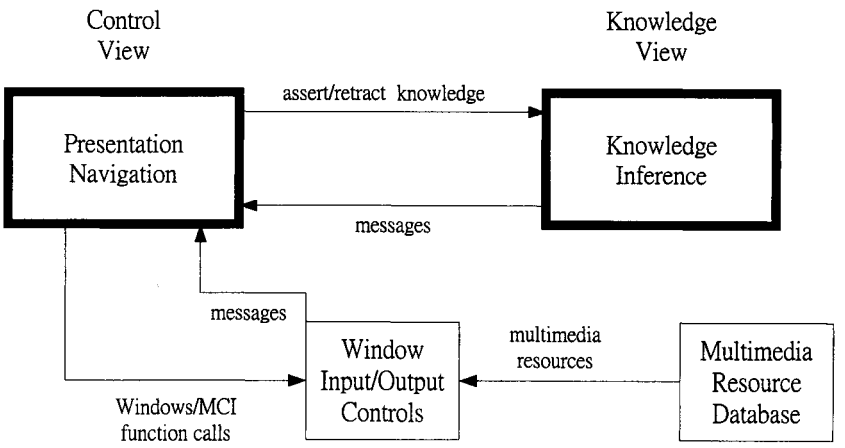


FIGURE 2 A presentation from different points of view.

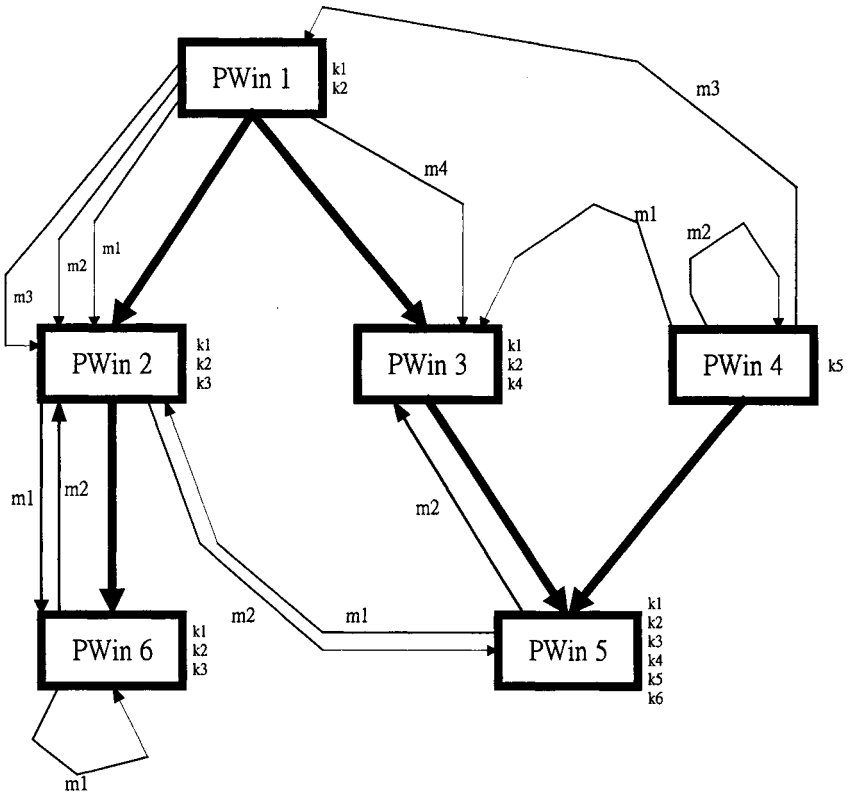


FIGURE 3 Graph and DAG representations of a multimedia presentation.

presented, and a number of knowledge rules (e.g., k1, k2, k3). A message (e.g., m1, m2) with optional parameters can be passed between two presentation windows (or passed back to the same presentation window). The graph edges representing navigation links are shown in thin lines, with message names as labels. The DAG edges representing knowledge inheritance are shown in thick lines without labels. In the figure, to the right of each presentation window, we show the knowledge rules that can be used in the presentation window. Even though knowledge rules "k1" and "k2" are shared among PWin 1, PWin 2, PWin 3, PWin 5, and PWin 6, they are stored only once in PWin 1. Note that multiple inheritance is also allowed, as PWin 5 inherits knowledge rules from both PWin 3 and PWin 4.

There are a number of restrictions applied to our message passing system and knowledge inheritance system. For instance, a message passed between two presentation windows has a unique name. Only the destination presentation window can receive the specific message sent to it. Each message has only one source and one destination. A child presentation window inherits all knowledge rules and facts from its parent presentation windows. The relation of knowledge inheritance is transitive. However, the inheritance architecture is acyclic. That is, a presentation window can not be a parent presentation window of its own. If a child presentation window contains a knowledge rule that has the same rule name as one of the rules the child presentation window inherits (directly or indirectly), the rule defined in the child presentation window

overrides the one from its parent presentation windows. If two rules belonging to two presentation windows have the same name, the rule should be stored in a common parent presentation window only once to avoid inconsistency.

IMMPS can be used in general purpose presentations or demonstrations in different fields such as education, training, product demonstration, and others. This system serves as a sample application showing our multimedia database research results [40,41]. The multimedia database of IMMPS has two major layers: the frame object layer and the resource object layer. Figure 4

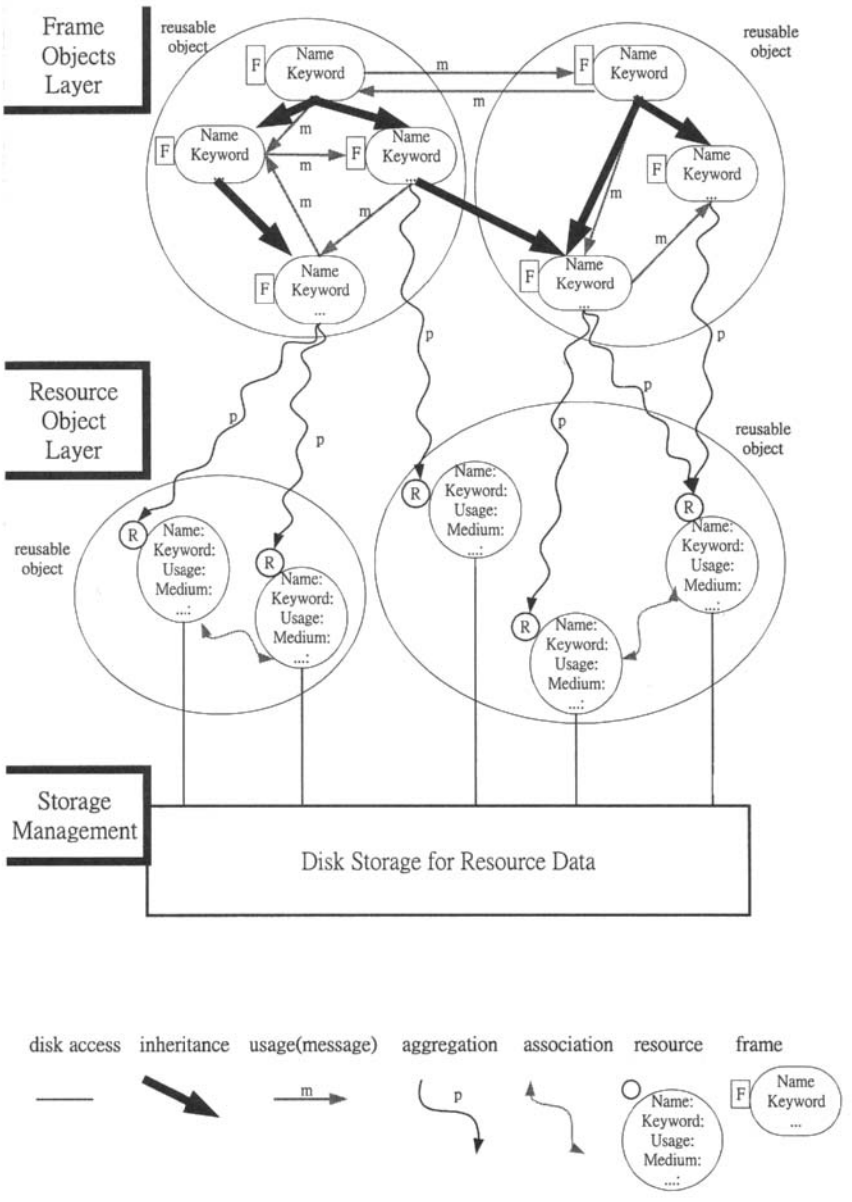


FIGURE 4 The object-oriented database hierarchy of IMMPS.

gives an overview of the database architecture. The two types of objects in our database are frames and resources. A frame is denoted by a box while a resource is represented by a circle with its associated properties given in an attached, rounded box. The actual data of a resource are stored in a commercial multimedia file format on the hard disk. This approach allows our system to be fully portable. Since the underlying multimedia file format is independent of the operating system (such as MS Windows 98), the database server program can be recompiled and used in a different environment (such as UNIX).

In the database architecture, the frame object layer contains presentation frames. A frame, similar to a presentation window, may contain a number of presentation items, such as video clips and music background. These presentation items are stored in the resource object layer. In the frame object layer, each frame is associated with a number of attributes:

- *Name:* a unique name of the frame.
- *Keyword:* one or more keywords are used to describe the frame.
- *Inheritance links:* pointers to other frames that inherit properties from the current frame.
- *Usage links:* messages from the current frame to the destination frames, including possible parameters.
- *Aggregation links:* pointers to resources that are used in the current frame.
- *Presentation knowledge:* presentation properties and knowledge represented as logic facts, rules, and a query used in the frame.
- *Frame layouts:* screen coordinates of resources.

To create a high-quality multimedia presentation, one needs good multimedia resources as well as a good presentation design environment. Multimedia resources are recorded or captured via camera, tape recorder, or video camera, converted to their digital formats, and saved on disk. Each resource can be associated with a number of attributes. We use the following attributes for multimedia resources in the resource object layer of our database:

- *Name:* a unique name of the resource.
- *Keyword:* one or more keywords are used as the description of a multimedia resource. For instance, "Paris" is a keyword of a bitmapped picture of Paris.
- *Usage:* how the resource is used (e.g., background, navigation, or focus).
- *Medium:* what multimedia device is used to carry out this resource (e.g., sound, animation, MPEG-coded video, or picture).
- *Model:* how the resource is presented (e.g., table, map, chart, or spoken language).
- *Temporal endurance:* how long the resource lasts in a presentation (e.g., 20 or permanent).
- *Synchronization tolerance:* how does a participant feel about the synchronization delay of a resource. For instance, a user typically expects an immediate response after pushing a button for the next page of text; however, one might be able to tolerate a 2-s delay for a video playback.

- *Detectability:* how strongly a resource attracts attention (e.g., high, medium, or low).
- *Startup delay:* the time between a request and the presentation of the corresponding resource starts, especially when the resource is on a remote computer connected via a network.
- *Hardware limitation:* what kind of hardware is essential to present the resource ensuring a minimal quality of service (e.g., MPC level 1, level 2, level 3, or other limitations).
- *Version:* the version of this resource file.
- *Date/time:* the date and time this resource file was created.
- *Resolution:* the resolution of this resource file, specified as $X \times Y$ screen units, or 8-bit/16-bit for sound.
- *Start/end time:* for nonpermanent resources, the starting cycle and the ending cycle of the piece of video, sound, or other kind of resources. A cycle can be a second, one-tenth of a second, or an interval between two consecutive video frames of a video clip.
- *Resource descriptor:* a logical descriptor to a physical resource data segment on the disk.
- *Association links:* pointers to other resources who have the coexistence relation with the current resource.

The attributes in the two object layers are used in database queries to retrieve suitable objects for a multimedia presentation. These attributes are specified in a database browser when a query is requested by the user. Since each resource has a number of attributes, it would be cumbersome to require each query searching for a resource to contain all of these attributes. Thus, we propose an intelligent mechanism to simplify a query. The system contains several inference rules. Each rule describes an if–then relation between two attributes. For example, the following are some of the rules used in our system:

If usage = focus then detectability = high
If model = illustration then medium = picture
If medium = picture then temporal_endurance = permanent
If medium = MPEG then hardware_limitation = MPEG_card
If model = map then medium = picture
If ... etc.

In this way, nonspecified attributes can be deduced from others. Thus, a user does not need to specify all attributes of a resource when he/she is using a query to search for the resource.

Data mining has become a hot research topic in the community of database systems. We can analyze the dependencies among the above multimedia attributes and use data mining techniques to improve our system. For instance, we found that many presentation designers use a bit-mapped picture with a low detectability for background. A MIDI resource is often used as background music. We are constructing an interactive database subsystem to collect the ways that presentation designers use our database. Based on past usage, the subsystem may suggest a suitable resource to a designer.

A presentation is a heterogeneous collection of resource groups and frame groups. A multimedia class database is a heterogeneous collection of resource

object classes and frame object classes. Groups in a presentation are co-related since aggregation links and usage links are used among groups for resource access and navigation. However, in a class database, object classes are individual elements.

We are led to a two-layered approach by the nature of reusable objects. The reuse of a presentation script (i.e., in a frame group) and the reuse of multimedia resources (i.e., a resource group) are the two levels of reuse. Other reuse processes can be modeled by a combination of the two levels. For instance, the reuse of a whole presentation can be achieved by grouping the presentation as a frame group with several resource groups. The reuse of a single resource can be achieved by treating the single resource as a resource group. Therefore, the two-layered approach is a natural design. A multimedia presentation is a collection of frame groups and resource groups. Strictly speaking, a presentation is not a part of the database, at least from the database modeling perspective. Similarly, the underlying storage management layer is not a part of the database model. However, the two-layered model is to emphasize the two levels of object reuse.

There are also other models that can be used for presentations (e.g., the Petri net model). Petri nets, especially timed Petri nets, are suitable for interstream synchronization, which is not the mechanism used in our presentation system. Our system, on the other hand, relies on event-based synchronization with assistance from a message passing mechanism. Therefore, the Petri net model was not adopted.

We present an overview of the database storage management, which includes memory management and disk storage management. Memory management is used to run a presentation. A presentation contains a number of object groups (frame groups and resource groups). Each object group is a set, represented by a dynamic array that contains addresses of frames or resource descriptors. Object classes are also represented using dynamic arrays. A DAG is used to store frames and the inheritance links among them. At the resource object level, a digraph with bi-directional pointers (association links) is used to store resource descriptor addresses. Each resource descriptor contains a resource file name. The binary resource data are loaded at run time. This strategy avoids the dynamic memory from being over used by the huge amount of binary data. Pointers (aggregation links) to resource descriptors are stored in a frame for the frame to access its resources. The disk storage is rather complicated. Figure 5 pictures a simplified overview. The database server (MDBMS) allows multiple databases. Two types of databases are allowed: the presentation database (i.e., P.Database) and the object class database (i.e., C.Database). Each database has two files: a data file and an index file. A database may store a number of presentations or object classes. Each presentation contains several pointers to frame groups and resource groups. Each object group (frame or resource) contains a number of chained records. Similarly, an object class has a number of chained records. These records of resource groups (or resource object classes) are pointers to the BLOBs and attributes of multimedia resources, which can be shared. The index file has two types for presentation indices (P. Indices) and object class indices (C. Indices). Each part is a hashing table that contains data file offsets and other information, such as locking statuses.

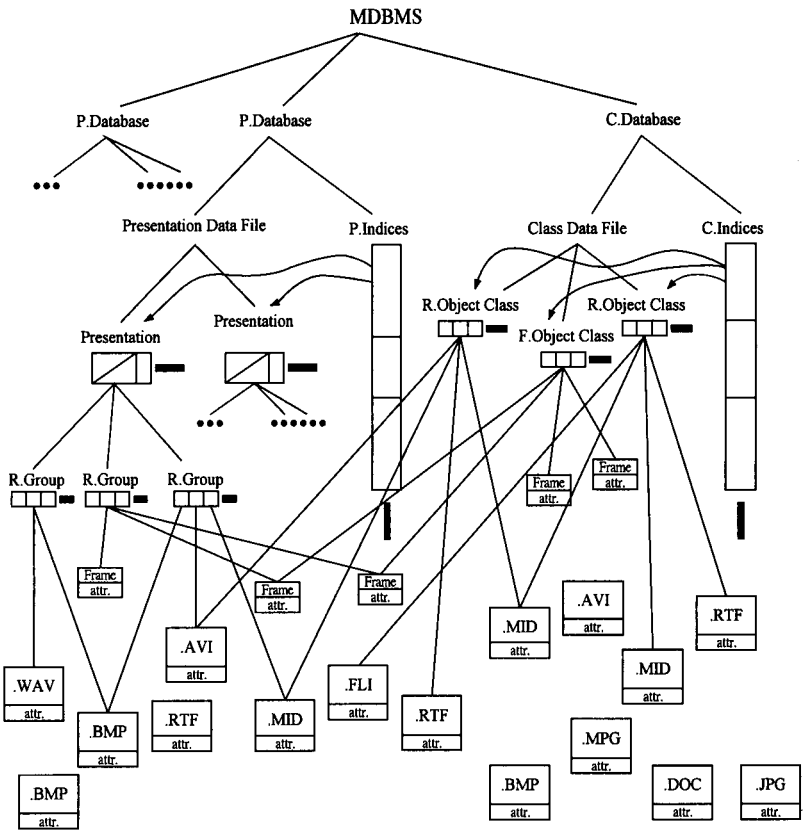


FIGURE 5 Disk storage management of the database.

III. DATABASE APPLICATIONS IN EDUCATION—A WEB DOCUMENT DATABASE

Multimedia Micro-University (MMU) is a joint research project with the participation of researchers from around the world. The primary goal of the MMU consortium is to develop technologies and systems for the use of virtual university. In this section, we propose the software architecture for a multimedia-based virtual course system. The architecture supports a multiplatform due to the availability of Web browsers and the Java virtual machine. We aim to provide a system on the Internet for instructors to design and demonstrate lectures. This system serves as a step toward our research goal—virtual university over the Internet. The primary directive of our MMU systems has four goals:

- *Adaptive to changing network conditions.* The system adapts to QoS requirements and network conditions to deliver different levels of service.
- *Adaptive to changing user needs.* Users are using the system from different perspectives. Types of users include students, instructors, and administrators. The system supports the demand of various kinds of information delivery from time to time.
- *Adaptive to Web-based environment.* The system is Web-savvy. That is, a standard Web browser is the only software required to students and

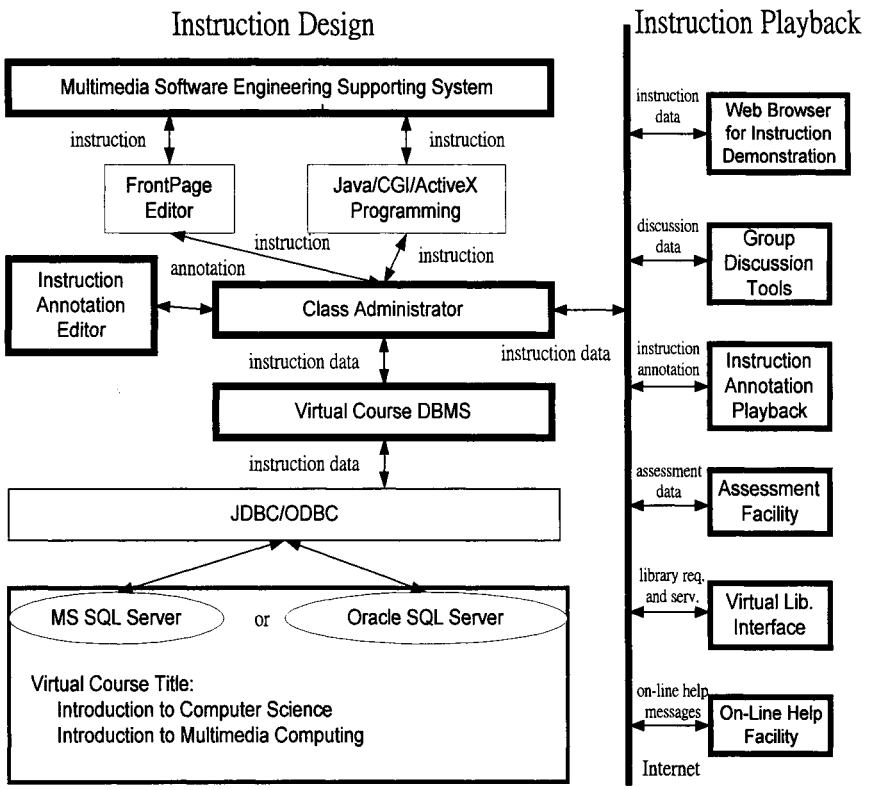


FIGURE 6 Overview of the MMU Project.

administrators. The instructors will use our system running on a Web browser in conjunction with some commercial available software.

- ***Adaptive to open architecture.*** A minimal compatibility is defined as the requirement for the open architecture. Compatibility requirements include presentation standard, network standard, and database standard.

Figure 6 illustrates the system architecture of our MMU system. On the instruction design side, we encourage instructors to use the Microsoft FrontPage Web Document editor, or an equivalent on a Sun workstation, to design virtual courses. Virtual courses may also be provided via some Java application programs, which are embedded into HTML documents. Since HTML and Java are portable languages, multiplatform courses are thus feasible. An instruction annotation editor, written as a Java-based daemon, is also running under the Java virtual machine (which is supported by Web browsers). This annotation daemon allows an individual instructor to draw lines, text, and simple graphic objects on top of a Web page. Different instructors can use the same virtual course but with different annotations. These annotations, as well as virtual courses, are stored as software configuration items (SCIs) in the virtual course database management system. An SCI can be a page showing a piece of lecture,

an annotation to the piece of lecture, or a compound object containing the above. A class administrator performs bookkeeping of course registration and network information, which serves as the front end of the virtual course DBMS. The implementation of the virtual course DBMS uses JDBC (or ODBC) as the open database connection to some commercial available database systems, such as the MS SQL server, Sybase, Informix, or Oracle servers. Currently, our system uses the MS SQL server.

On the other side of the system architecture, a student can use an ordinary Web browser to traverse virtual lectures. However, some underlying subsystems are transmitted to a student workstation to allow group discussions, annotation playback, and virtual course assessment. We also provide an interface to the virtual course library. The interface allows object searching and browsing. These subsystems, again, are written as Java-based daemons running under the Internet Explorer or the Netscape Navigator. Help facilities are provided.

This research project, besides providing a prototype system for virtual university realization, also focuses on some research issues in multimedia computing and networking. From the perspective of software engineering, several paradigms were used in software development (e.g., water-fall model, spiral model, object-oriented model). Can these models be applied to multimedia course development? Or, can we refine these models to be used? On the other hand, how we estimate the complexity of a course and how we perform a white box or black box testing of a multimedia presentation are research issues that we have solved partially. From the perspective of CSCW, the virtual course system maintains the smooth collaboration and consistency of distributed course designs. A software configuration management system allows checking in/out of course components and maintains versions of a course. All instruction systems require assessment. The virtual course system has methodologies to support the evaluation of student progress and achievement.

To support the storage requirement of our Web document development paradigm, we have designed a Web document database. We use an off-the-rack relational database system as the underlying supporting system. In this section, we discuss the design considerations of our database system. We design the database based on three objectives: object reuse, object distribution, and resource sharing. In this subsection, a three-layered database hierarchy is proposed (see Fig. 7). In the Web document DBMS, multiple Web document databases are allowed. Each database can have a number of documents. Each document is identified by a unique script name. A script, similar to a software system specification, can describe a course material or a quiz. With respect to a script, the instructor can have different tries of implementation. Each implementation contains at lease one HTML file, and some optional program files, which may use some multimedia resources. A course document is used by several instructors. An instructor can use our annotation tool to draw lines and text to add notes to a course implementation. Thus, an implementation may have different annotations created by different instructors. To test the implementation, test records are generated for each implementation. Also bug reports are created for each test record. The database has three layers. Objects in the three layers contain the following attributes:

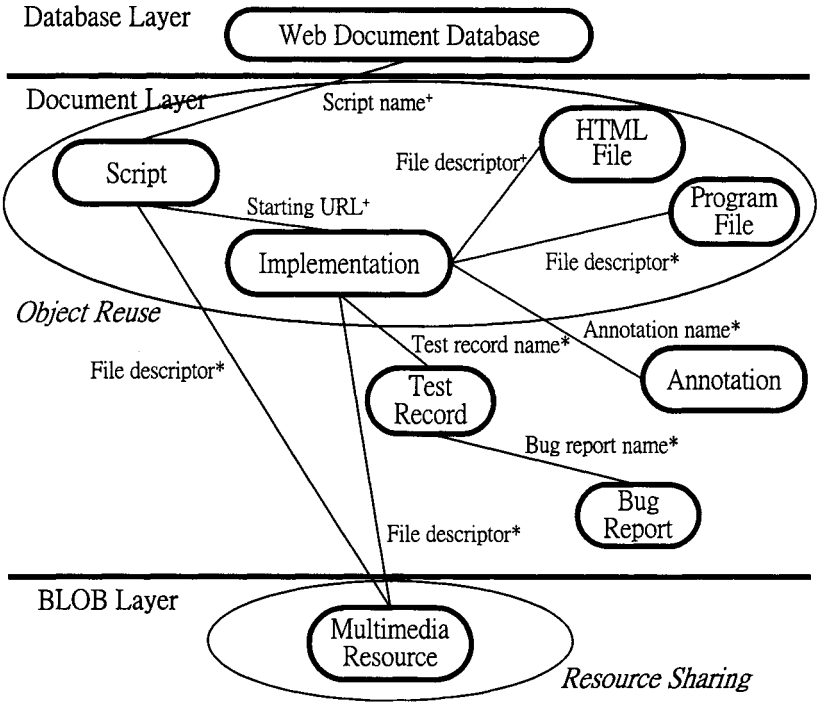


FIGURE 7 The three-layered database architecture of the MMU system.

Database Layer

- ***Database name:*** a unique name of the database.
- ***Keywords:*** one or more keywords are used to describe the database.
- ***Author:*** author name and copyright information of the creator.
- ***Version:*** the version of this database.
- ***Date/time:*** the date and time this database was created.
- ***Script names:*** pointers to script tables belong to the database.

Document Layer

- ***Script table:*** content of a script object.
- ***Implementation table:*** content of an implementation object.
- ***Test record table:*** content of a test record.
- ***Bug report table:*** content of a bug report.
- ***Annotation table:*** content of an annotation.
- ***HTML files:*** standard HTML files used in the implementation.
- ***Program files:*** add-on control program files used in the implementation.
- ***Annotation files:*** Annotation files that store document annotations.

BLOB Layer

- ***Multimedia sources:*** multimedia files in standard formats (i.e., video, audio, still image, animation, and MIDI files). Objects in this layer are shared by instances and classes.

In the database hierarchy, objects in each layer are represented as rounded boxes. Each object may be linked to other relative objects. A link in the hierarchy is associated with a label, which has a reference multiplicity indicated in its superscript. A "+" sign means the use of one or more objects, and a "*" sign represents the use of zero or more references. Database objects can be reused. The ellipses in the document layer indicate that a number of database objects are grouped into a reusable component. The component can be duplicated to another compound object with modifications. However, the duplication process involves objects of relatively smaller sizes, such as HTML files. The ellipses in the BLOB layer indicate that, BLOBs in large sizes are shared by different compound objects, including different scripts and implementations. However, BLOB resource sharing is limited to a workstation. Upon demand, BLOB objects may be duplicated in other workstations in order to realize real-time course demonstration.

The document layer contains the most important items of a Web document. Since we use a relational database management system to implement our object hierarchy, we summarize some content of major tables here.

Script Table

- ***Script name:*** a unique name of the document script.
- ***Keywords:*** keywords of the script.
- ***Author:*** the author of the document.
- ***Version:*** the version of the document.
- ***Date/time:*** the creation date and time.
- ***Description:*** the content of the script, which is described in text. However, the author may have a verbal description stored in a multimedia resource file.
- ***Expected date/time of completion:*** a tentative date of completion.
- ***Percentage of completion:*** the status of current work.
- ***Multimedia resources:*** file descriptors point to multimedia files.
- ***Starting URLs:*** foreign key to the implementation table.
- ***Test record names:*** foreign key to the test record table.
- ***Bug report names:*** foreign key to the bug report table.
- ***Annotation names:*** foreign key to the annotation table.

Implementation Table

- ***Starting URL:*** a unique starting URL of the Web document implementation.
- ***HTML files:*** implementation objects such as HTML or XML files.
- ***Program files:*** implementation objects such as Java applets or ASP programs.
- ***Multimedia resources:*** implementation objects such as audio files.
- ***Script name:*** foreign key to the script table.
- ***Test record names:*** foreign key to the test record table.
- ***Bug report names:*** foreign key to the bug report table.
- ***Annotation names:*** foreign key to the annotation table.

Test Record Table

- *Test record name:* a unique name of the test record.
- *Testing scope:* local or global.
- *Web traversal messages:* windowing message controls Web document traversal.
- *Script name:* foreign key to the script table.
- *Starting URL:* foreign key to the implementation table.
- *Bug report names:* foreign key to the bug report table.

Bug Report Table

- *Bug report name:* a unique name of the bug report.
- *Quality assurance engineer:* name of the QA person.
- *Test procedure:* a simple description of the test procedure.
- *Bug description:* the test result.
- *Bad URLs:* a number of URLs that cannot be reached.
- *Missing objects:* multimedia or HTML files missing from the implementation.
- *Inconsistency:* a text description of inconsistency.
- *Redundant objects:* a list of redundant files.
- *Test record name:* foreign key to the test record table.

Annotation Table

- *Annotation name:* a unique name of the annotation.
- *Author:* the author of the annotation.
- *Version:* the version of the annotation.
- *Date/time:* the creation date and time.
- *Annotation file:* a file descriptor to an annotation file.
- *Script name:* foreign key to the script table.
- *Starting URL:* foreign key to the implementation table.

The Web document database, when updated, should be proceeded in a consistent way. Each Web document SCI has a number of references. We use these references to maintain the referential integrity of the database. Figure 8 illustrates a referential integrity diagram. Each link in the diagram connects two objects. If the source object is updated, the system will trigger a message, which alerts the user to update the destination object. Each link in the diagram is associated with a label, with various possible alert messages:

- 1: one message
- *: zero or more messages
- +: one or more messages

For instance, if a script SCI is updated, the corresponding implementations should be updated, which further triggers the changes of one or more HTML programs, zero or more multimedia resources, and some control programs.

Due to the locking mechanism used in object-oriented database systems, we have defined an object-locking compatibility table. In general, if a container has a read lock by a user, its components (and itself) can have the read access by another user, but not the write access. However, the parent objects of the

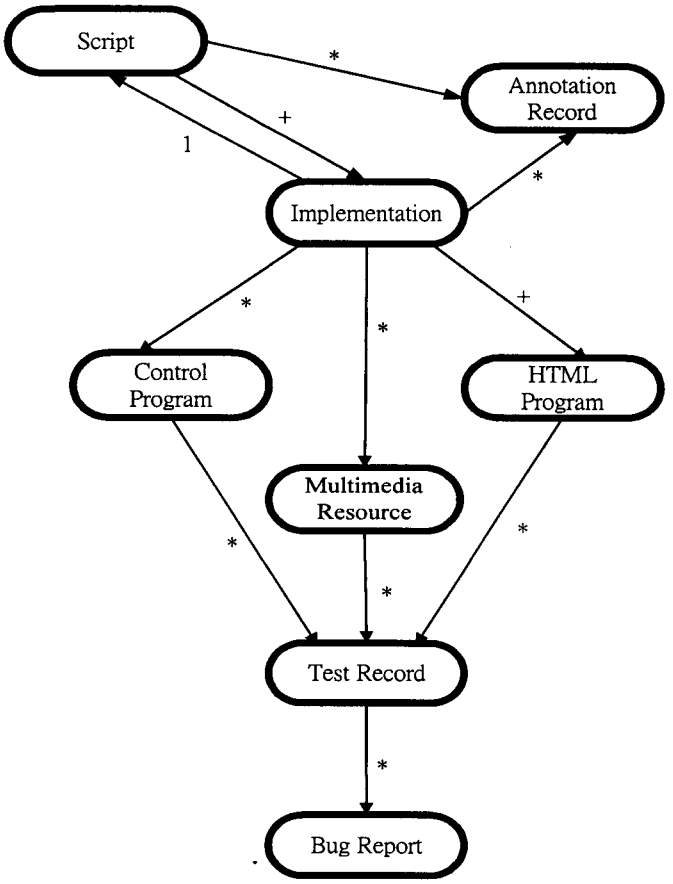


FIGURE 8 Referential integrity diagram.

container can have both read and write access by another user. Of course, the accesses are prohibited in the current container object. Locking tables are implemented in the instructor workstation. With the table, the system can control which instructor is changing a Web document. Therefore, collaborative work is feasible.

Web documents are reusable. Among many object reuse paradigms, classification and prototyping are the most common ones. Object classification allows object properties or methods at a higher position of the hierarchy to be inherited by another object at a lower position. The properties and methods are reused. Object prototyping allows reusable objects to be declared as templates (or classes), which can be instantiated to new instances. A Web document in our system contains SCIs for script, implementation, and testing. As a collection of these three phrases of objects, a Web document is a prototype-based reusable object. Object reuse is essentially important to the design of Web documents. However, the demonstration of Web documents may take a different consideration due to the size and the continuous property of BLOB.

Web documents may contain BLOB objects, which are infeasible to be demonstrated in real-time when the BLOB objects are located in a remote station due to the current Internet bandwidth. However, if some of the BLOB

objects are preloaded before their presentation, even though the process involves the use of some extra disk space, the Web document can be demonstrated in real-time. However, BLOB objects in the same station should be shared as much as possible among different documents. We aim to provide a system to make distributed Web documents to be reused in a reasonable efficient manner.

The design goal is to provide a transparent access mechanism for the database users. From different perspectives, all database users look at the same database, which is stored across many networked stations. Some Web documents can be stored with duplicated copies in different machines for the ease of real-time information retrieval. A Web document may exist in the database at different physical locations in one of the following three forms:

- Web document class
- Web document instance
- Web document reference to instance

A document class is a reusable object, which is declared from a document instance. A document instance may contain the physical multimedia data, if the instance is newly created. After the declaration of the document instance, the instance creates a new document class. The newly created class contains the structure of the document instance and all multimedia data, such as BLOBs. The original document instance maintains its structure. However, pointers to multimedia data in the class is used instead of storing the original BLOBs. When a new document instance is instentiated from a document class, the structure of the document class is copied to the new document instance and pointers to multimedia data are created. This design allows the BLOBs to be stored in a class. The BLOBs are shared by different instances instantiated from the class.

A document instance is a physical element of a Web document. When a database user looks at the Web document from different network locations, the user can access the Web document in two ways. The first is to access the document directly. The second mechanism looks at the document via document reference. A document reference to instance is a mirror of the instance. When a document instance is created, it exists as a physical data element of a Web document in the creation station. References to the instance are broadcast and stored in many remote stations.

When a document instance is retrieved from a remote station more than a certain amount of iterations (or more than a watermark frequency), physical multimedia data are copied to the remote station. The duplication process may include the duplication of document classes, which contain the physical BLOBs.

The duplication process is proceeded according to a hierarchy distribution strategy. Assuming that, N networked stations join the database system in a linear order. We can arrange the N stations in a full m-ary tree according to a breadth first order. A full m-ary tree is a tree with each node containing exactly m children, except the trailing nodes. The nth station, where $1 <= n <= N$, in the linear joining sequence has its ith child, where $1 <= i <= m$ at the following position in the linear order:

$$m*(n-1)+i+1.$$

In a Web document system utilizing a distance learning system, an instructor can broadcast lectures to student workstations. Essentially, the broadcast process is a multicasting activity. With the appropriate selection of m, the transmission of physical data can be proceeded in an efficient manner, starting from the instructor station as the root of the m-ary tree. The implementation of this multicasting system has a broadcast vector containing a linear sequence of workstation IP addresses. The system maintains the sizes of m's, based on the number of workstations and the physical network bandwidth for different types of multimedia data. This design achieves one of our project goals: adaptive to changing network conditions.

On the other hand, a student can look at an off-line lecture presentation prepared by the instructor. In this case, the instructor station serves as a lecture server. Lecture presentations are transmitted to student workstations upon demands. The broadcast route can use an inverse function of the above expression. The kth station, where $1 <= k <= N$, in the linear joining sequence has its unique parent at the following position in the linear order:

$$(k - i - 1)/m + 1, \quad \text{where} \quad i = (k - 1) \bmod \text{m}, \quad \text{if} \quad i \mid m;$$
$$i = m, \text{ otherwise.}$$

The duplication of lecture presentations is upon demand. A child node in the m-ary tree copies information from its parent node. However, if a workstation (and its child workstations) does not review a lecture, it is not necessary to duplicate the lecture. The station only keeps a document reference in this case. Since the duplication process may involve extra disk space, one may argue that disk spaces are wasted. However, the duplicated document instances live only within a period of time. After a lecture is presented, duplicated document instances migrate to document references. Essentially, buffer spaces are used only. However, the instructor workstation has document instances and classes as persistence objects. The above equations are proved by mathematical induction and double-induction techniques. They are also implemented in our system.

Another issue of object propagation is that objects in the BLOB layer of the database are shared by objects at a higher level in the hierarchy. That is, in both the instructor station and the student station, BLOBs are shared among different lecture presentations. Since an individual multimedia resource is used only by a presentation in a workstation with respect to a time duration, concurrent access is not a consideration. This strategy avoids the abuse of disk storage.

As discussed previously, the database has three sorts of objects: classes, instances, and references. Document classes support object reuse. Instances are physical objects of a Web document, which are referred by document references. In the proposed virtual university architecture, in order to support off-line learning, we encourage students to "check out" lecture notes from a virtual library. Web document instance are stored in the virtual library. An instructor has a privilege to add or delete document instances, which contain lecture notes as Web pages. Students can check out and check in these Web pages. However, in general, there is no limitation of the number of Web pages to be checked out. The check in/out procedure serves as an assessment criterion to the study performance of a student. We provide a browsing interface, which allows students to retrieve course materials according to matching keywords, instructor

names, and course numbers/titles. This virtual library is Web-savvy. That is, the searching and retrieve processes are running under a standard Web browser. The library is updated as needed. The mechanism follows another guidance of our project goals: adaptive to changing user needs. We are developing three Web courses based on the virtual library system: introduction to computer engineering, introduction to multimedia computing, and introduction to engineering drawing.

Distance learning, virtual university, or remote classroom projects change the manner of education. With the tremendous growing amount of Internet users, virtual university is a step toward the trend of future university. However, most development of distance learning systems relies on the high bandwidth of a network infrastructure. As it is not happening everywhere on the Internet to meet such a high requirement, it is worthwhile to investigate mechanisms to cope with the real situation. Even in the recent future, with the next generation of Internet, the increasing amount of users consumes an even higher network bandwidth. The primary directive of the Multimedia Micro-University Consortium is looking for solutions to realize virtual university. Some of our research results, as pointed out in this chapter, adapt to changing network conditions. Using an off-line multicasting mechanism, we implemented a distributed virtual course database with a number of on-line communication facilities to fit the limitation of the current Internet environment. The proposed database architecture and database system serves as an important role in our virtual university environment. We are currently using the Web document development environment to design undergraduate courses including introduction to computer science and others.

IV. FUTURE DIRECTIONS

Future multimedia applications will be even more attractive. For instance, intelligent agents, based on artificial intelligence techniques, will further assist multimedia software to interact with its users. Also, intelligent searching mechanisms will add power to content-based retrieval of multimedia information. Advanced input/output devices, such as head-mounted displays, sensors, auditory displays, and haptic displays, will further improve the efficiency of human–computer communication. High-speed processors, high-performance multimedia hardware architectures and networks, and high-capacity optical storage will support the need of high-quality multimedia applications in the future.

APPENDIX A: THE DESIGN AND IMPLEMENTATION OF IMMPS

The Multimedia Information Network (MINE) Lab at Tamkang University focuses on the development of many research projects related to multimedia presentation and MDBMS. In the past few years, we have developed several multimedia presentation design systems. In this appendix, we discuss two such systems in terms of database design and the graphical user interface. The first is an intelligent presentation system used in product demonstrations. The second is a Web-based documentation development environment.

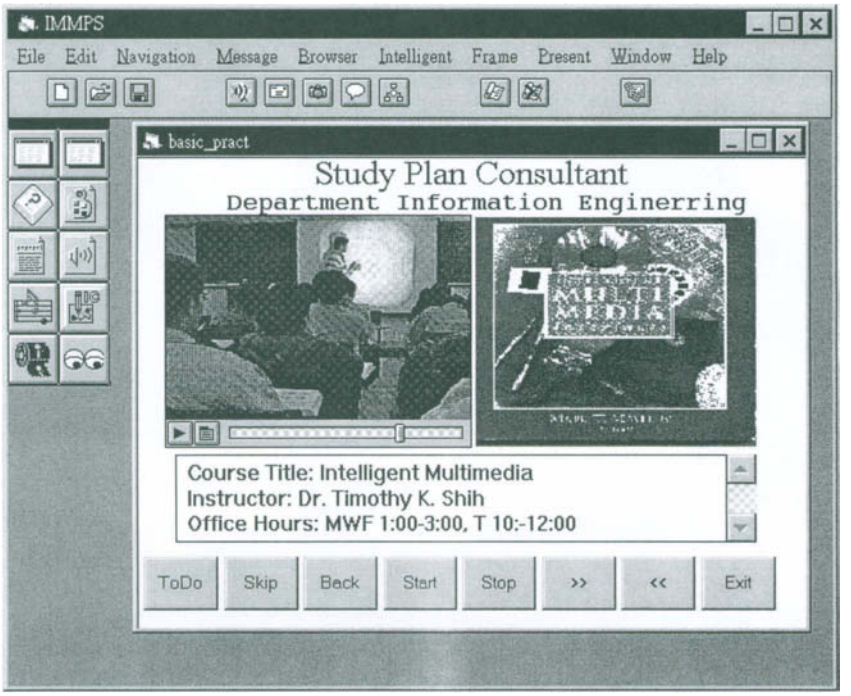


FIGURE A1 The main window of IMMPS.

We present the graphical user interface of IMMPS. The first window is the main window of IMMPS shown in Fig. A1. This window allows the user to drop various presentation topics in the design area shown in the subwindow. Presentation topics may include text, audio, etc. shown as buttons on the left side of the window. The main window is used in conjunction with the presentation knowledge inheritance window and the presentation messages passing window shown in Fig. A2. The knowledge inheritance hierarchy is constructed by using different functions shown in the menu bar of the window. Inheritance links among presentation window objects are thus created as the inheritance relations are declared.

To add a usage (or message) link, the user accesses the presentation messages passing window. A vertical bar represents a presentation window indicated by its name on the top of that bar. Messages are passed when a button in a presentation window is pushed (or, as a side effect of a knowledge inference). Push buttons are shown as boxes on vertical bars. Each message, with its name displayed as the label of a usage link (shown as a horizontal arrow), is associated with one or more parameters separated by commas. These parameters are entered in the message window shown in the lower-left corner of Fig. A2.

To add rules or facts to the knowledge set of a presentation window, the presentation intelligence window shown in Fig. A3 is used. When a message is received by a presentation window, the corresponding actions that the presentation window needs to perform will be entered in the presentation action control window in Fig. A3. The presentation button control window is used to specify actions associated with each button when it is clicked.

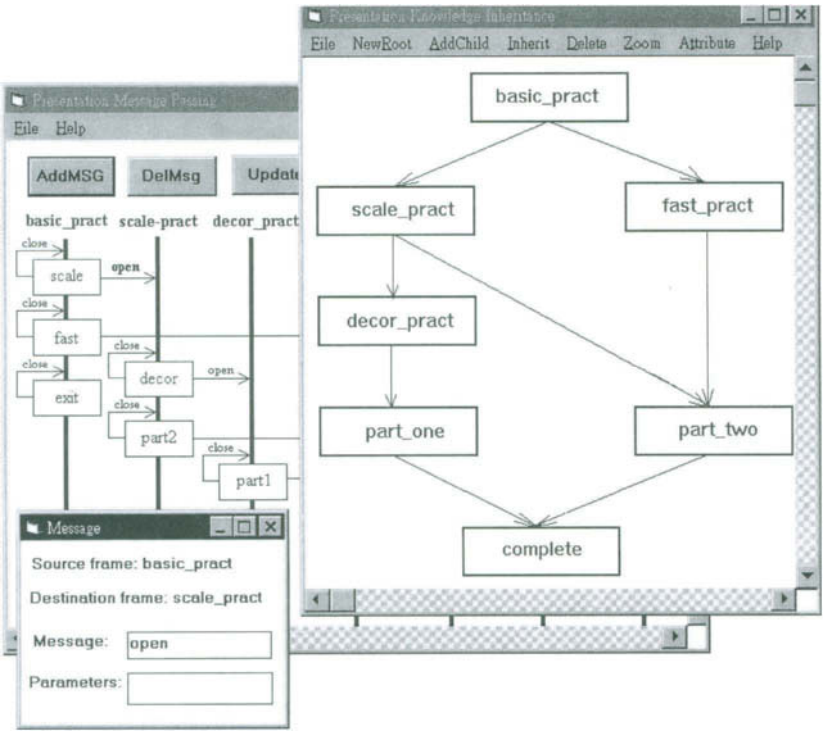


FIGURE A2 The presentation knowledge inheritance window and the presentation message passing window.

A multimedia resource browser (Fig. A4) is designed to allow a user to retrieve resources ready for inclusion in presentations. Resource attributes are selected or entered in different components of the window. When the topic button is pushed, the highlighted resource in the list box is linked to a topic of a presentation window.

Objects' reuse of presentation windows and resources is controlled by the presentation reuse control window shown in Fig. A5. Each presentation window object class or resource object class contains a number of presentation windows or resources shown in the middle list box of the individual subwindow in the presentation reuse control window. The association links are declared in the multimedia resource association control window shown in Fig. A5. Each highlighted resource (shown in the first list box from the left) is linked to a number of resources shown in the second list box. The available resources are shown in the third list box.

We use Arity Prolog as the underlying implementation language of our inference engine. However, some C++ functions are used to take care of presentation navigation and serve as stub functions that communicate between Prolog predicates and Basic procedures. The graphical user interface is written in Visual Basic (VB). The database management system is also implemented in Visual C++ (VC). No other supporting software is used, besides the underlying database management system and Windows 95/3.1. Figure A6 illustrates the implementation environment of our prototype system.

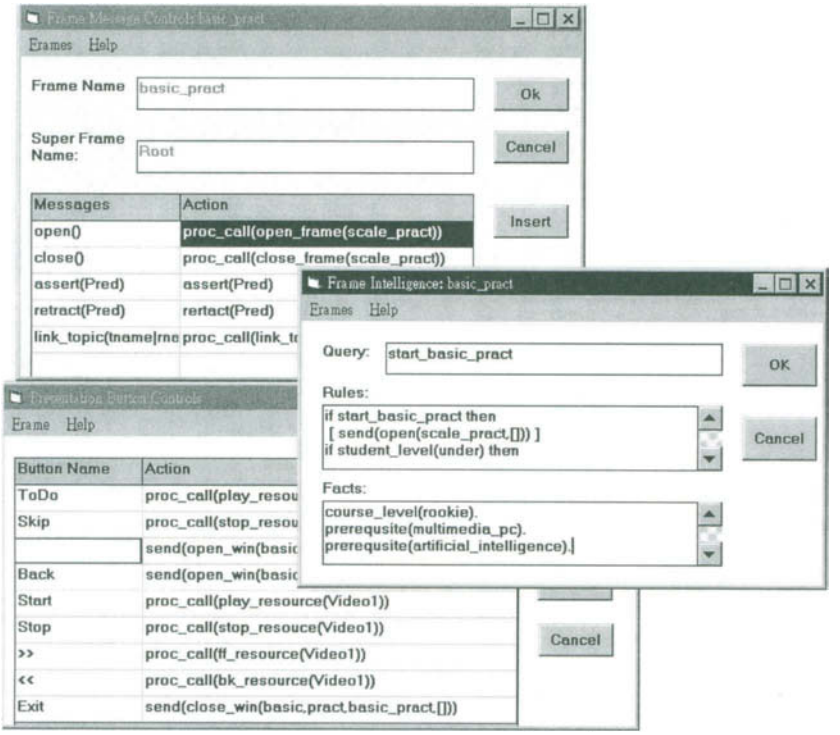


FIGURE A3 The presentation intelligence window, the presentation action control window, and the presentation button control window.

We spent about two years developing the system. In the first half of a year, we surveyed other presentation systems and researches before documenting a draft specification. The functional specification and design of IMMPS take about nine months. Two Ph.D. students and three Masters students spent about almost a year in the implementation, before the prototype was tested by a few undergraduate students for one month. About 9000 lines of VB, 4000 lines of VC, and another 4000 lines of Prolog code were used.

APPENDIX B: THE DESIGN AND IMPLEMENTATION OF MMU

We have implemented a collection of tools to support distance learning of MMU. Figure A7 illustrates a Web document development environment. The user starts from the script design phrase of the spiral model. Requirements of the document script as well as auxiliary information is given in the Web Script window (see the left of Fig. A7). According to the script, the Microsoft FrontPage editor is used to generate the Web document implementation shown in the Web browser. While the user is designing the script, he/she can look at some multimedia resources via the resource icons. In the implementation phrase, the user also needs to provide auxiliary information shown in the Web implementation window in Fig. A8.

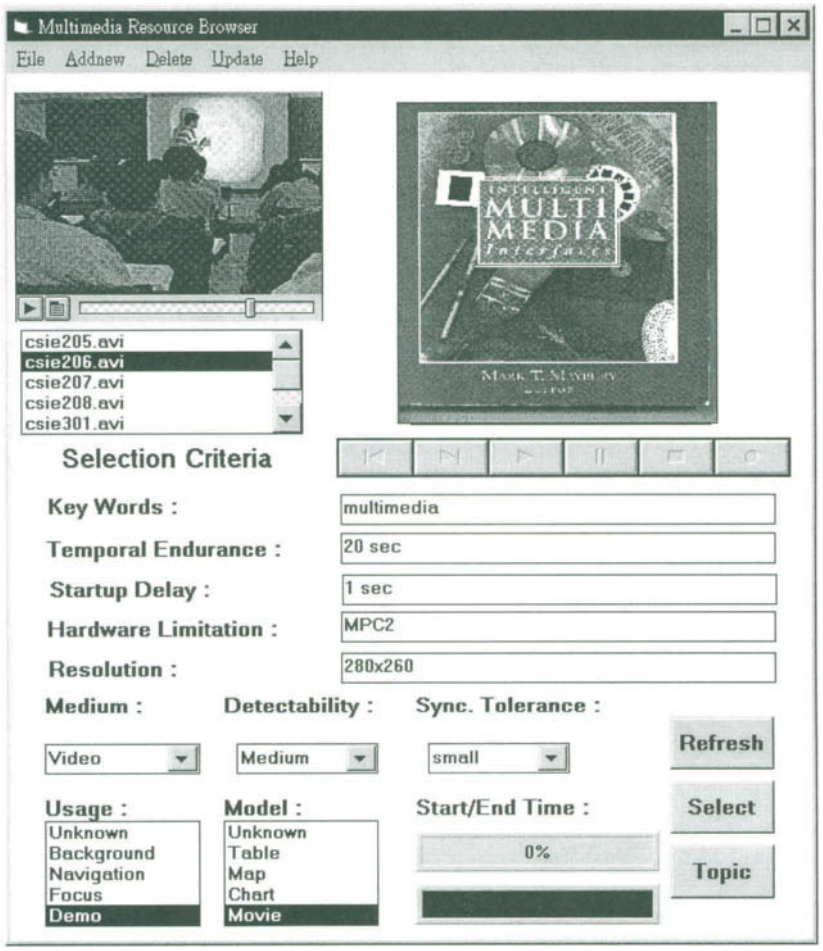


FIGURE A4 The multimedia resource browser.

The development process may proceed through several cycles in the spiral model that we have proposed. As the Web document evolves, the user may start from a Web document only containing the script portion. The implementation and test records are omitted at the beginning. In the next iteration, the user uses the FrontPage editor to design his/her Web page and add intradirectory and intrastation hyperlinks. The test record has a local testing scope in this cycle. Assessment is conducted to decide the status of the document, including checking the percentage of completeness and inconsistency, to decide whether to proceed with the next iteration. When the user starts to add interstation hyperlinks in the next cycle, the testing scope should be changed. Assessment is then performed again to decide whether to post the Web page. The evolution process of a Web document can be assessed from another perspective. A document may start from an implementation without add-on control programs. Gradually, the user adds Java applets. Regression tests should be proceeded at each cycle.

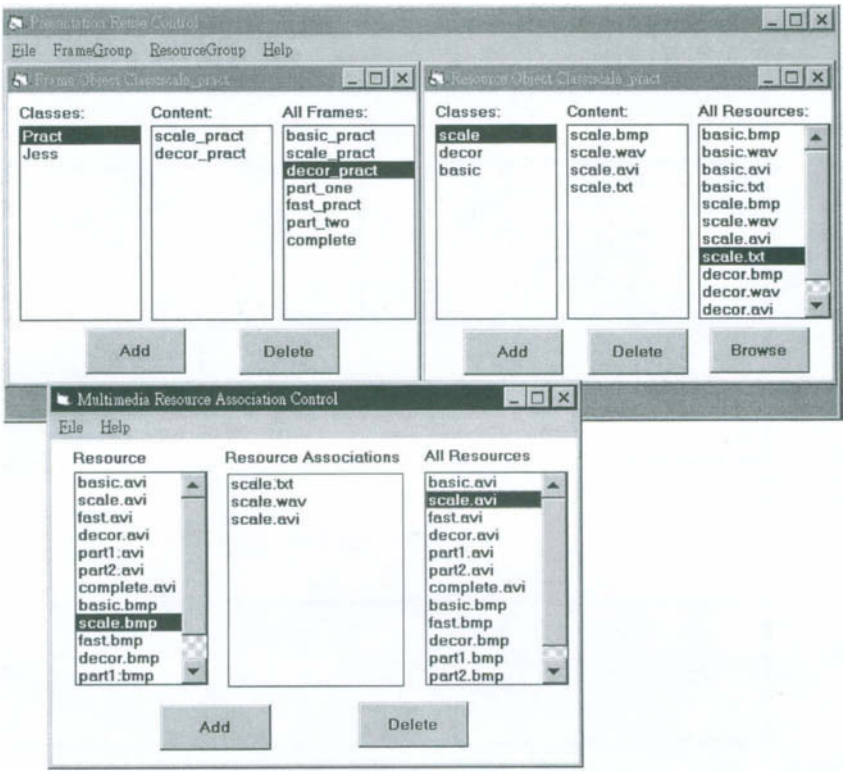


FIGURE A5 The presentation reuse control window and the multimedia resource association control window.

Another way to look at the evolution is that the user can start from a Web document without image and audio records. The user can add images in the next cycle, and audio records in the following cycle, and so on (for video clips). The spiral model can be used in different types of Web document evolutions.

After a Web document is developed, the instructor can use the document annotation editor (shown in Fig. A9) to explain his/her lecture. The annotations, as well as audio descriptions, can be recorded with time stamps encoded for real-time playback. Note that a Web document (serving as a lecture) can be used by different instructors with different annotations.

Even though the annotation system allows instructors to explain course materials, in a virtual university system, it is necessary to have on-line discussion tools for students to ask questions and conduct group discussions. Figures A10 and A11 illustrate a chat room tool, which allows multiple users to send messages to each other. The chat room tool is written in Java running on a standard Web browser. The chat room tool, as well as a white board system (see Fig. A12), has four floor control modes:

- ***Free access:*** All participants can listen and talk.
- ***Equal control:*** Only one person can talk but all participants can listen. An individual user sends a request to the speaker for the floor. The speaker

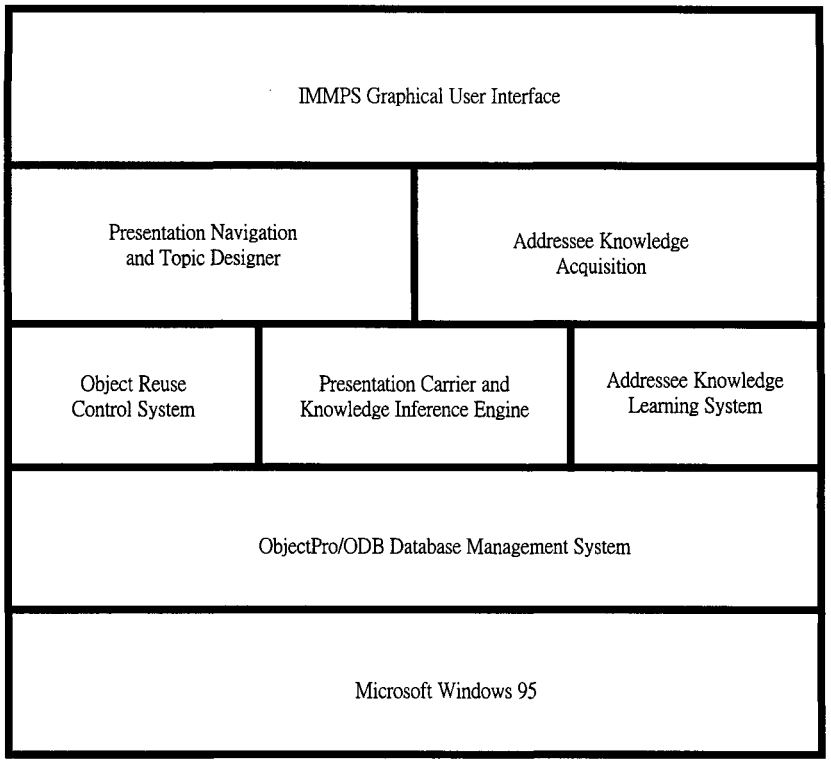


FIGURE A6 The implementation environment of IMMPS.

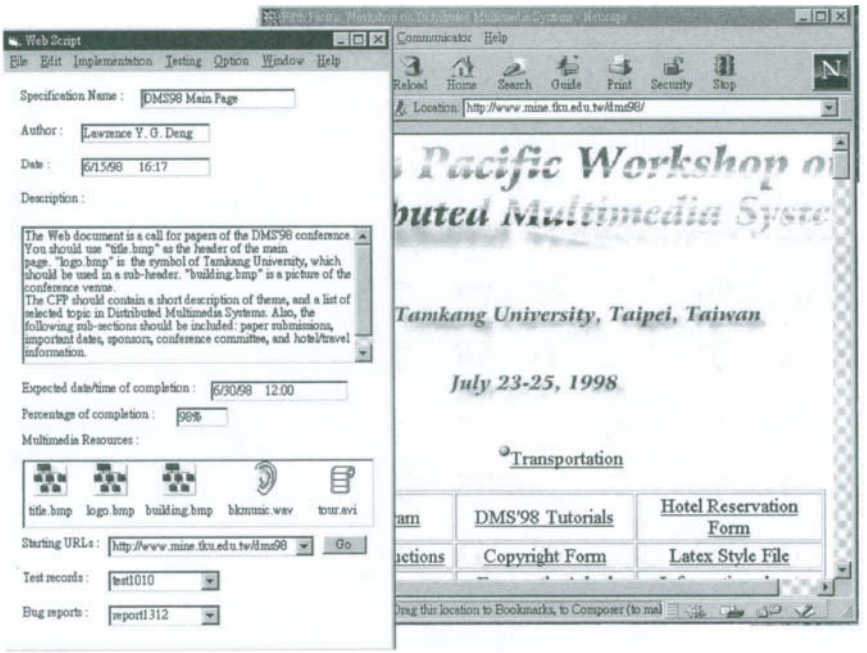


FIGURE A7 A Web document script and its implementation.

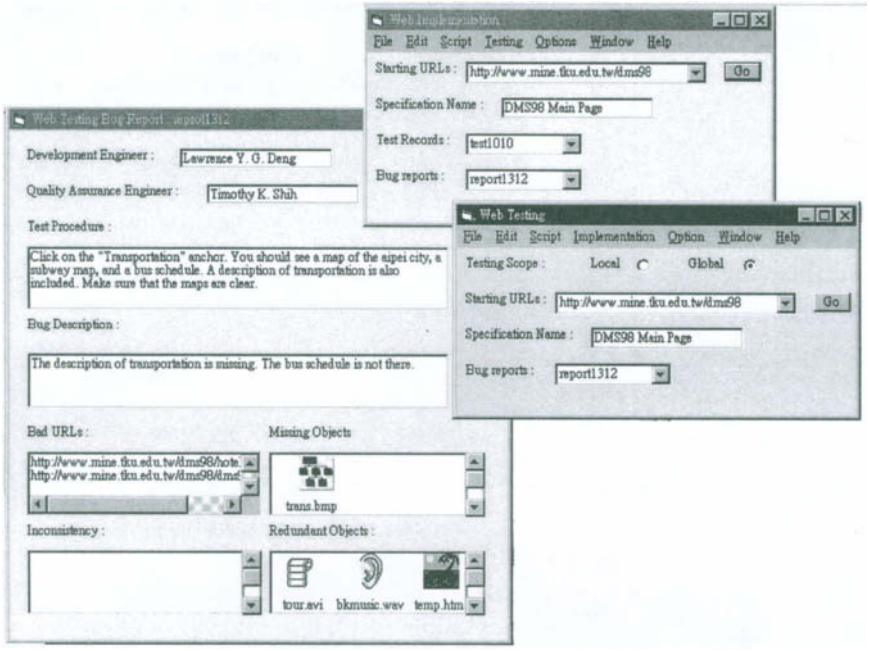


FIGURE A8 A Web document testing environment.

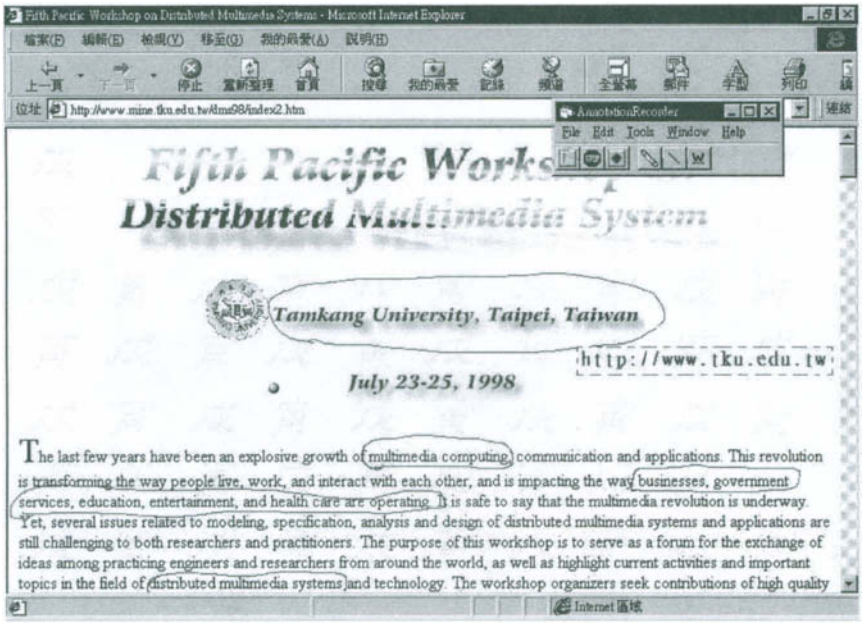


FIGURE A9 A Web course annotation system.

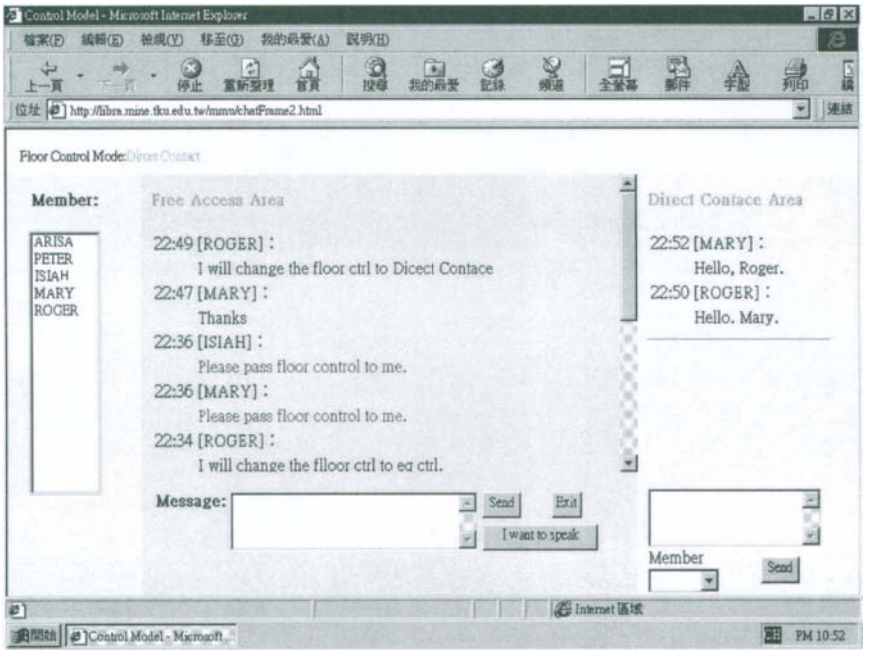


FIGURE A10 A chat room tool.

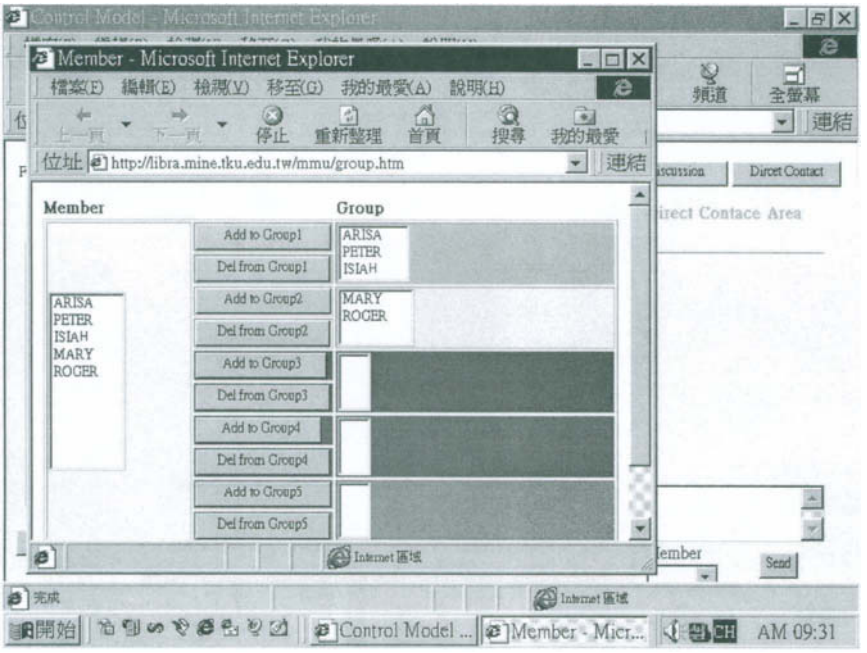


FIGURE A11 The group association control window.

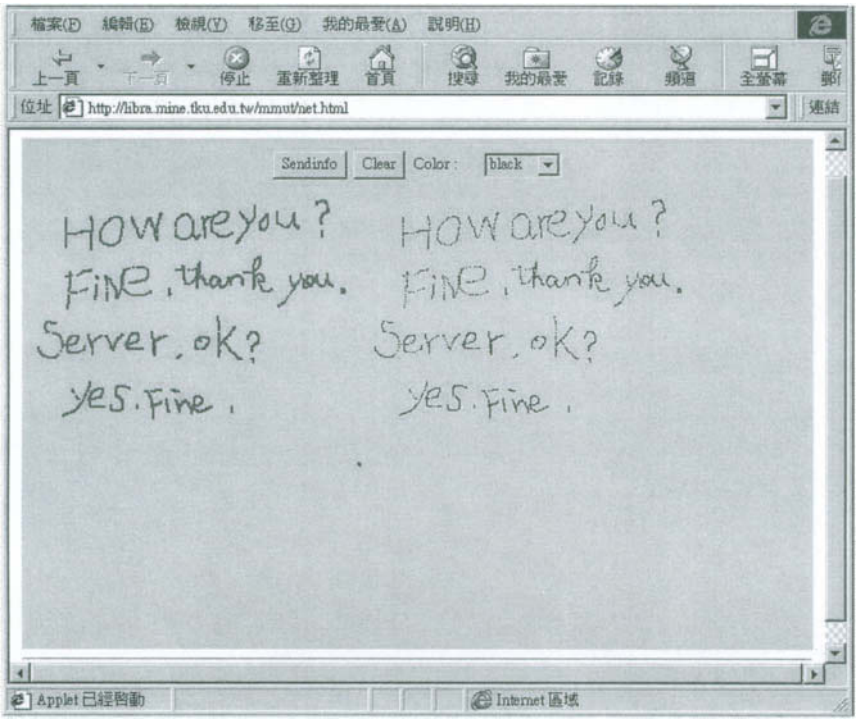


FIGURE A12 A white board system.

grants the control to the individual (first come, first serve-based). The first person logged in into the chat room has the first floor control. No chairperson is assigned. That is, everyone has the same priority.

- ***Group discussion:*** A participant can select a group of persons to whom he/she wants to talk and agrees to another group to whom he/she wants to listen. When an individual is asked to listen to another person, the individual can decide if he/she wants to listen. If so, the name of this individual is added to the listen group of the talking person.
- ***Direct contact:*** Two persons talk and listen to each other (i.e., private discussion). This is for a private conversation in the direct contact area illustrated in the GUI below. A person can have a private discussion with his/her partner while still joining the chat room.

The control mode of the discussion is decided by the course instructor. Drawing works the same as chatting in the three modes. Direct contact can be turned on/off by the instructor.

We use a relational database management system to implement our object-oriented database. The following database schema is implemented on the MS SQL server. However, a similar schema can be used in the Sybase or the Oracle database server. The schema can be used by ODBC or JDBC applications, which serve as open connections to various types of other database servers.

```
CREATE TABLE Database_table
(NAME                    CHAR        NOT NULL,
 AUTHOR                  CHAR,
 VERSION                 CHAR,
 DATE/TIME               DATETIME,
 SNAME                   CHAR,
 PRIMARY KEY (NAME),
 UNIQUE(NAME, AUTHOR, VERSION)
)
CREATE TABLE Script_table
(SNAME                   CHAR        NOT NULL,
 AUTHOR                  CHAR,
 VERSION                 INT,
 DATE_TIME               DATETIME,
 DESCRIPTION             CHAR,
 E_DATE_E_TIME           DATETIME,
 P_OF_COMPLETION         INT,
 PRIMARY KEY(SNAME),
 UNIQUE(SNAME, AUTHOR, VERSION, DESCRIPTION)
)
CREATE TABLE Implementation_table
(STARTING_URL            CHAR        NOT NULL,
 SNAME                   CHAR        NOT NULL,
 PRIMARY KEY(STARTING_URL),
 FOREIGN KEY(SNAME) REFERENCES Script_table(SNAME),
 UNIQUE(STARTING_URL,SNAME)
)
CREATE TABLE Test_Record_table
(TNAME                   CHAR        NOT NULL,
 SNAME                   CHAR        NOT NULL,
 TEST_SCOPE              CHAR,
 STARTING_URL            CHAR        NOT NULL,
 PRIMARY KEY(TNAME),
 FOREIGN KEY(SNAME) REFERENCES Script_table(SNAME),
 FOREIGN KEY(STARTING_URL) REFERENCES
    Implementation_table(STARTING_URL),
 UNIQUE(TNAME, SNAME, STARTING_URL)
)
CREATE TABLE Bug_Report_table
(BNAME                   CHAR        NOT NULL,
 TEST_PROCEDURE          CHAR,
 BUG_DESCRIPTION         CHAR,
 TNAME                   CHAR,
 PRIMARY KEY(BNAME),
 FOREIGN KEY(TNAME) REFERENCES
    Test_Record_table(TNAME),
 UNIQUE(BNAME, TEST_PROCEDURE, BUG_DESCRIPTION, TNAME)
)
```

```
CREATE TABLE Annotation_table
(ANAME                  CHAR        NOT NULL,
 SNAME                  CHAR,
 STARTING_URL           CHAR,
 AUTHOR                 CHAR,
 VERSION                INT,
 DATE_TIME              DATETIME,
 ANNOTATION_FILE        CHAR,
 PRIMARY KEY(ANAME),
 FOREIGN KEY(SNAME) REFERENCES Script_table(SNAME),
 FOREIGN KEY(STARTING_URL) REFERENCES
   Implementation_table(STARTING_URL),
 UNIQUE(ANAME, SNAME, STARTING_URL, AUTHOR,
   ANNOTATION_FILE, VERSION)
)
CREATE TABLE Resource_table*
(R_ID                   INT         NOT NULL,
 NAME                   CHAR,
 AUTHOR                 CHAR,
 DATE_TIME              DATETIME,
 SIZE                   INT,
 DESCRIPTION            CHAR,
 LOCATION               CHAR        NOT NULL,
 TYPE                   CHAR,
 PRIMARY KEY(R_ID),
 UNIQUE(R_ID, AUTHOR, NAME, LOCATION, TYPE)
)
CREATE TABLE Html_table*
(H_ID                   INT         NOT NULL,
 NAME                   CHAR,
 AUTHOR                 CHAR,
 DATE_TIME              DATETIME,
 DESCRIPTION            CHAR,
 LOCATION               CHAR        NOT NULL,
 PRIMARY KEY(H_ID),
 UNIQUE(H_ID, AUTHOR, NAME, LOCATION)
)
CREATE TABLE Program_table*
(P_ID                   INT         NOT NULL,
 NAME                   CHAR,
 AUTHOR                 CHAR,
 SIZE                   INT,
 DATE_TIME              DATETIME,
 DESCRIPTION            CHAR,
 TYPE                   CHAR,
 LOCATION               CHAR        NOT NULL,
 PRIMARY KEY(P_ID),
 UNIQUE(P_ID, AUTHOR, LOCATION, TYPE)
)
```

```
CREATE TABLE Keyword_table
(K_ID                    INT      NOT NULL,
 KNAME                   CHAR,
 PRIMARY KEY(K_ID),
 UNIQUE(K_ID, KNAME)
)
CREATE TABLE Script_Resource_table
(SNAME                   CHAR     NOT NULL,
 R_ID                    INT      NOT NULL,
 PRIMARY KEY(SNAME, R_ID),
 FOREIGN KEY(SNAME)      REFERENCES Script_table(SNAME),
 FOREIGN KEY(R_ID) REFERENCES Resource_table(R_id)
)
CREATE TABLE Implementation_Resource_table
(STARTING_URL            CHAR     NOT NULL,
 R_ID                    INT      NOT NULL,
 PRIMARY KEY(STARTING_URL, R_ID),
 FOREIGN KEY(STARTING_URL) REFERENCES
    Implementation_table(STARTING_URL),
 FOREIGN KEY(R_ID) REFERENCES Resource_table(R_ID)
)
CREATE TABLE Implementation_Program_table
(STARTING_URL            CHAR     NOT NULL,
 P_ID                    INT      NOT NULL,
 PRIMARY KEY(STARTING_URL,P_ID),
 FOREIGN KEY(STARTING_URL) REFERENCES
    Implementation_table(STARTING_URL),
 FOREIGN KEY(P_ID)REFERENCES Program_table(P_ID)
)
CREATE TABLE Implementation_Html_table
(STARTING_URL            CHAR     NOT NULL,
 H_ID                    INT      NOT NULL,
 PRIMARY KEY(STARTING_URL,H_ID),
 FOREIGN KEY(STARTING_URL) REFERENCES
    Implementation_table(STARTING_URL),
 FOREIGN KEY(H_ID) REFERENCES Html_table(H_ID)
)
CREATE TABLE Web_Traversal_Message_table
(WEB_ID                  INT      NOT NULL,
 TNAME                   CHAR,
 PRIMARY KEY(WEB_ID),
 FOREIGN KEY(TNAME) REFERENCES
    Test_Record_table(TNAME),
 UNIQUE(WEB_ID)
)
```

```
CREATE TABLE QA_Engineer_table
(Q_ID                    INT      NOT NULL,
 NAME                    CHAR,
 TITLE                   CHAR,
 UNIT                    CHAR,
 PRIMARY KEY(Q_ID),
 UNIQUE(Q_ID,NAME)
)
CREATE TABLE B_Q_table
(BNAME                   CHAR     NOT NULL,
 Q_ID                    INT      NOT NULL,
 PRIMARY KEY(BNAME, Q_ID),
 FOREIGN KEY(BNAME) REFERENCES Bug_Report_table(BNAME),
 FOREIGN KEY(Q_ID) REFERENCES QA_Engineer_table(Q_ID)
)
CREATE TABLE B_Bad_table
(BNAME                   CHAR     NOT NULL,
 H_ID                    INT      NOT NULL,
 PRIMARY KEY(BNAME, H_ID),
 FOREIGN KEY(BNAME) REFERENCES Bug_Report_table(BNAME),
 FOREIGN KEY(H_ID) REFERENCES Html_table(H_ID)
)
CREATE TABLE B_Missing_table
(BNAME                   CHAR     NOT NULL,
 R_ID                    INT      NOT NULL,
 PRIMARY KEY(BNAME, R_ID),
 FOREIGN KEY(BNAME) REFERENCES Bug_Report_table(BNAME),
 FOREIGN KEY(R_ID) REFERENCES Resource_table(R_ID)
)
CREATE TABLE B_Redundant_table
(BNAME                   CHAR     NOT NULL,
 R_ID                    INT      NOT NULL,
 PRIMARY KEY(BNAME,R_ID),
 FOREIGN KEY(BNAME)  REFERENCES Bug_Report_table(BNAME),
 FOREIGN KEY(R_ID)   REFERENCES Resource_table(R_ID)
)
CREATE TABLE B_Inconsistence_table
(BNAME                   CHAR     NOT NULL,
 PRIMARY KEY (BNAME)
)
CREATE TABLE D_K_TABLE
(NAME                    CHAR     NOT NULL,
 K_ID                    INT,
 PRIMARY KEY(NAME, K_ID),
 FOREIGN KEY(K_ID)     REFERENCES Keyword_table(K_ID)
)
```

```
CREATE TABLE Script_Keyword_table
(SNAME                    CHAR   NOT NULL,
 K_ID                     INT    NOT NULL,
 PRIMARY KEY(SNAME, K_ID),
 FOREIGN KEY(SNAME) REFERENCES Script_table(SNAME),
 FOREIGN KEY(K_ID) REFERENCES Keyword_table(K_ID)
)
```

REFERENCES

1. Agosti, M., Melucci, M., and Crestani, F. Automatic authoring and construction of hypermedia for information retrieval. *Multimedia System* 3(3):15–24, 1995.
2. Allen, J. F. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11), 1983.
3. Anderson, D. P., and Homsy, G. A continuous media I/O server and its synchronization mechanism. *IEEE Computer* 51–57, 1991.
4. Arens, Y. Presentation design using an integrated knowledge base. In *Intelligent User Interfaces* (J. W. Sullivan and S. W. Tyler, Eds.), pp. 241–258. Assoc. Comput. Mach., New York, 1991.
5. Arens, Y. On the knowledge underlying multimedia presentations. In *Intelligent Multimedia Interfaces* (M. T. Maybury, Ed.), pp. 280–306. AAAI Press, Menlo Park, CA, 1993.
6. Backer, D. S. Multimedia presentation and authoring. In *Multimedia Systems* (J. F. K. Buford, Ed.), pp. 285–303. Assoc. Comput. Mach. New York, 1994.
7. Blakowski, G., Hubel, J., and Langrehr, U. Tools for specifying and executing synchronized multimedia presentations. In *Network and Operating System Support for Digital Audio and Video* (R. G. Herrtwich, Ed.), Second International Workshop Heidelberg, Germany, pp. 271–282, 1991.
8. Botafogo, R., and Mosse, D. The MORENA model for hypermedia authoring and browsing. In *Proceedings of the International Conference on Multimedia Computing and Systems,* Washington, DC, May 15–18, pp. 42–49, 1995.
9. Chen, C. R., Meliksetian, D. S., Chang, M. C.-S., and Liu, L. J. Design of a multimedia object-oriented DBMS. *Multimedia Systems.* 3:217–227, 1995.
10. Chih-Wen Cheng, *et al.* Networked hypermedia systems. In *Proceeding of the 1994 HD-MEDIA Technical and Applications Workshop,* October 6–8, Taipei, Taiwan, 1994.
11. Chin, D. N. Intelligent interfaces as agents. In *Intelligent User Interfaces* (J. W. Sullivan and S. W. Tyler, Eds.), pp. 177–206. Assoc. Comput. Mach., New York. 1991.
12. Chung, C. M., Shih, T. K., Huang, J.-Y., Wang, Y.-H., and Kuo, T.-F. An object-oriented approach and system for intelligent multimedia presentation designs. In *Proceedings of the ICMCS'95 Conference,* pp. 278–281, 1995.
13. Gibbs, S., Dami, L., and Tsichritzis, D. An object-oriented framework for multimedia composition and synchronization. In *Eurographic Seminars, Tutorials and Perspectives in Computer Graphics Multimedia Systems, Interaction and Applications,* (L. Kjelldahl Ed.), Chap. 8, 1991.
14. Halang, W. A., and Wannemacher, M. High-precision temporal synchronization in distributed multimedia systems. In *Proceedings of the Second ISATED/ISMM International Conference on Distributed Multimedia Systems and Applications,* Stanford, CA, August 7–9, pp. 221–223, 1995.
15. Chen, H.-Y. *et al.* A novel audio/video synchronization model and its application in multimedia authoring system. In *Proceedings of the 1994 HD-MEDIA Technical and Applications Workshop,* 1994.
16. Hoepner, P. Presentation scheduling of multimedia objects and its impact. In *Network and Operating System Support for Digital Audio and Video Second International Workshop Heidelberg,* Germany (R. G. Herrtwich Ed.), pp. 132–143, 1991.
17. Hoepner, P. Synchronizing the presentation of multimedia objects—ODA extensions. In *ACM SIGOIS,* pp. 19–31, 1991.
18. Hoepner, P. Synchronizing the presentation of multimedia objects. *Comput. Commun.* 15(9): 557–564, 1992.

19. Schnepf, J. *et al.* Doing FLIPS: Flexible interactive presentation synchronization. In *Proceedings of the International Conference on Multimedia Computing and Systems,* pp. 213–222, 1995.
20. J. E. C. JR. and Roussopoulos, N. Timing requirements for time-driven systems using augmented Petri nets. *IEEE Trans. Software Engrg.* 9(5):603–616, 1983.
21. Kunii, T., Shinagawa, Y., Paul, R., Khan, M., and Khokhar, A. A. Issues in storage and retrieval of multimedia data. *Multimedia Systems* 3:298–304, 1995.
22. Leydekkers, P., and Teunissen, B. Synchronization of multimedia data streams in open distributed environments. In *Network and Operating System Support for Digital Audio and Video, Second International Workshop Heidelberg, Germany,* (R. G. Herrtwich, Ed.), pp. 94–104, 1991.
23. Little, T. D., and Ghafoor, A. Multimedia synchronization protocols for broadband integrated services. *IEEE J. Selected Areas Commun.* 9(9):1368–1382, 1991.
24. Little, T. D. C., and Ghafoor, A. G. Spatio-temporal composition of distributed multimedia objects for value-added networks. *IEEE Computer.* 42–50, 1991.
25. Little, T. D. C., and Ghafoor, A. Synchronization and storage models for multimedia objects. *IEEE J. Selected Areas Commun.* 8(3):413–427, 1990.
26. Lundeberg, A., Yamamoto, T., and Usuki, T. SAL, A hypermedia prototype system. In *Eurographic Seminars, Tutorials and Perspectives in Computer Graphics, Multimedia Systems, Interaction and Applications* (L. Kjelldahl, Ed.), Chapter 10, 1991.
27. Oomoto, E., and Tanaka, K. OVID: Desibn and implementation of a video-object database system. *IEEE Trans. Knowledge Data Engrg.* 5(4):629–643, 1993.
28. Ouyang, Y. C., and Lin, H.-P. A multimedia information indexing and retrieval method. In *Proceedings of the Second ISATED/ISMM International Conference on Distributed Multimedia Systems and Applications,* Stanford, CA, August 7–9, pp. 55–57, 1995.
29. Ozsu, M. T., Szafron, D., El-Medani, G., and Vittal, C. An object-oriented multimedia database system for a news-on-demand application. *Multimedia Systems,* 3:182–203, 1995.
30. Paul, R., Khan, M. F., Khokhar, A., and Ghafoor, A. Issues in database management of multimedia information. In *Proceedings of the 18th IEEE Annual International Computer Software and Application Conference (COMPSAC'94),* Taipei, Taiwan, pp. 209–214, 1994.
31. Prabhakaran, B., and Raghavan, S. V. *Synchronization Models for Multimedia Presentation with User Participation,* Vol. 2 of Multimedia Systems. Springer-Verlag, Berlin, 1994.
32. Dannenberg, R. B. *et al.* A computer based multimedia tutor for beginning piano students. *Interface* 19(2–3):155–173, 1990.
33. Rhiner, M., and Stucki, P. Database requirements for multimedia applications. In *Multimedia System, Interaction and Applications* (L. Kjelldahl Ed.), pp. 269–281, 1991.
34. Rosenberg, J., Cruz, G., and Judd, T. Presenting multimedia documents over a digital network. In *Network and Operating System Support for Digital Audio and Video, Second International Workshop Heidelberg, Germany,* (R. G. Herrtwich Ed.), pp. 346–356, 1991.
35. Schurmann, G., and Holzmann-Kaiser, U. Distributed multimedia information handling and processing. *IEEE Network,* November: 23–31, 1990.
36. Shih, T. K. An Artificial intelligent approach to multimedia authoring. In *Proceedings of the Second IASTED/ISMM International Conference on Distributed Multimedia Systems and Applications,* pp. 71–74, 1995.
37. Shih, T. K. On making a better interactive multimedia presentation. In *Proceedings of the International Conference on Multimedia Modeling,* 1995.
38. Shih, T. K., and Chang, A. Y. Toward a generic spatial/temporal computation model for multimedia presentations. In *Proceedings of the IEEE ICMCS Conference,* 1997.
39. Shih, T. K., and Davis, R. E. IMMPS: A multimedia presentation design system. *IEEE Multimedia* Fall, 1997.
40. Shih, T. K., Kuo, C.-H., and An, K.-S. Multimedia presentation designs with database support. In *Proceedings of the NCS'95 Conference,* 1995.
41. Shih, T. K., Kuo, C.-H., and An, K.-S. An object-oriented database for intelligent multimedia presentations. In *Proceedings of the IEEE International Conference on System, Man, and Cybernetics Information, Intelligence and Systems Conference,* 1996.
42. Shivakumar, N., and Sreenan, C. J. The concord algorithm for synchronization of networked multimedia streams. In *Proceedings of the International Conference on Multimedia Computing and Systems,* Washington, DC, May 15–18, pp. 31–40, 1995.

43. Smoliar, S. W., and Zhang, H. J. Content-based video indexing and retrieval. *IEEE MultiMedia,* 62–72, 1994.
44. Staehli, R., Walpole, J., and Maier, D. A quality-of-service specification for multimedia presentations. *Multimedia Systems,* 3: 251–263, 1995.
45. Steinmetz, R. Synchronization properties in multimedia systems. *IEEE J. Selected Areas Commun.* 8(3):401–412, 1990.
46. Woodruff, G. M., and Kositpaiboon, R. Multimedia traffic management principles for guaranteed ATM network performance. *IEEE J. Selected Areas Commun.* 8(3):437–446, 1990.
47. Al-Salqan, Y. Y. *et al. MediaWare: On Multimedia Synchronization,* pp. 150–157, 1995.
48. Yavatkar, R. MCP: A protocol for coordination and temporal synchronization in multimedia collaborative applications. In *IEEE 52th Intl Conference on Distributed Computing Systems,* June 9–12, pp. 606–613, 1992.
49. Yoshitaka, A., Kishida, S., Hirakawa, M., and Ichikawa, T. Knowledge-assisted content-based retrieval for multimedia database. *IEEE Multimedia Magazine* 12–21, 1994.
50. Tracz, W. Software reuse myths. *ACM SIGSOFT Software Engineering Notes* **13**(1):17–21, 1988.
51. Kaiser, G. E. *et al.* Melding software systems for reusable building Blocks. *IEEE Software* 17–24, 1987.
52. Lenz, M. *et al.* Software reuse through building blocks. *IEEE Software* 34–42, 1987.
53. Gargaro, A. *et al.* Reusability issues and Ada. *IEEE Software* 43–51, 1987.
54. Prieto-Diaz, R. *et al.* Classifying software for reusability. *IEEE Software* 6–16, 1987.
55. Prieto-Diaz, R. Implementing faceted classification for software reuse. *Commun. ACM* **34**(5): 88–97, 1991.
56. Ghezala, H. H. B. *et al.* A reuse approach based on object orientation: Its contributions in the development of CASE tools. In *Proceedings of the SSR'95 Conference,* Seattle, WA, pp. 53–62.
57. Bieman, J. M. *et al.* Reuse through inheritance: A quantitative study of C^{++} software. In *Proceedings of the SSR'95 Conference,* Seattle, WA, pp. 47–52.
58. Bieman, J. M. *et al.* Cohesion and reuse in an object-oriented system. In *Proceedings of the SSR'95 Conference,* Seattle, WA, pp. 259–262.
59. Burton, B. A. *et al.* The reusable software library. *IEEE Software* 25–33, 1987.
60. Fischer, G. Cognitive view of reuse and redesign. *IEEE Software* 60–72, 1987.
61. Tyugu, E. Three new-generation software environments. *Commun. ACM* **34**(6):46–59, 1991.
62. Tyugu, E. et al. NUT—An object-oriented language. *Comput. Artif. Intell.* 5(6):521–542, 1986.

11 DATA STRUCTURE IN RAPID PROTOTYPING AND MANUFACTURING

CHUA CHEE KAI, JACOB GAN, AND DU ZHAOHUI

School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

TONG MEI

Gintic Institute of Manufacturing Technology, Singapore 638075

Database and Data Communication Network Systems, Vol. 2

I. INTRODUCTION

A. Data Structure Technique

The efficient management of geometric information, such as points, curves, or polyhedrons is of significant importance in many engineering applications, such as computer-aided design (CAD), computer-aided manufacturing (CAM), robotics, and rapid prototyping and manufacturing (RP&M). In addition to representing the objects correctly and sufficiently, a good representation scheme maps the original data objects into a set of objects that facilitate efficient storage and computation.

In geometric computing encountered in engineering applications, it is often necessary to store multiple representations of the same data in order to facilitate efficient computation of a great variety of operators. Moreover, the same data may be utilized by categories of users and across heterogeneous systems during different phases of the product design and manufacturing process; thus more than one representation may be necessary. Multiple representations incur a significant overhead to ensure availability and consistency of the data.

A data structure is the form of organization imposed on the collection of those data elements. It is defined by specifying what kind of elements it contains and stating the rules of how to store the elements and how to retrieve them when needed. Also, data structures are the materials from which computer programs are built, just as physical materials are built from molecules of their component substances. In engineering computing, they reduce to only a few simple entities, mainly numbers and characters. Correspondingly, any kind of representation should be realized with numbers and characters.

Data structures may be classified into linear and nonlinear types [1]. Linear structures are those elements that have a sequential relationship. For example, a list of houses along a street, is a linear structure: collections of likewise elements with a clearly defined ordering. Such structures are often represented in diagrams by collections of boxes with lines to show their relationship. Linear structures occupy a special place in the study of data structures because the addressing of storage locations in a computer is nearly always linear, so the set of memory storage locations in the machine itself constitutes a linear structure. Linear structures may be further classified as addressable or sequential. Arrays are important types of addressable structure: a specific element can be retrieved knowing only its address, without reference to the others. In a sequential data structure an element can only be reached by first accessing its predecessor in sequential order. Because much engineering software is devoted to mathematically oriented tasks that involve solving simultaneous equations, arrays and metrics play a large role in its design. Nonlinear data structures are of varied sorts. One category important to the software designer is that of hierarchical structures, in which each element is itself a data structure. One key feature distinguishes a hierarchy: there is only a single unique path connecting any one element to another. Given that computer memory devices are arranged as linear strings of storage locations, there is no "natural" way of placing nonlinear structures in memory.

That more than one data structure type exist means there is no single data structure type that can be suitable for all applications. Data structure is often

selectively designed for the efficiency of storing, retrieving, and computing operators. A good data structure is vital to the reliability and efficiency of a program or software.

RP&M is a set of manufacturing processes that can fabricate complex freeform solid objects with various materials directly from a CAD file of an object without part-specific tooling. Despite the fast development and worldwide installation of 3289 systems in 1998 [2], there is no suitable single information standard. The RP&M information processing involves transforming the CAD file into a special 3D facet representation, slicing the geometric form of the part into layers, generating the contours of the part for each layer, and hatching the contours of each layer.

B. Rapid Prototyping and Manufacturing Technology

For better appreciation for the need of good data structure techniques to be used in RP&M, a discussion on the various RP&M processes is in order. RP&M entails the fabrication of an object from its CAD by selectively solidifying or bonding one or more raw materials into a layer, representing a slice of the desired part, and then fusing the successive layers into a 3D solid object. The starting material may be liquid, powder, or solid sheets, while the solidification process may be polymerization, sintering, chemical reaction, plasma spraying, or gluing.

RP&M technology has the potential of ensuring that quality-assured prototypes or parts are developed quickly for two major reasons. There are almost no restrictions on geometrical shapes, and the layered manufacturing allows a direct interface with CAD to CAM, which almost eliminates the need for process planning. These advantages of RP&M technology bring the results with enhancing and improving the product development process and, at the same time, reducing the costs and time required for taking the product from conception to market. The technology has already shown potential and valuable in ever-increasing application fields, including the manufacturing/tooling industry, automobile industry, architecture, and biomedical engineering.

1. RP&M Principle and Processes

Over the past few years, a variety of RP&M techniques has been developed. RP&M processes are classified into variable energy processes and variable mass processes [3]. Variable energy processes are those where a uniform mass is selectively activated, removed, or bonded by a variable energy controlled by the layer description. These processes include molecule bonding, particle bonding, and sheet lamination. Variable mass processes are those where a constant source of energy is used to fuse or solidify a variable mass controlled by the layer description. These processes are the droplet deposition process, the particle deposition process, and the melt deposition process.

Liquid Solidification Processes

The photopolymerization processes are those where the optical energy like laser or UV light driven by layer information solidifies the scanned areas of thin photopolymer resin layers.

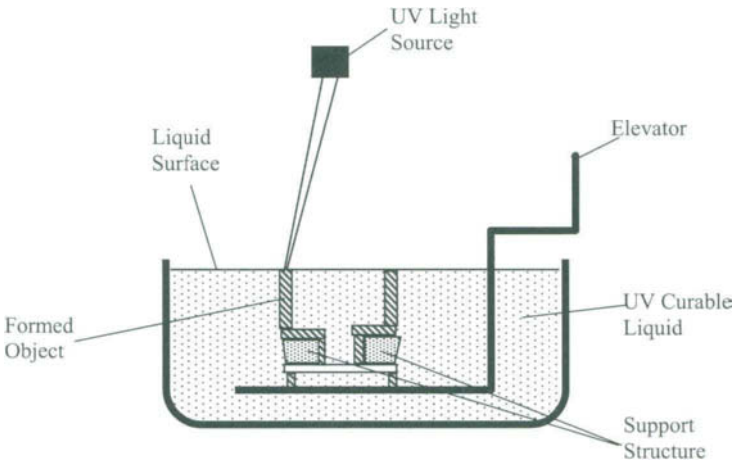


FIGURE 1 SLA process.

1. SLA—Stereolithography Apparatus. The SLA process was developed and commercialized by 3D Systems [4,5]. Its product models include SLA-190, SLA-250, and SLA-500. The process of the SLA system is a kind of liquid solidification process as shown in Fig. 1 [6].

First, a three-dimensional CAD solid model file is loaded into the system. If needed, supports are designed to stabilize the part during building and post-curing. Then the control unit slices the model and support into a series of cross sections from 0.004 to 0.020 in. thick.

After that, the cross-sectional information is passed onto SLA machines. The process of SLA begins with the vat being filled with the photopolymer liquid and the elevator table set just below the surface of the liquid. Then a computer-controlled UV light source scans the photopolymer liquid that is partially cured to form one layer. After a layer is formed, the elevator is lowered to allow uncured liquid to flow over the top of the part, in preparation for scanning the next layer's features. This process continues building the part from bottom to top, until the system completes the product. Finally, UV post-curing following scanning of the entire part in an oven completes the production cycle.

2. SGC—Solid Ground Curing. The SGC process was developed by the Cubital Ltd [7]. Cubital's products are the Solider 4600 and Solider 5600. SGC is the other example of photopolymerization but is different from SLA in that it uses a lamp instead of a point-by-point laser. The Cubital's Solid Ground Curing process includes three main steps: data preparing, mask generating, and model making.

In the data preparing step, the job for production is prepared and the cross sections in imaging format are transferred to a mask generator. After data are received, a mask plate is charged through an "image-wise" ionographic process. Then the charged image is developed with electrostatic toner. Then model building begins with spreading a thin layer of photopolymer of any viscosity. Secondly, the photomask from the mask generator is placed above the workpiece, in close proximity, and both are aligned under a collimated UV lamp.

Thirdly, the UV light is turned on for a few seconds. Part of the resin layer is hardened according to the photomask. Then the unsolidified resin is collectedly sunk out from the workpiece. After that, melted wax is spread into the cavities created after collecting the uncured liquid resin. Consequently, the wax in the cavities is cooled to produce a wholly solid layer. Finally, the layer is milled to its exact thickness, producing a flat solid surface ready to receive the next layer.

Particle Bonding Processes

Particle bonding process uses the modulated energy to selectively bond particles in a thin layer of powder material. Those two-dimensional layers are stacked together to form a complex three-dimensional solid object. The commercial machines use particle-bonding technology in very different designs. The typical systems using particle-bonding technology are introduced as follows.

1. 3DP—Three-Dimensional Printing. Three-dimensional printing creates parts by a layered printing process, based on the sliced cross-sectional information [8,9]. One layer is created by deposition of a layer of powder. The powder layer is then selectively joined where the part is to be formed by "ink-jet" printing of a binder material. This process is repeated layer by layer until the part is created.

Three-dimensional printing works by building parts in layers (see Fig. 2). First, a thin distribution of powder is spread over the surface of a powder bed. From a computer model of the desired part, a slicing algorithm computes information for the layer. Using a technology similar to ink-jet printing, a binder material joins particles where the object is to be formed. A piston then lowers so that the next powder layer can be spread and selectively joined. This layer-by-layer process repeats until the part is completed. Following a heat treatment, unbound powder is removed, leaving the fabricated part.

2. SLS—Selective Laser Sintering. The SLS process creates three-dimensional objects, layer by layer, from powdered materials with heat generated by a CO_2 laser [10]. This process was developed by DTM Corporation, USA. DTM's products are the Sinterstation 2000, which is capable of producing objects measuring 12 in. in diameter by 15 in. in height, and Sinterstation 2500 whose build chamber is rectangular, 15 in. in width, 13 in. in depth, and 18 in. in height, to accommodate larger parts.

The SLS process is described in Fig. 3. The three-dimensional CAD data must be in the de facto industry standard, the STL format. As the SLS process begins, a thin layer of the heat-fusible powder is deposited onto the part-building cylinder within a process chamber. An initial cross section of the object under fabrication is selectively "drawn" on the layer of powder by a heat-generating CO_2 laser. The interaction of the laser beam with the powder elevates the temperature to the point of melting, fusing the powder particles and forming a solid mass. The intensity of the laser beam is modulated to melt the powder only in the areas defined by the object's design geometry.

An additional layer of powder is deposited via a roller mechanism on the top of the previously scanned layer. The process is repeated, with each layer fusing to the layer below it. Successive layers of powder are deposited and the

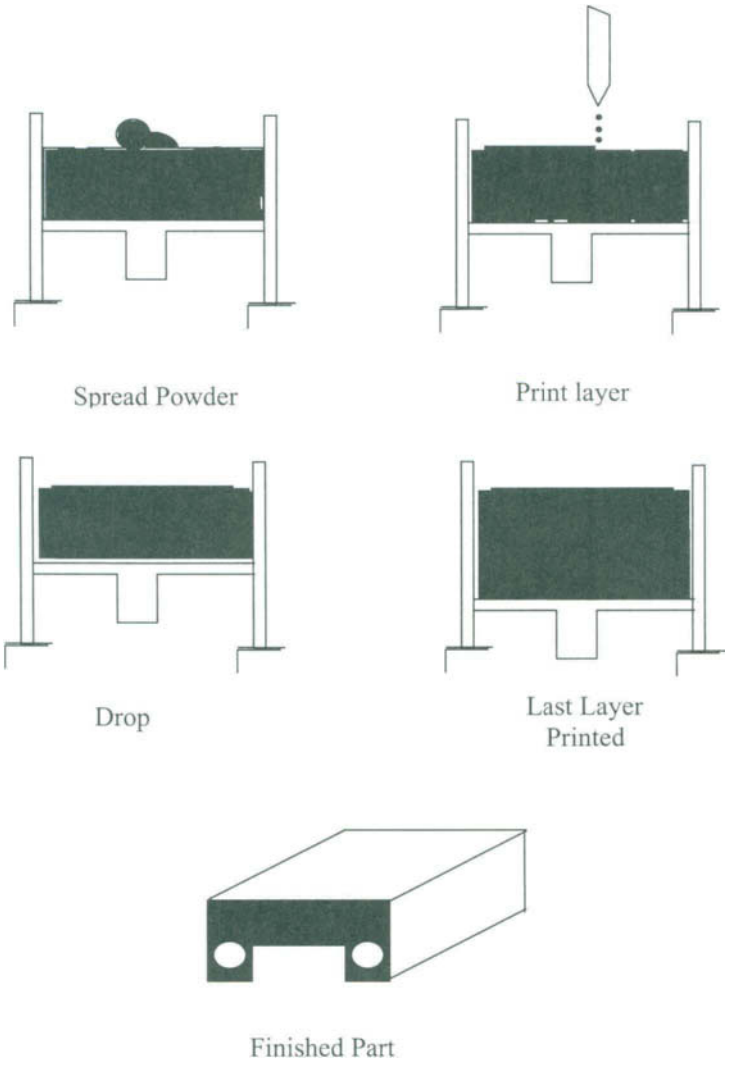


FIGURE 2 Three-dimensional printing process.

process is repeated until the part is completed. Finally, the part is removed from the build chamber, and the loose powder falls away. SLS parts may then require some post-processing, such as sanding, depending upon the application.

Sheet Lamination Processes

Sheet lamination processes are those by which layer information in the form of electronic signals is used to cut thin layers of sheet material to individual two-dimensional cross-sectional shapes. Those two-dimensional layers are bonded one upon another to form a complete three-dimensional object.

A representative sheet lamination process is the laminated object manufacturing (LOM) process, which was first developed by Helisys Inc, USA. The LOM process is an automated fabrication method in which a three-dimensional object is constructed from a solid CAD representation by sequentially

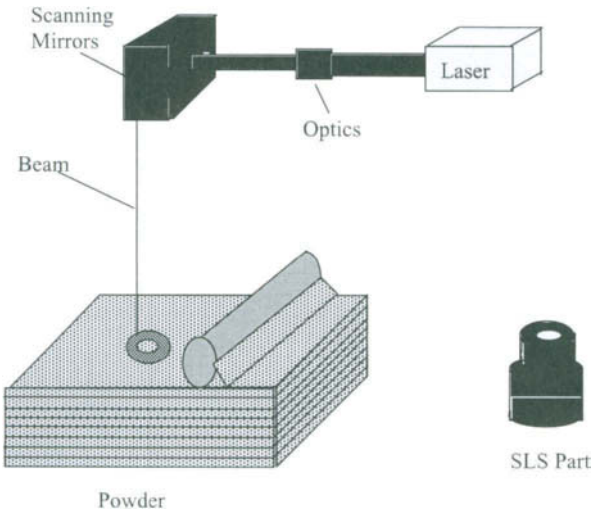


FIGURE 3 Selective laser sintering (SLS) process (with permission from Springer-Verlag).

laminating the constituent cross sections [11]. The process consists of three essential phases: preprocessing, building, and postprocessing.

The preprocessing phase encompasses several operations. The initial steps include generating an image from a CAD-derived .STL file format of the part to be manufactured, sorting input, and creating secondary data structures.

During the building phase, thin layers of adhesive-coated material are sequentially bonded to each other and individually cut by a CO_2 laser beam (see Fig. 4). The building cycle includes the following steps. Firstly, a cross section

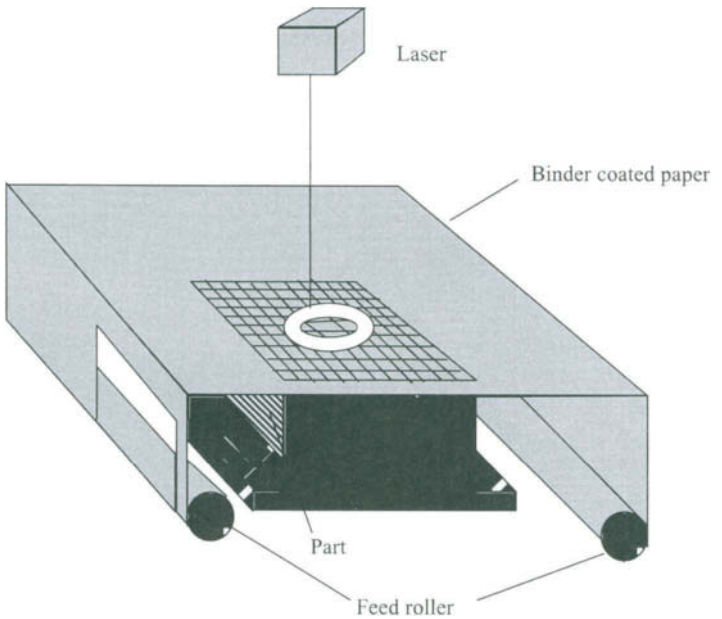


FIGURE 4 Sheet lamination manufacturing system (with permission from Springer-Verlag).

of the 3D model is created through measuring the exact height of the model and slicing the horizontal plane accordingly. Secondly, the computer generates precise calculations that guide the focused laser beam to cut the cross-sectional outline, the crosshatches, and the model's perimeter. The laser beam power is designed to cut exactly the thickness of one layer of material at a time. After that, the platform with the stack of previously formed layers descends and a new section of the material is advanced. The platform ascends and the heated roller laminates the material to the stack with a single reciprocal motion, thereby bonding it to the previous layer. Finally, the vertical encoder measures the height of the stack and replays the new height to the slicing software, which calculates the cross section for the next layer as the laser cuts the model's current layer. These procedures repeat until all layers are built.

In the final phase of the LOM process, postprocessing, the part is separated from support material. After that, if necessary, the finishing process may be performed.

Droplet Deposition Processes

Droplet deposition is a variable mass process where layer information in the form of electronic signals is used to modulate the deposition of liquid droplets in a thin layer. Successive layer formation, one atop another, forms a complex three-dimensional solid object as illustrated in Fig. 5.

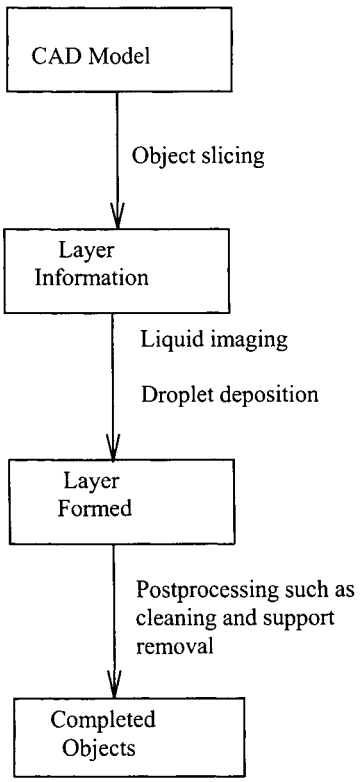


FIGURE 5 Droplet deposition process (with permission from Springer-Verlag).

The ballistic particle manufacturing (BPM) process is a typical droplet deposition process [12]. The BPM process uses three-dimensional solid model data to direct streams of material at a target. The three-dimensional objects are generated in a manner similar to that of an ink-jet printer producing two-dimensional images.

Melt Deposition Processes

Melt deposition is a variable mass process where layer information in the form of electronic signals is used to modulate the deposition of molten material in a thin layer. Successive layer formation, one atop another, forms a three-dimensional solid object.

The fused deposition modeling (FDM) process is a melted deposition fabrication process that uses the Stratasys so-called 3-D Moduler in conjunction with a CAD workstation [13,14]. The 3-D Moduler operates on the principle of a three-axis NC-machine tool. A nozzle is controlled by a computer along three axes. The nozzle guides the specific material, which is melted by heating. So the material leaves the nozzle in a liquid form, which hardens immediately at the temperature of the environment. This is realized through precise control of the temperature in the building cabinet. FDM builds the model upwards from a fixtureless base.

2. RP&M Information Processing

Rapid prototyping techniques, whether using lasers on photopolymers, particle bonding, or sheet lamination, share the common ground of quickly constructing accurate models or prototypes directly from a CAD description, under computer control. From the process principle of various RP&M techniques introduced above, it is obvious that producing parts with them requires a transformation from the geometrical description of parts into a form that can be receivable and manipulated by the particular RP&M technique.

The procedure, sometimes called information processing for RP&M, can be regarded as a generation of the computer numerical control (CNC) scanning vector from CAD models. It shows the information processing to be somewhat a reversal of the physical procedure of building the model by means of stacking a series of 2D layers together. It has been proved that the interface transferring data from the CAD to RP&M system is a very important factor because it affects the quality of the finished part and the cycle time.

The whole information processing (shown in Fig. 6) includes various manipulations with multiple types of input. One particular manipulation brings the requirements of specialized data representation, data storage, and data structure for the purpose of enhancing the reliability, improving the efficiency of algorithms, and reducing the times to access the data. Thus, various interfaces and data formats are introduced among the different phases in the procedure. However, some of the formats proposed are imperfect due to the lack of sufficient information supplied, the inclusion of errors due to the lack of data validation, the increase of the file size due to redundant information, or the difficulty of the manipulations followed. Even though these formats and data structures include internal flaws, they have been adopted so popularly since their creation that they have become de facto standards in industry. Some of those proposed

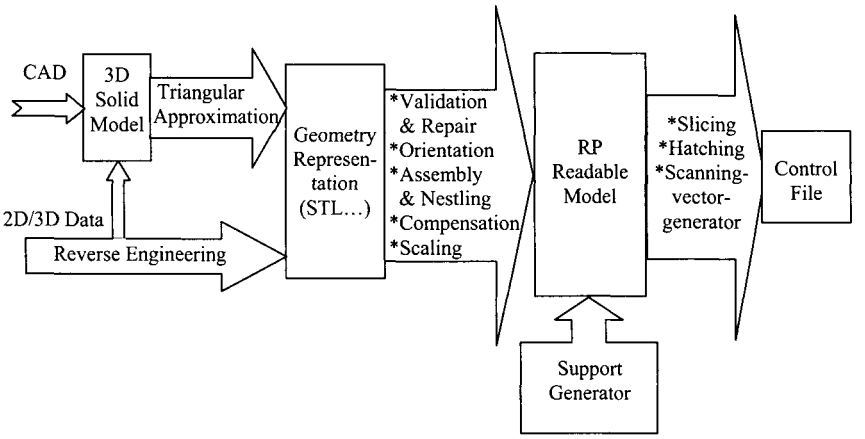


FIGURE 6 P&M information processing diagram (with permission from Springer-Verlag).

lately or to be created in the future will prove their potential as long as they overcome the existing shortcomings. In this chapter, the involved manipulations and related data structures and formats will be reviewed and discussed.

II. INTERFACES BETWEEN CAD AND RP&M

Fabrication of a physical model by RP&M requires a mathematical description of the part's geometry, which must be a solid or surface model stored in the computer. Many methods for the generation of geometric models have been developed. This kind of model can be designed and drafted by means of surface or solid modeling via commercial CAD systems which provide the application tools and commands. In other cases, although the parts already exist like human organs, limbs, or architectures, it seems impossible for designers to make a correct and detailed description in a simple way. However, nowadays various metrologies, such as CMM (Coordinate Measurement Machine), Laser Scanner, or other optical scanners, can capture the surface of the part with a cloud point set. Each point is represented by a three-dimensional coordinate (X, Y, Z). Most of the scanners like these have difficulties in measuring the inner surface of parts if the probe of the CMM or light, even including a laser, cannot touch the inside cavity because of the obstacles of the outer section of parts. In the medical imaging area, technologies including CT, MRI, PET/SPETC, X-ray, and Ultrasound are candidates for scanning three-dimensional shape data of internal and external human body structures with a series of 2-D image section planes. In order for the imaging data to be used in RP&M, they need to be converted to a 3-D geometric representation of the structures.

A. CAD Modeling

The prime purpose of a CAD system is that it should enable a designer to use the computer in his design work. While drafting-centered CAD systems could

satisfy the limited requirements involved in producing engineering drawings in their first generation, it is clear that a system that allows the design to be generated in three dimensions would be more advantagious in many cases. With two-dimensional display and input devices, many attempts have been made to provide 3-D images. Since the true three-dimensional modeling first appeared in the form of a wire frame, several steps and modeling methods have been put forward. Many industrial users are now beginning to encourage the direct use of full 3-D modeling in design to replace 2-D drawings, and major benefits from complete 3-D modeling have been achieved already in many cases. However, RP&M technology requires a 3-D model of the part before building it, which makes it significant to discuss the modeling methods in CAD technology.

1. Wire-Frame Modeling

As the first tryout, the wire-frame modeling method uses lines and curves in space to represent the edges of surfaces and planes for providing a 3-D illusion of the object. When the object is displayed on a computer monitor, it appears as if wires had been strung along the edges of the object, and they are those edges being seen. So in a wire-frame modeler, entities such as lines and curves (mostly arcs) are used to connect the nodes. It is obvious that the data structure and database are relatively simple with a small amount of basic elements.

However, a wire-frame model does not have all the information about a designed object. The most obvious drawback is that such a representation includes all the "wires" necessary to construct the edge lines of every surface and feature, even if those are normally hidden from view. As a result the image is often confusing and difficult to interpret. Although it works in a limited extent, the removal of the hidden-lines involves making some assumptions regarding the nature of the nonexistent surfaces between the frames. If a wire-frame model is used to build a part by RP&M, the systems would be confused as to which side of the part should be solidified, sintered, or deposited because the frame shows no direction. Furthermore, slicing manipulation in RP&M, which cuts the 3-D model into a series of 2-D contours, would meet a large difficulty when neither plane nor surface exists in wire-frame model to intersect with the cross sections. What would be received would only a set of points instead of contours.

Partly due to the difficulties of interpreting wire-frame images and partly due to the lack of a full description of an object's information, this method has not been well utilized by industry. Therefore, a wire-frame modeler is declared not to be suitable for any commercialized RP&M systems. This brings us to the concept of surface modeling and to the next module.

2. Surface Modeling

The addition of surface patches to form the faces between the edges of a wire-frame model is an attempt to put more information about the designed object into the computer model. CAD systems that can include surfaces within the description of an object in this way are called surface modelers. Most of the surface modeling systems support surface patches generated from standard shapes such as planar facets, spheres, and circular cylinders, which is not enough to represent arbitrary shapes and thus often the case with typical parameteric

surface forms such as Bezier surfaces, B-spline surfaces, and nonuniform B-spline (NURBS) surfaces.

In order to store information about the inserted surfaces, some CAD systems make use of a face list in addition to the node and entity lists. Each face list might consist of the sequence of the numbers of the entities that bound the face. The data structure is such that entities are defined as sets of nodes and faces are defined as sets of entities. For many applications, such as the aircraft industry and the design of car body shells, the ability to handle complex surface geometry, which is often called a sculptured surface for function or aesthetic reasons, is demanded. Surface modelers have sufficient sophistication in their surface types to allow these requirements to be satisfied while solid modelers are often restrictive and not needed.

Although surface modeling is simple in data organization and highly efficient, it still holds partial information about the designed object. The biggest question concerns the lack of explicit topological information, which makes certain validity checks with surface models difficult. The drawback frequently happens while using STL format as an interface between CAD and RP&M, in which the STL file can be regarded as a surface model with triangular facets to represent an object. Topological information is not provided in surface modeling to indicate how the connection is among the primitives that make up the object. Topological information may also be serviced to check the validation of data in objects. Normally, it is unnecessary for a surface model to be constrained as a close space. For RP&M application this will bring confusions as to which side is filled with material and which side is empty. Moreover, a model with open surfaces cannot be processed in RP&M. A solid modeling system is intended to hold complete information about the object being designed for that purpose.

3. Solid Modeling

Three-dimensional solid modeling is a relatively new concept for design and manufacturing, only developed in the early 1980s to overcome the limitations of early wire-frame and surface modeling CAD systems. As the name suggests, solid models provide the message as to which sections of the design space contain solid material and which sections are empty. An object is constructed from the combination of simple solid primitives with rigorously defined set operations such as union, difference, intersection, and so on. With more information including topology available in the system, it becomes possible to conduct more analysis on the design, like validation of data.

Emerging as dominant representations in commercial systems, solid modelers can usually be classified into two types: constructive solid geometry (CSG) and boundary representation (B-rep). In CSG systems, the object is built up from standard, predefined, and parameterized "primitives." Usually the primitives include truncated cones, rectangle blocks, spheres, toruses, and so on. With CSG the geometry of an object is constructed as a binary tree whose leaf nodes are oriented instances of primitives and whose nonterminal nodes are the set operations. With this type of data organization, a designer can find it a "natural" way of describing engineering parts because the brain tends to break down any complex object into simpler components and then combine them together. The user of a purely CSG system is naturally limited by the repertoire of

standard primitives that is available. Many of the commercial systems provide the primitive set, consisting of a quadric surface (e.g., sphere, cylinder, cone), torus, and surfaces generated by revolving or sweeping 2D profiles. All the primitives are in closed form and can be computed with Boolean operations.

Even though the CSG modeling method maintains a large advantage in that it provides a very compact way of containing the complete information about a design, it is an implicit representation. For the slicing process in RP&M, the CSG model must be converted into a boundary representation of the solid. The same issues occur in many other applications such as finite element mesh generation.

The other usually identified approach to solid modeling is boundary representation. In B-rep, the component is stored as a collection of entities forming the boundaries of the faces, together with information about the surface patches defining those faces. In B-rep, all the faces, edges, and vertices of the faces are nodes that are connected together with arcs. The topology information of objects is explicit with the adjacency relationships. Additionally, the normal of each face in a B-rep model usually points outward of the object. The largest difference between B-rep and surface modeling is that the faces in B-rep are held in such a systematic way that the system can tell which side is within the object and which is without. Compared to a CSG structure, most B-rep modeling systems have the benefit that the information is always readily available, especially when the sequential computation is concerned.

B. Interfaces between CAD Systems

As CAD technology has evolved over more than 20 years, a lot of issues including modeling method, data technique, software/hardware upgrading, system development, and extended applications have become important. The attractive market has brought about ever-increasing commercialized systems. Normally the data structure and database are designed individually by the various system developers, which becomes an obstacle for communications between these systems. Even if the information has been generated in one of the CAD systems, it is not available for other applications as a result of inaccessible data. This urges a need for efficient storage, retrieval, and exchange of information. Many different interfaces have been developed for various systems. In order to be reliable and accepted by a wide community of both vendors and users, a standard should be stable but flexible enough to accommodate present as well as expected future developments.

1. Initial Graphics Exchange Specification (IGES)

Initial Graphics Exchange Specification (IGES) is a standard used to exchange graphics information between commercial CAD systems [15,16]. It was set up as the American National Standard in 1981. The IGES file can precisely represent CAD models. It includes four sections: Start Section, Global Section, Directory Entry Section, and Parameter Data Section.

The Start Section is designed to provide a human-readable prologue to the file. The Global Section of the file contains information describing the preprocessor and information needed by the postprocessor to handle the file. The

Directory Entry Section provides an index and contains attribute information and topological information about each entity. It also has one directory entry for each entity in the file. The Parameter Data Section of IGES contains geometrical parameter data associated with each entity.

In IGES, both constructive solid geometry and boundary representation are catered for representing models. Especially, the ways of representing the regularized operations for union, intersection, and difference have also been defined. IGES is a generally used data transfer medium between various CAD systems. It can precisely represent a CAD model. The advantages of using IGES, which include precise geometry representation, few data conversions, smaller data files, and simpler control strategies, over current approximation methods, may make it survive for a long term.

The advantages of the IGES standard are the following:

1. Since IGES was set up as the American National Standard, virtually every commercial CAD/CAM system has adopted IGES implementation.
2. It provides a wide range of entities of points, lines, arcs, splines, NURBS surfaces, and solid elements. Therefore, it can precisely represent a CAD model.

The disadvantages of the IGES standard are listed as follows:

1. Because IGES is the standard format to exchange data between CAD systems, it also includes much redundant information not needed for RP&M, such as electrical and electronic application information and architecture and construction information.
2. The algorithm for slicing an object in the IGES format is more complex than that in the STL format.
3. The support structures occasionally needed may not be easily created according to the IGES format.

2. Hewlett-Packard Graphics Language (HP/GL)

Hewlett-Packard Graphics Language (HP/GL) is a standard data format for graphic plotters [5,17]. Data types are all two-dimensional, including lines, circles, splines, and texts. The approach, as seen from a designer's point of view, would be to automate a slicing routine that generates a section slice, invokes the plotter routine to produce a plotter output file, and then loops back to repeat the process.

The advantages of the HP/GL format are listed as follows:

1. Many commercial CAD systems have the interfaces to output HP/GL formats.
2. It is a 2D geometric data format that can be directly passed to rapid prototyping and manufacturing systems without slicing.

The disadvantages of the HP/GL format include the following:

1. Because HP/GL is a 2D data format and the files would not be appended, hundreds of small files are needed to describe a model.

2. All the support structures required are generated in the CAD system and sliced in the same way before the CAD model is converted into HP/GL files.

3. Standard for the Exchange of Product Data Model (STEP)

The standard for the exchange of product data model, STEP, is a new engineering product data exchange standard that is documented as ISO 10303 [18]. The aim of STEP is to produce a single and better standard to cover all aspects of the product life cycle in all industries. European Action on Rapid Prototyping (EARP) is working on using STEP as a tool for data transfer from CAD to 3D layer manufacturing systems [19].

The reasons why STEP is recommended to be used as the interface between CAD and RP&M are given as follows:

1. It will be an international standard format to exchange product data and be supported by all CAD systems.
2. Since it is a complete representation of products for data exchange, the information in STEP is enough for the data exchange from CAD to 3D layer manufacturing systems.
3. It is efficient in both the file size and computer resources needed for processing.
4. It is independent of hardware and software.

However, STEP still has some disadvantages as the interface between CAD and RP&M.

1. It still carries much redundancy information that may be not necessary to RP&M.
2. New interpreters and algorithms must be developed to transfer data to rapid prototyping and manufacturing systems.

4. CT/MRI Data

One of the important application areas of RP&M is in medical, especially for complex surgery diagnosis, planning, simulation, and implantation. Computerized tomography (CT) scan data is one approach for medical imaging [20]. This is not standardized data. Formats are proprietary and somewhat unique from one CT scanning machine to another. The scanner generates data as a grid of three-dimensional points, where each point has a varying shade of gray indicating the density of the body tissue found at that particular point.

Originally, the models coming from the CT scan are produced using thin plates of aluminum cut according to contours on CT scans and stacked to obtain the model. Simultaneously, molds are used to make prostheses or implants in biocompatible materials. Later, CNC machines are introduced to make these models. However, these processes have reached the limits of the technique such as limited resolution and others. RP&M is very suitable for the layer-by-layer information from a CT scanner. It is possible to produce structures of human bodies by RP&M systems through interfacing the CT data to them. However, converting CT data to rapid prototyping and manufacturing systems is much more difficult than converting the STL format. A special interpreter is needed to

process CT data. Currently, there are three approaches to make models out of CT scan information: through CAD systems, STL interfacing, and direct interfacing [20]. Besides a CT scanner, MRI (magnetic resonance imaging), ultrasound imaging, X-ray imaging, etc. may all be tools to generate the layered images that represent the human organs and also can be reconstructed into what they represent. Recently, the most successful reverse engineering cases that build human parts by RP&M with the layered image data are from a CT or MRI scanner.

C. Interface between CAD and RP&M: STL Format

1. STL File

The STL file, introduced by 3D Systems, is created from the CAD database via an interface to CAD systems [5,21,22]. It is a polyhedral model derived from a precise CAD model by a process called tessellation. This file consists of an unordered list of triangular facets representing the outside skin of an object. The STL has two file formats, ASCII format and binary format. The size of an ASCII STL file is larger than that of the same binary format file but is human readable. In a STL file, triangular facets are described by a set of X, Y, and Z coordinates for each of the three vertices and a unit normal vector to indicate the side of the facet that is inside the object as shown in Fig. 7.

The following is a facet representation as an example trimmed from an STL file that consists of a series of similar facets.

```
solid Untitled1
facet normal 9.86393923E-01  1.64398991E-01  0.00000000E+00
  outer loop
    vertex 9.73762280E-01  7.40301994E-01  1.35078953E+00
    vertex 1.00078931E+00  5.78139828E-01  1.35078953E+00
    vertex 1.00078931E+00  5.78139828E-01  3.50789527E-01
  endloop
endfacet
...
```

The accuracy level of a facet model represented by the STL format is controlled by the number of facets representing the model. Higher part accuracy

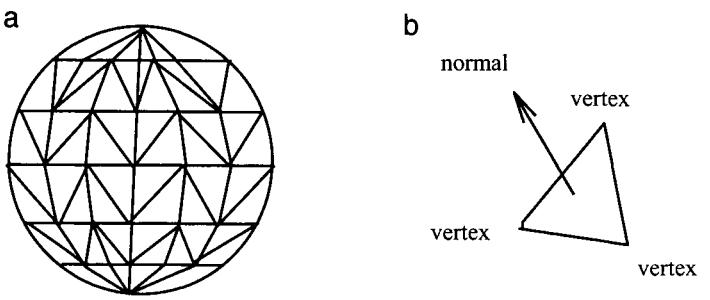


FIGURE 7 Facet model: (a) Triangular representation and (b) single facet (with permission from Springer-Verlag).

requires higher resolution on the tessellation of the CAD model. However, higher tessellating resolution means longer processing time, more and smaller triangles, and larger file size.

2. The Advantages and Problems of the STL Format

The STL format is widely used by many 3D layer manufacturing systems because of its advantages. They are the following:

1. It is a simple solid model. The STL format provides a simple method of representing 3D CAD models. It contains only low-level vertex information and no high-level information such as splines and NURBS curves or surfaces in the boundary representation.
2. It is independent. The STL file is a neutral file. It is independent of not only specific CAD systems but also 3D layer manufacture processes.
3. It has been widely used. It is a *de facto* industry standard and has been implemented by most CAD systems and accepted by most 3D layer manufacturing systems.
4. It can provide small and accurate files for data transfer of certain shapes.

However, problems still exist in the STL [23,24]. They are given as follows:

1. It carries much redundant information. A STL file is many times larger than the original CAD data file for a given accuracy parameter. It contains redundant information such as duplicate vertices and edges as shown in Fig. 8.
2. It is not information rich. The STL file only describes the coordinates of vertices in a triangle and the normal of the triangle. Little topological information is available. This is one reason why a long slicing time is needed.
3. Cracks or holes exist in the facet models. The STL file format describes a CAD model solid by its surface; i.e., the solid is defined by a closed shell. The open shell means that cracks and holes exist in the model as shown in Fig. 9.
4. Nonmanifold topology is encountered in the STL file. The tessellation of fine features is susceptible to rounding off errors, which leads to nonmanifold topology of the part where more than two facets share a single edge (Fig. 10a), or facets in different cycles meet at a single vertex (Fig. 10b).
5. An incorrect facet normal may not be absolutely avoided. The problem of incorrect normals happens where two adjacent triangles indicate that the mass of the object is on opposite sides, as shown in Fig. 11.
6. Facets may overlap. Overlapping may occur when one facet intersects another, or the same edge may be shared by more than two facets as illustrated in Fig. 12.

3. Verification and Validation

Except for the flaws of redundant information and incomplete representation, all the problems mentioned previously would be difficult for slicing

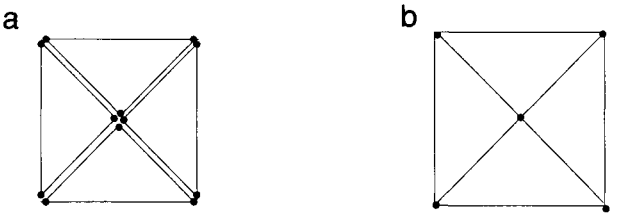


FIGURE 8 Edge and vertex redundancy in the STL file: (a) Duplicate edges and vertices in an STL file and (b) coincident edges and vertices are stored only once (with permission from Springer-Verlag).

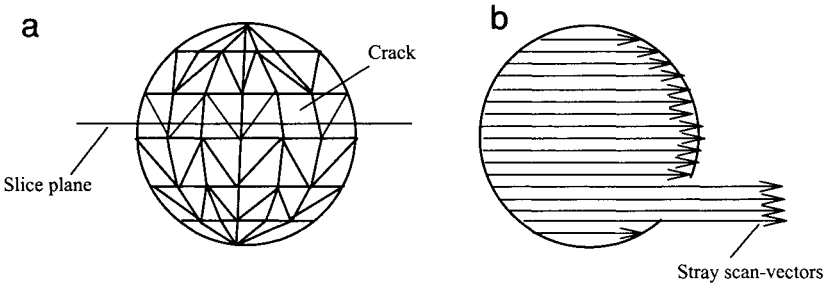


FIGURE 9 Cracks in the STL causes lasers to produce stray scan vectors: (a) Facet model with a crack and (b) cross-sectional view of a slice (with permission from Springer-Verlag).

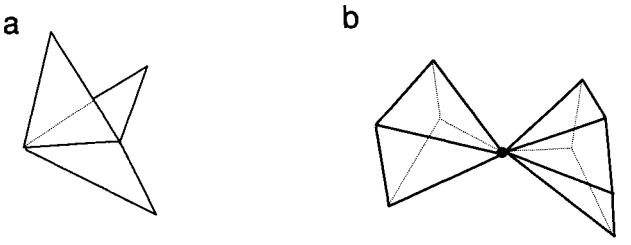


FIGURE 10 (a) Correct and (b) incorrect orientation (with permission from Springer-Verlag).

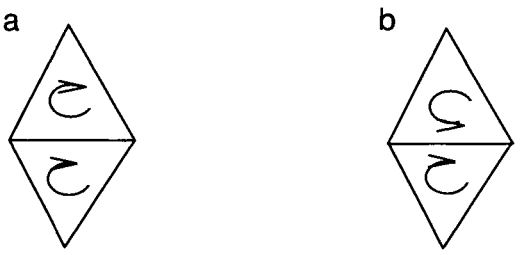


FIGURE 11 (a) Correct and (b) incorrect normal (with permission from Springer-Verlag).

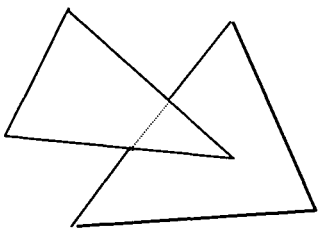


FIGURE 12 Overlapping facets in the STL file (with permission from Springer-Verlag).

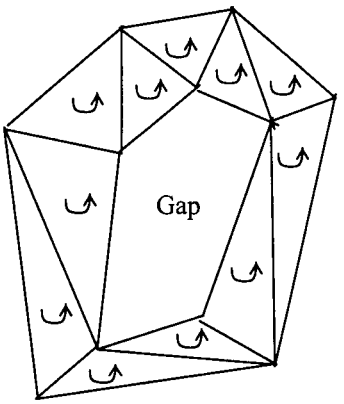


FIGURE 13 A representation of a portion of a tessellated surface with a gap present (with permission from Springer-Verlag).

algorithms to handle and would cause failure for RP&M processes which essentially require a valid tessellated solid as input. Moreover, these problems arise because tessellation is a first-order approximation of more complex geometric entities. Thus, such problems have becomes almost inevitable as long as the representation of the solid model is made using the STL format, which inherently has these limitations.

Instead of canceling the non-error-free STL format, many efforts have been made to find out the invalidation within a STL file and repair the faceted geometric model [23–26] generalizing all STL-files-related errors and proposing a generic solution to solve the problem of missing facets and wrong orientations. The basic approach of the algorithm would be to detect and identify the boundaries of all the gaps in the model. The basis for the working is due to the fact that in a valid tessellated model, there must be only two facets sharing every edge. If this condition is not fulfilled, then this indicates that there are some missing facets, which cause gaps in STL files. Figure 13 gives an example of such a case. Once the boundaries of a gap are identified, suitable facets would then be generated to repair and "patch up" the gaps. The size of the generated facets would be restricted by the gap's boundaries while the orientation of its normal would be controlled through comparing it with the rest of the shell. This is to ensure that the generated facet orientation is correct and consistent throughout the gap closure process, as shown in Fig. 14.

Some RP&M system vendors have built up the functions inside their data processing software, like SGC from Qubical, Israel. In the multifunctional Rapid prototyping system (MRPS) developed by Tsinghua University, China, topological information is generated through reading an STL file. This information is not only useful for data checking and validation, but also helpful for shortening the time on slicing with an upgraded algorithm, by which it becomes unnecessary for searching all the facets in the file when slicing one layer.

Besides RP&M vendors, several software companies supply products based on STL format. Many functions include generation, visualization, manipulation, support generation, repair, and verification of STL files. Materialise

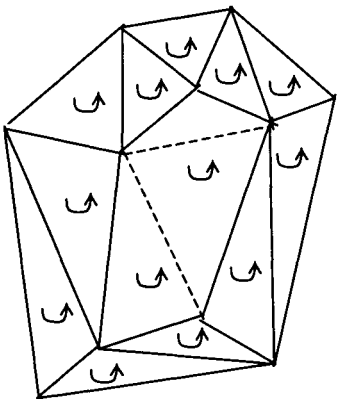


FIGURE 14 A repaired surface with facets generated to patch up the gap (with permission from Springer-Verlag).

(Belgium) provides almost all the operations in the Magic module and is popularly spread across the RP&M world. Imageware (MI, USA) offers its users several competitive advantages for the creation, modification, and verification of polygonal models. It begins every prototyping process by allowing the import and export of IGES, STL, DXF, ASCII, VRML, SLC, and 3D measurement data.

D. Proposed Standard: Layer Manufacturing Interface (LMI)

In the past, most of the RP&M systems are used to produce prototypes for visualization and verification. Now, it is expected that these technologies would be used to produce tooling, production, and functional components as well. As seen in the previous section, there is a need for improving the current *de facto* industry standard for RP&M, or STL format. This section, in which a new edge-based flexible interface, LMI, is proposed to meet the requirements of RP&M processes [27,28], will give a detailed view of how the data structure technique is applied in engineering. The LMI format supports not only faceted solids but also precise solids represented by boundary representation. Design considerations of the LMI format will be discussed first. Before the data structure of the new format is presented, the basic concepts and principles of boundary representation and the topological issues are introduced. This data structure will be mapped into the file of the LMI format, and a geometrical description of the LMI format is presented. Finally, a comparison between the STL format and the LMI format is discussed.

1. Design Considerations

The design considerations must be carefully examined to allow a smooth flow of data from CAD systems to RP&M processes. In addition, CAD solid modeling and RP&M are two expanding fields. Any attempts to specify a standard interface must take these expansions into account. The following requirements are considered in the design of LMI.

1. Facet modeling should be supported. The facet model as used in the STL format is accepted by nearly all RP&M systems. The STL file is, so far, the most common way to send a CAD model to rapid prototyping and manufacturing systems because it is a simple representation of the CAD model and is able to be generated by most CAD systems. Therefore, the newly proposed format should support the facet model. The optimization of the facet model (STL file) is necessary in the new format to avoid defects in the STL format. The improvements and intended improvements of the LMI format are given as follows:

- Add topological information. In the STL format, only unordered vertices with facet normals are provided to describe the faceted models. A vertex is the simplest geometric entity that contains little topological information. In the LMI format, the topological information is supplied by adding new geometric entities and topological data that refer to the topological relations between geometric entities.
- Remove redundant information. The STL file consists of a lot of unnecessary repeated information such as sets of coordinates and strings of text. Normals in the STL can also be omitted because they can be derived from other information.
- Repair errors existing in the STL format like cracks, nonmanifold topology, and overlaps.

2. Support precise models. With increasing experience in data transfer between CAD systems and 3D layer manufacturing systems, it is obvious that preservation of geometric accuracy and geometric intent is important. The problems occur especially with high-accuracy downstream processes such as slicing and high-accuracy machines. Therefore, the LMI format should support precise models.

3. Be flexible and extensible. Because CAD technology, especially solid modeling, and RP&M are two developing areas, flexibility and extensibility of the interface between CAD and 3D layer manufacturing systems must be considered to meet future needs.

4. Be relatively easy to be implemented and used. The format should be easy to be created robustly by CAD systems and processed in the downstream slicing process.

5. Be unambiguous. The LMI format should not contain ambiguous information.

6. Be independent of computer platforms and rapid prototyping and manufacturing processes and commercial systems. This format should be a neutral file so that it can be created by commercial CAD systems.

7. Be as compact as possible. The LMI format should not contain redundant information that would make a file unnecessarily large.

2. Boundary Representation

Geometric modeling techniques have evolved considerably over the past 20 years, embodying more and more information about the physical shape of the objects that they model. A variety of representational forms for solid models have been developed. Each form has strengths and weaknesses in the context of various applications. One approach to classifying solid modeling has been developed by Weiler [29]. Solid modeling representations can be classified based

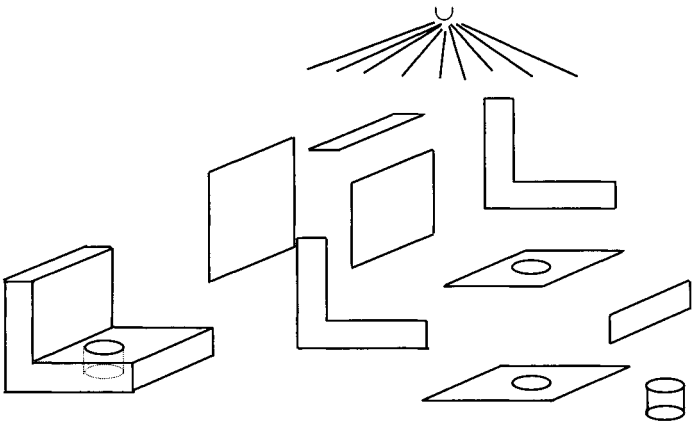


FIGURE 15 Boundary representation (with permission from Springer-Verlag).

on three independent criteria: boundary based (B-rep modeling) or volume based (CSG modeling), object based or spatially based, and evaluated or unevaluated. A representation is boundary based (B-rep, see Fig. 15) if the solid volume is specified by its surface boundary; a representation is volume based (CSG, see Fig. 16) if the solid is specified directly by its volumetric primitives. A representation is object based if it is fundamentally organized according to the characteristics of the actual solid shape; a representation is spatially based if it is organized around the characteristics of the spatial coordinate system it uses. The evaluated and unevaluated characterizations are roughly a measure of the amount of work necessary to obtain information about the objects being represented.

Boundary representation is one of the earliest and widely used geometric modeling techniques. In boundary representation, an object is defined by its boundary that contains a set of entities, such as faces, edges, and vertices embedded in 3D Euclidean space. This boundary divides the space into two distinct volumes, the inside and the outside. The information for boundary representation has two parts, topological and geometric. The geometric information is about dimensions and locations of entities in space. The topological information is the connectivity (or adjacency) information between entities on

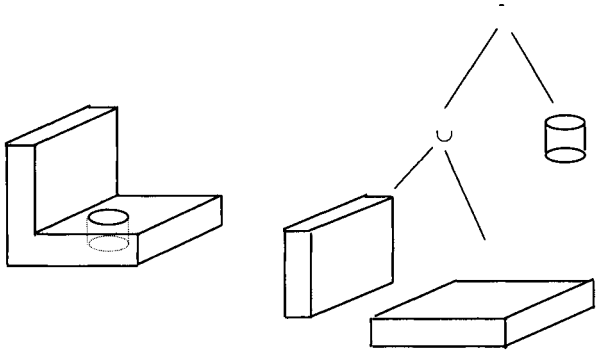


FIGURE 16 CSG representation (with permission from Springer-Verlag).

the boundary. The topological structure in a representation can simplify the application algorithms and greatly improve their efficiency.

3. Data Structure of LMI Format

Any representation requires a domain that contains the complete set of possibilities for which it is valid. It must be carefully specified because it is a key factor affecting the representation. A series of specifications on the geometric and topological domains of the data structure of the LMI format are described as follows.

- *Edge-based boundary representation.* The modified winged-edge structure, and edge-based B-rep, is applied as a representation of the solid object, where a solid is an arcwise-connected set of closed finite regions of space.
- *Compact, oriented two-manifold.* The surfaces of the solids in the data structure are of the solid compact, oriented two manifolds. This implies that the faces that self-intersect or intersect with each other are allowed, forcing the adjacency topology to carry explicitly all surface intersection information through adjacency information. The capability of orientation guarantees that the interior of a solid volume is distinguishable from its exterior.

The boundary solid representation by the data structure of the LMI format uses a graph of edges and is embedded into a compact, oriented two-manifold surface. The embedded graph divides the surface into arcwise-connected regions called faces. The edges and vertices therefore form boundaries for each face in the solid.

- *No restriction on the genus.* There is no restriction on the genus of the modeled object or on the number of interior voids it may have.
- *Pseudographs.* Pseudographs mean multigraphs and may contain self-loops, which are allowed. Self-loops are edges that have the same vertex at either end, and multigraphs are graphs where multiple edges share the same two vertices as end points.
- *Multiply connected.* Faces may be multiply connected; that is, they may have more than one boundary, such as the case of a face with a hole in it.
- *Labeled graph.* All graph elements are labeled. Therefore, each entity in the graph has a unique identity. As the result, nongeometric information to be associated with entities is allowed.

The modified winged-edge data structure, one of the B-rep data structures, is used in the LMI format to represent topological information of entities. The facet representation is a special case of the general B-rep in which polygonal planar regions are used to define the model boundary [30]. The data structure for a triangular faced model is shown in Fig. 17. When the model is a triangular faceted model, the adjacent relationships to be stored are simpler than those for general cases because of triangle characteristics. The adjacent relationship of the topological entities for the facet model is illustrated in Fig. 18.

The data structure of the facet model in the LMI format represents the edge adjacency information as a single unified structure of the modified winged-edge data structure. However, the topological information stored in this structure only consists of the adjacencies of the reference edge with other vertices and

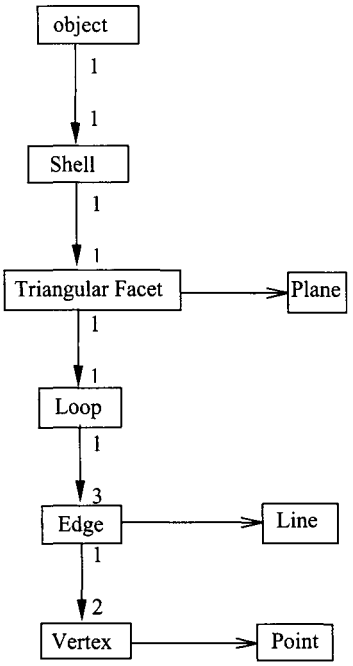
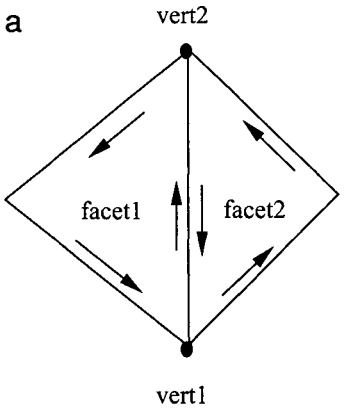


FIGURE 17 Data structure of the facet model in the LMI format (with permission from Springer-Verlag).

facets. The data structure depicted in Fig. 18 will be mapped into the LMI format file.

From the topological viewpoint, the facet modeling technique (indicating triangular facet model) models the simplest solids that have a closed oriented surface and no holes or interior voids. Each facet is bounded by a single loop of adjacent vertices and edges; that is, the facet is homeomorphic to a closed disk. Therefore the number of vertices V, edges E, and facets F of the solid satisfy



b

vert1	vert2
facet1	facet2

FIGURE 18 Adjacency relationship of the facet model in the LMI format: (a) diagram and (b) storage allocation description (see also Fig. 19).

the Euler formula

$$V - E + F = 2. \tag{1}$$

This fact can be used to check the correctness of the faceted model, especially for checking errors such as cracks or holes and nonmanifolds.

4. Description of LMI Format

The LMI format is a universal format for the input of geometry data to model fabrication systems based on RP&M. It is suitable for systems using the layer-wise photocuring of resin, sintering, or binding of powder, the cutting of sheet material solidification of molten material, and any other systems that build models on a layer-by-layer basis.

The LMI format can represent both the precise model and the triangular faceted model derived from the STL file. It can be divided into two parts: header and geometry.

In B-rep, the fundamental unit of data in the file is the entity. Entities are categorized as geometric or nongeometric. Geometric entities represent the physical shapes and include vertices, faces, edges, surfaces, loops, and facets. The nongeometric entity is the transformation matrix entity. All geometrical entities are defined in a three-dimensional Cartesian coordinate system called the model coordinate system. The entity is defined by a series of parameters based on a local coordinate system and a transformation matrix entity to transfer entity to the model coordinate system if needed. The parameterized expression of a general entity is given as

$$(x, y, z) = (X(u, v), Y(u, v), Z(u, v)) \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}. \tag{2}$$

In facet modeling, vertices, edges, and facets are the three fundamental entities. Transformation matrices are unnecessary in the facet model. Therefore, the LMI format includes four sections for a facet model. Each section, except for the Header Section, starts with a keyword specifying the entity type in the section and an integer number specifying the number of the entities in the section. In the section, the data of an entity are a record. The section is ended by two words END and a keyword that is the same as the keyword at the beginning of the section. The four sections in the LMI format are Header, Vertex, Edge, and Facet.

The Header Section is a unique nongeometry section in the LMI file and designed to provide a human readable prologue to the file. The Vertex Section includes geometrical information of all vertices in the facet model. Each vertex is defined by its coordinates in 3D space. The Edge Section is a major section for an edge-based B-rep modeling. It not only gives the geometrical information about all the edges but also contains adjacency relationships between entities. In facet modeling, it contains the information of the adjacent relationship of edges and facets as well as edges and vertices. The Facet Section is a collection of all facets in the facet model. The facet is defined by the edge indices.

a

vertex-index
x, y, z

b

```
typedef struct vertex
{
  int vertex_index;
  double x, y, z;
}Vertex;
```

FIGURE 19 Vertex entity description: (a) Storage allocation and (b) C declaration.

Header Section

The Header Section is designed as a prologue for the file. It consists of a series of keywords and numbers. The keyword defines the item name and the number defines the item by the number.

VERSION: X.X specifies the LMI file format version number.
UNIT: INCH/MM indicates by which kind of units LMI data are represented.
SOLID: SOLIDNAME specifies the solid object name that is given by the user.
LIMITS: MINX, MAXX, MINY, MAXY, MINZ, MAXZ describe the X, Y and Z limitations of the CAD model respectively.

Vertex Section

The Vertex Section consists of X, Y, and Z coordinates for all vertices in the facet model. A vertex is a record. Figure 19 shows the vertex entity description. The content of the Vertex Section in a LMI format file is given as follows:

Vertices: number of the vertices
No. X-coordinate Y-coordinate Z-coordinate
1 9.547921e+01 0.0000000e+00 0.000000e+00
2 7.104792e+02 4.5000000e+00 0.000000e+00
...
End Vertices

Edge Section

The Edge Section in the LMI format provides information of all the edges in a facet model. Each edge includes an edge index, two vertex indices, and two facet indices associated with it. Each edge is composed of two half-edges that have directions according to the vertices of the edges, vert1→vert2 associated with facet1 and vert2→vert1 associated with facet2 as shown in Fig. 20.

a

edge-index	
vert1	vert2
facet1	facet2

b

```
typedef struct edge
{
  int edge-index;
  int vert1, vert2;
  int facet1, facet2;
}Edge;
```

FIGURE 20 Edge entity description: (a) Storage allocation and (b) C declaration.

a

facet-index
half_edge1
half_edge2
half_edge3

b

```
typedef struct facet
{
   int facet_index;
   int half_edge1,
      half_edge2,
      half_edge3;
}Facet;
```

FIGURE 21 Facet entity description: (a) Storage allocation and (b) C declaration.

Facet Section

The Facet Section of the LMI facet model is used to define facets in the model. Each facet is composed of three half-edges. Each half-edge is specified by the edge index and the edge side according to facet fields in the edge, facet field facet1 associated with the half-edge vert1→vert2 and facet2 with the half-edge vert2→vert1. These half-edges are arranged based on the right-hand screw rule. Figure 21 shows the facet entity description.

5. Comparison of the LMI Format with the STL format

It is necessary to make the LMI format and STL format exchangeable. A translator to convert data from STL to LMI needs to be independent of computer platforms and can be implemented on both PCs and workstations. The algorithm used by the translator is shown in Fig. 22. First, a STL file is opened but not fully read in. Otherwise, much space is needed to store a STL file. Then read the data for a facet and compare them with the existing data. After that, the topological relationship between this facet and the other facets is obtained, and the new entities like edges and facets are created and stored. These steps are repeated until the end of the file. Finally, output the LMI format.

In the algorithm, vertex, edge, and facet entities are stored in three link-lists respectively. The functions of operating link-lists, such as, comparing, searching, adding, and others, are needed.

The conversion capability has been developed, allowing the LMI format file to be generated from CAD data or the STL format. The LMI format offers a number of features that are unavailable in the STL format. The comparison between the LMI format and the STL format will be based on the following two aspects: storage and slicing algorithms.

After generating the LMI facet model from STL format data, the topology has been constructed from an unordered STL model facet. The work is done using ASCII format files. The comparison still remains valid for binary formats, because the binary and the ASCII files are identical in contents. Let the number of facets in the facet model be F, the number of edges be E, and the number of vertices be V. For the objects homeomorphic to a sphere, that is, its genus is 0, the Euler formula is

$$V - E + F = 2. \tag{3}$$

Assuming that an edge consists of two half-edges, a facet is composed of three

```
Routine begin

Open a STL file.

If it is not the end of the file

   Read data for a facet.

   For each of the three vertices
        Check if the vertex exists.
        If exists
                Retrieve the pointer to the vertex.
        Else
                Create and store pointer.
        end if.

   For each of the three pointers to vertices
        Check if an edge exists between each pair of the vertices.
        If exists
                Retrieve the edge pointer.
        Else
                Create and store the edge.
        end if

   Create the facet with the three edge pointers.

Else

Output the LMI format file.

end if

End the routine.
```

FIGURE 22 Algorithm translating the STL format to the LMI format.

half-edges. Then

$$F = 3^*(1/2E) = 3/2E. \tag{4}$$

For the STL format, each triangle is described by a set of X, Y, and Z coordinates for each of the three vertices. Each coordinate is a double-precision value that requires eight bytes. Therefore, the size of the storage used to describe a facet in the STL format requires 72 bytes (3 vertices $\times$ 3 doubles/vertex $\times$ 8 bytes/double). Hence, the total size of triangles in the STL file is $72F$, not including normal description and other characters for specification. Then

$$S_s = 72F. \tag{5}$$

For the LMI format, all vertices are described by three coordinates in the Vertex Section. The edges and facets are described respectively in the Edge Section and Facet Section by index values. An index value usually requires two bytes. Therefore, referring to Fig. 19, Fig. 20, and Fig. 21, the sizes of Vertex Section, Edge Section, and Facet Section are $26V$ bytes (2 bytes + 3 vertices $\times$ 8 bytes/vertex), $10E$ (2 bytes + 2 edges $\times$ 2 bytes/edge + 2 facet $\times$ 2 bytes/edge), and $8F$ (2 bytes + 3 edges $\times$ 2 bytes/edge) respectively. Therefore, the total size of

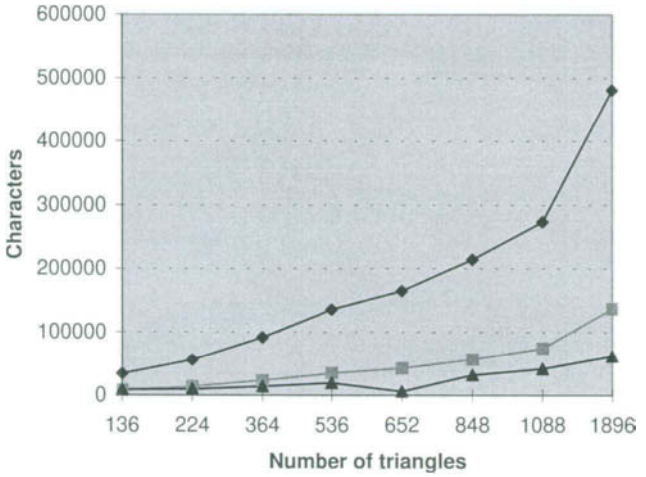


FIGURE 23 Comparison of sizes of STL (♦) and LMI (■) format files. ▲, B-rep model.

the LMI facet model is

$$S_L = 26V + 10E + 8F. \tag{6}$$

Replacing V with formula (3) and E with formula (4), then

$$S_L = 36F + 52. \tag{7}$$

Compare formula (7) with (5):

$$S_L/S_S = 36F/72F + 52/72F = 1/2 + 13/18F. \tag{8}$$

For the simplest solid object, the tetrahedron, $F = 4$. However, usually, it is much larger. Therefore, the size of the facet model in the LMI format is almost half the size of the STL format. However, the real size of the LMI format file is almost one-fourth of the size of the STL format file as shown in Fig. 23.

Figure 23 shows the comparison of the file size of STL format files with the size of corresponding LMI format files. As illustrated in the graph, the file sizes of both STL facet models and LMI facet models increase with the increase in the number of triangles. However, the size of the B-rep model only depends on the complexity of the part; it is not related to the number of triangles. In addition, not only do both the sizes of the STL and LMI facet models increase with the number of the triangles, but also the size of the LMI format is always about one-fourth of the size of the STL format when the same part has the same number of triangles. This is because there is still much other redundant information in the STL file like normal and text.

III. SLICING

Rapid prototyping relies on "slicing" a 3-dimensional computer model to get a series of cross sections that can then be made individually, typically 0.1–0.5 mm in intervals. Slicing is the process of intersecting the model with planes parallel

to the platform in order to obtain the contours. There are various ways to make the slices, each of which has its own advantages and limitations.

A. Direct Slicing of a CAD File

In RP&M, direct slicing refers to slicing the original CAD model. In this way, one obtains the contours without using an intermediary faceted model such as an STL file. Direct slicing has many benefits besides avoiding an intermediary representation. One important benefit is that it makes it possible to use techniques embedded in the application program that results in parts that exhibit a better surface finish, which comes from a better precise contour sliced with the exact CAD model. The computation involves a geometry of the part designed of higher order than that of its triangular description. An approach has been proposed by Guduri *et al.* [31,32] to directly slice a model based on constructive solid geometry (CSG). In the process the primitives in the CSG assembly are sliced individually, generating a cross section for each primitive. The whole contour in the sliced layer is then calculated through combining the primitive slices based on the same Boolean operations as what connects amongst the primitives. The contour of the part in each layer is a collection of piecewise continuous curves. A similar method for manipulating CAD parts in a B-rep or surface model has been introduced. No matter which modeling method is used, the contour finding is realized through solving the degraded equations. For common quadric surfaces (sphere, cones, torus, and ellipsoids), the surface–plane intersection calculation is exact by neglecting the round-off error. For example, the description equation for the boundary of a sphere is listed as

$$X^2 + Y^2 + Z^2 = R^2, \tag{9}$$

where X, Y, Z represent the coordinates of a point on the surface of the sphere that is centered at the original point (0,0,0) with a radius of R.

A given plane with height h_i that is adopted to slice the sphere could be generalized as

$$Z = h_i. \tag{10}$$

For computing the intersection of the sphere and plane, it is just needed to replace Z in (9) with Eq. (10). The result is then

$$X^2 + Y^2 = R^2 - h_i^2, \tag{11}$$

which shows the intersection is a planar circle described by a definite formulation, which is more compact and accurate than an approximate representation. For other surface catalogues, the method requires an approximation of the slice contour of the primitive. Such an approximation is still much more accurate and efficient than the linear approximations obtained from a faceted model.

It is obvious that this intersection calculation method overdepends upon the knowledge of geometric representation in individual CAD systems. Slicing, on the other hand, requires very good process knowledge and is dependent on a user's specification. It also alters the relationship between the user and the manufacturer. Therefore, slicing should be carried out by an expert normally located at the manufacturing site instead of CAD vendors or RP&M users. This

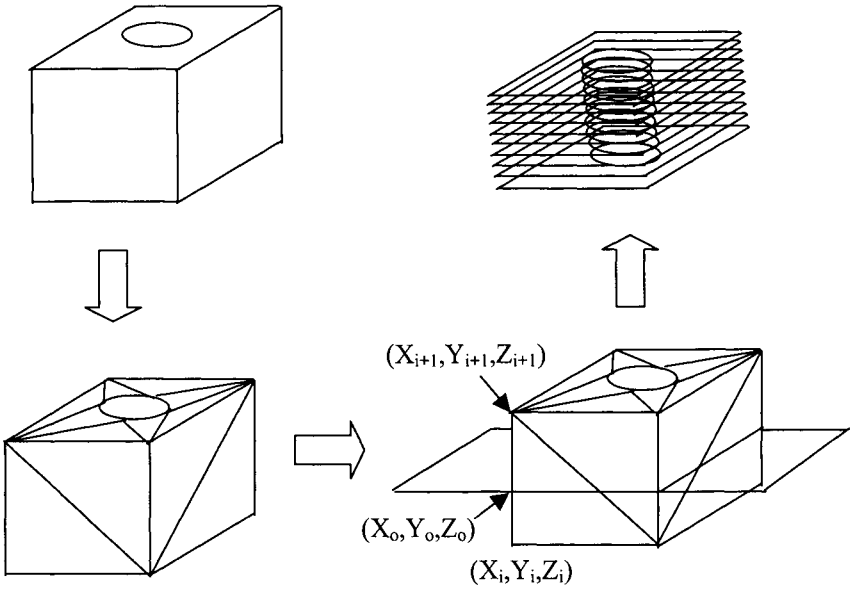


FIGURE 24 Slicing process of an STL file.

relationship will bring to RP&M many of the problems that could be avoided by using a data exchange interface between incompatible CAD and RP&M systems. However, an STL file can be generated so easily by users with little or even no knowledge of the RP&M process that many efforts have been made to slice an STL file.

B. Slicing an STL File

The operation of an STL file is classified as an indirect slice as the result of the difference from slicing with a CAD model. Figure 24 illustrates the procedure of the operation. The triangular facet used in the STL format simplifies the calculations further, since each facet is exactly convex. Mathematically, determining the intersection of a triangular facet and a slicing plane reduces to computing the intersection of the slicing plane with each of the three line segments that define the triangular facets. If the coordinates of the facet vertices are denoted $(X_i, Y_i, Z_i, i = 1, \ldots, 3)$ and the slicing plane is given by $Z = Z_o$, where Z_o is the height of the slicing plane, then the coordinates of the intersection point are obtained by solving the equation

$$\frac{Z_o - Z_i}{Z_{i+1} - Z_i} = \frac{X_o - X_i}{X_{i+1} - X_i} = \frac{Y_o - Y_i}{Y_{i+1} - Y_i}, \tag{12}$$

where (X_o, Y_o, Z_o) are the coordinates of the intersection point. Normally there will be two such intersection points for each slicing plane. These intersection points are then assembled in the proper order to produce the approximate planar contour of the part. It is obvious that the result from slicing a STL file is only polygonal contours in terms of a series of loops at each layer. Each loop is described by listing the vertices that compose the loop, and ordered according to

the right-hand rule (ccw for outer loops and cw for inner loops). This approach may meet some special cases, namely degenerated facets, where a vertex, edge, or entire facet lie in the slicing plane, so that the algorithm should be robust enough to handle them.

C. Slicing an LMI File

Anyway, the random method mentioned above is not efficient since to generate each contour segment during the march is to intersect sequentially each facet with the slicing plane without considering whether the intersection may happen. The performance can be improved by utilizing topological information. Rock and Wozny [33] and Crawford *et al.* [34] have implemented another approach to the slicing facets model based on abstraction of the topological message from the input file and explicitly representation of adjacenct facets through a face–edge–vertex structure. The enhanced method utilizes the edge information and generates contours by marching from edge to edge. Each edge must reference its two vertices and the two facets that define the edge. Each facet must reference its neighboring faces and three edges. The method saves time on searching for the sequential facet to be sliced while other irrelevant facets are put aside untouched temporarily during current slicing. Given Rock's test case performance, it is clear that the slicing algorithm enables it as an on-line operation at the process controller during building.

The slicing of a LMI file is a topological information-based method. It is briefly described using a flow chart in Fig. 25.

Compare the slicing algorithm based on the STL format and that on the LMI format in terms of complexity:

$$\text{Complexity of the STL-based algorithm} = S^*n + S^*p, \tag{13}$$

where S is the number of slices, n the number of facets in the model, and p the average number of points in each slice.

The algorithm in Fig. 25 shows that the searching goes from the first facet to the last facet for each slicing. Hence, the complexity of the algorithm searching for triangles that intersect the z plane is S^*n. S^*p is the complexity of the algorithm sorting intersection lines and linking them together, such that

$$\text{Complexity of the LMI based algorithm} = n + \alpha nS = n(1 + \alpha S), \tag{14}$$

where $\alpha \ll 1$ and α is used to represent the average of the proportions over the slices. The first n occurs during the first slice in the worst case; all n facets must be checked for intersection with the slice. Subsequently, only these facets connected directly to the facets being cut need to be checked, and their number can only be a small proportion of n.

Since $\alpha \ll 1$, $n(1 + \alpha S) \ll nS$ for large n, with $S > 1$. Comparing formula (13) with (14), the cost of the slicing algorithm based on the STL format is much higher than that based on the LMI format.

The LMI format is smaller than the STL format: in the slicing algorithm based on the LMI format, the memory used to store a facet model can be easily freed step by step by freeing the space used to store facets that have been sliced. Therefore, the algorithm based on LMI uses less memory to process.

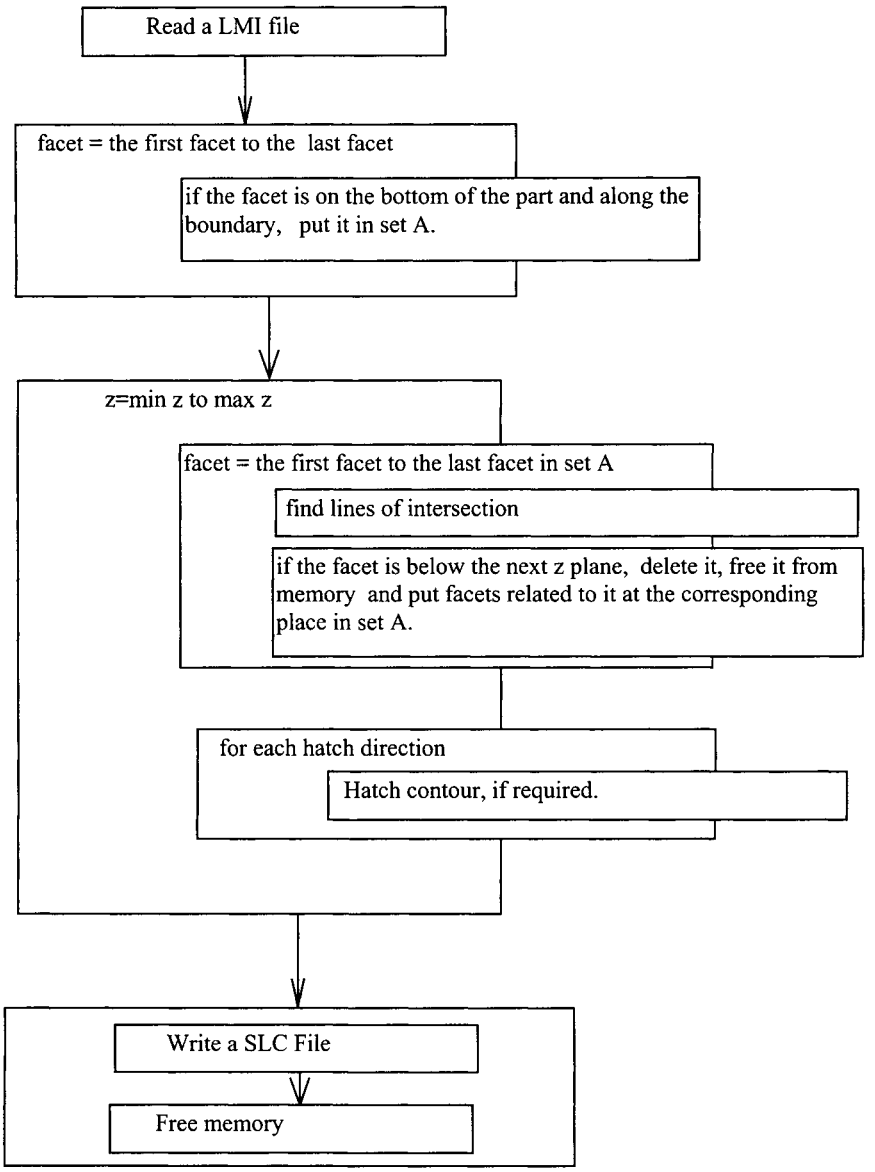


FIGURE 25 Slicing algorithm based on the LMI format.

Summarily, the slicing algorithm based on the LMI format will cost less in both space and processing time than those based on the STL format.

D. Adaptive Slicing

A lot of efforts have been made continuously on improving the accuracy and efficiency of slicing. One of the results of the research is adaptive slicing [35]. The main objective is to control the staircase effect by a user-supplied tolerance. The basic principles of adaptive slicing can be generalized as that vertical and

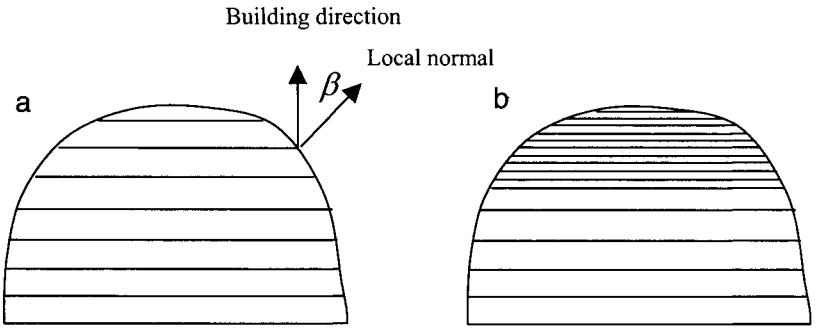


FIGURE 26 Comparison of (b) adaptive slicing and (a) slicing with constant layer thickness.

near-vertical features are built with thick layers while features with a bigger β (the angle between the local normal and building direction) are built with thin layers (as shown in Fig. 26).

Most of the adaptive slicing methods produce unnecessary layers that contribute to increasing fabrication times without improving the overall quality of the part surface; thus they are seldom commercialized. Tyberg and Bohn present an approach of fabricating each part and feature independently so that each thickness is commonly derived from the one part or feature existing at that height whose surface geometry requires the thinnest layer to meet a tolerance criterion [36].

An additional benefit is that one obtains a part with the desired surface finish using the minimal number of layers. Despite these benefits, this building technique cannot be used yet in practice. In some cases, this is due to the underlying characteristics of the RP&M processes. For example, in the FDM process, the layer thickness must be adjusted manually by the machine operator. In the LOM process developed and commercialized by Helisys, the thickness of the layer is determined by the current thickness of the sheet. In other cases, the problem is to find the optimal parameters for a given layer thickness. Let us take, for instance, the SLA. Although the SLA systems make it adjustable, both the manufacturer and the supplier of resins specify building parameters for the layer thickness within limited options.

IV. LAYER DATA INTERFACES

After the geometric model of the part has been sliced, it must be converted to scanning information that will be used to form the 2D layer. It is not clear whether the standard raster scan is the best scanning strategy for RP&M. For example, experiments have shown that tracing the boundary of a part provides a better edge definition. Additionally, the scanning pattern is much different between SLA and LOM. This section will describe the strategies that are derived from the geometric shape of a polygonal contour generated from slicing and data formats that represent the scanning vector.

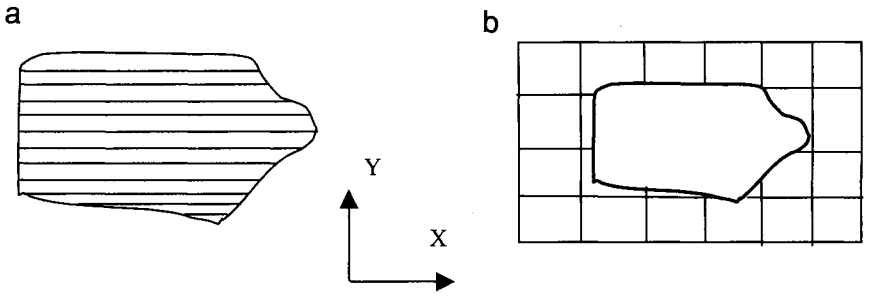


FIGURE 27 Hatching styles: (a) inside and (b) outside.

A. Scanning and Hatching Pattern

1. Raster Scanning

The scanning system from almost all of the RP&M processes consists of a set mirror mounted on a pair of *X*–*Y* linear motion or directly *X*–*Y* linear motion tables. Thus the simplest scanning pattern is raster style, in which scanning motion is always along one axis (either *X* or *Y*) only. The scanning path consists of locating points on the layer boundary at which the laser or nozzle must be on or off. The so-called "locating points" are generated through calculating the intersection points with mathematical rays directed along each scan line and each segment of the boundary contour in each layer. These intersections are then ordered one after another.

Figure 27a describes one typical scanning pattern that is along the *X* axis. RP&M processes like SLA and SLS solidify the liquid resin or powder and FDM depends on the nozzle extruding melted polymer to form the part with interior and boundary, which makes the scan area within the contour boundary as shown in Fig. 27a. However, LOM adopts a different concept to build one layer. In LOM, a laser is adopted to cut the paper sheet according to the contour information, which makes it unnecessary to solidify the interior of the contour. Hatch lines scanned outside of the contour boundary is for facilitating the removal of the redundant material (shown as Fig. 27b).

Of course scanning along the *X* or *Y* axis is not always parallel to the longest edge of the polygon of the contour. Thus the longest edge scan style has been considered because it results in longer uninterrupted active periods of processing material [37]. Changes in scanning direction will not only minimize the number of toggle points, but also affect the physical characteristics of the part built.

2. Boundary Scanning

Boundary scanning may improve a part's surface finish by tracing the part boundary, offset by the appropriate value-like laser beam radius or half-width of droplet of deposited material. The pattern allows the scanning to be continuous with a longer switching time. According to Wu and Beaman [37], geometric contour following for scanning control is used to refine the boundary of the parts for increasing the accuracy or to develop the capability to arrange various scanning directions and paths for improving the part strength. This benefit is

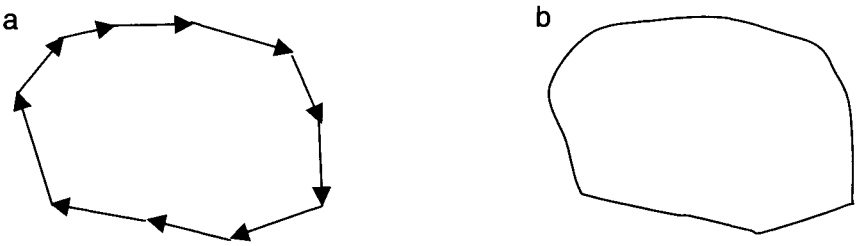


FIGURE 28 Boundary scanning: (a) segment contour and (b) generated continuous contour.

based on the fact that the scanner should be driven to follow the prescribed path as soon as possible, limited by available torque.

In Wu's tracking control strategy, the minimum time optimal control problem with specified path and limited control torque is formulated. The algorithm uses information about the curvature of the contour to determine the appropriate laser parameter to achieve the desired power density. A polygonal approximation of the contour, such as that obtained from slicing a faceted part model, is not accurate enough to support this scheme. Nevertheless, it is still possible that an appropriate continuous contour is obtained through smooth interpolation of the polygonal segment or adding an arc at sharp corners to allow near constant tracking speed, shown as Fig. 28.

3. Model-Based Scanning

Model-based scanning is characterized as adopting unfixed scanning directions compared to raster scanning style. It is proved that besides many other control parameters, the solidification sequence in each RP&M process will have an effect on the physical performance of the final part. For example, initial investigations [38] indicate that fabrication of a metal part with SLS will require local control of laser beam parameters, allowing these parameters to change from layer to layer or even within different areas in a given layer. Such model-based scanning is depicted conceptually in Fig. 29, where a part layer has been divided into several regions based on part quality predictions from a physical model of the process.

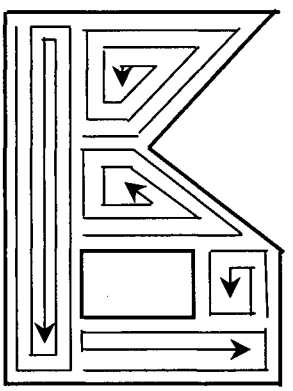


FIGURE 29 Model-based scanning.

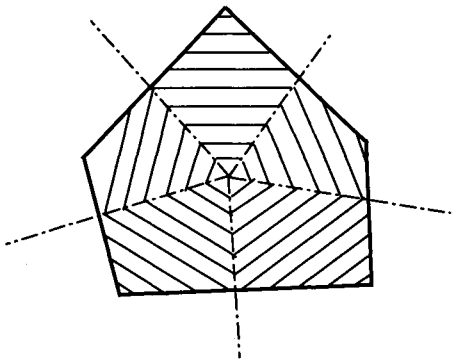


FIGURE 30 Spiral scanning with facilitating the heat expansion.

When describing intelligent slicing, Wozny [39] refers to spiral scanning, in which the center of the spiral is placed at the centroid of the polygonal slice, as shown in Fig. 30. Equiangular rays are extended from this point. An advantage of this approach is that the part is fabricated from the inside toward the outer boundary, which allows expansion due to the heat of the fabrication process to constantly move toward the unprocessed material. It appears that more accurate parts could be obtained in this manner.

Some RP&M processes, which directly build a mold shell prepared for casting rather than a part, need a particular scanning strategy different from other processes. The solidification happens within an area outside of the boundary in each layer. It is a more reasonable way of building a shell with a number of outward offsettings from the original boundary so that the mold made by this approach has a uniform thickness, which has an great effect on the transmission of gas and heat during casting. This characteristic is beneficial for obtaining a good performance from castings. A new algorithm [40] for obtaining the offset contours through a combination of finding the topological information of segments in a digitized image plane and calculating the endpoints of the contours by means of the intersection of two offset lines has been proposed. It has been proved that this method is efficient and robust no matter how complex the planar geometry is.

B. Two-Dimensional Contour Format

For many reasons, 2D slice data are of a valuable format. Some believe a 2D contour representation is more basic than a 3D solid model. At the primary level, each RP&M machine operates on the basis of stacks of 2D drawings. Two-dimensional contour software enables users to work efficiently with this data stream, thus solving special problems and enabling special applications. By tackling the interfacing problem at the contour level, contour software is able to solve all problems with bad STL files.

Many of the 2D contour formats differ in their details, but, in their present form, share common problems:

- *Flexibility.* They lack the possibility of being extended without breaking existing application programs. It is, perhaps, the major reason they will not

obtain industry-wide acceptance and become *de facto* standards. A standard format for slices is a moving target, and changes will certainly be required.

- *Loss of information.* Like the STL format, using these formats will also result in loss of information. For example, it is not possible to represent normal vectors. The normal vectors along the contours can provide valuable information for some processes and pre- and post-processing algorithms.
- *Lack of user control.* They do not allow the sender to have complete control over the process. Unlike other, traditional manufacturing processes like NC milling, the user of an RP&M machine does not yet have control over all variables that influence the process by means of an open data exchange format.
- *Ambiguity.* The usage of (informal) English in the specification can lead to different interpretations and implementations; witness the current practice regarding previously mentioned "standards" like VDAFS.
- *Impractical assumptions.* Both formats require that the senders make a clear distinction between outer and inner contours, and contours cannot intersect in strange ways (e.g., they cannot exhibit self-intersections). These requirements demand a certain level of sophistication from CAD translators, and our current experience with 3D CAD data exchange formats (VDAFS, IGES, and STL) from various systems indicates this is usually not the case.
- *Process-dependent.* They are often not independent of fabrication processes. For instance, SLC formats developed by 3D Systems are biased toward fluid-based processes such as SLA.

Development in 2D formats in Europe is motivated in good measure by biomedical applications (implants, operation planning), as well as by reverse engineering applications that utilize laser and other scanning devices. Technical interests range from general boundary curve forms (beyond polylines) to more flexible file structures. Rather than tessellate a surface and then slice the tessellated surface, it will reduce error, produce better surfaces, and result in smaller files if the original surface is sliced (direct slicing) and the contour curves with polylines are then approximated.

The requirements for a flexible and vendor-independent format led to the development of the Common Layer Interface (CLI) format. The goal of CLI was a flexible format that applies to all RP&M technologies, permits user-specific data (such as patient orientation in CT scans), and allows data transfer among a wide range of applications. User-specific data are included in the Header Section. There is still some concern that CLI (also the 3D Systems slice format, SLC) is still not flexible enough to be a general file format. CLI seems to be biased toward fluid-based processes and dominated by medical scan data applications. An experimental data exchange format, called Layer Exchange ASCII Format (LEAF), addresses these and other shortcomings such as flexibility, loss of information, lack of user control, ambiguity, impractical assumptions, and process dependency. LEAF is an experimental tool for researchers for promoting standardization and is being developed further by researchers at the Fraunhofer Institute for Manufacturing, Engineering, and Automation in Stuttgart.

The HP/GL format previously discussed in this chapter is adopted in Japan frequently for representing 2D slice data. HP/GL is the *de facto* standard for 2D plotting, where the data are represented in a vector format: start point coordinates, end point coordinates, and pen up/down.

SLC is 3D Systems' contour data format for importing external slice data, and SLI is the company's machine-specific 2D format for the vector commands that control the laser beam.

The Cubital Facet List (CFL) format is based on a polygon-based representation consisting of n-sided polygons that can have multiple holes. The format avoids redundant vertex information and maintains topological consistency. CFL consists of a header and fields containing the total number of vertices (points) and facets in the object, a numbered sequence of vertex coordinates, numbered facets (with a number of holes), and pointers back to their respective vertices.

C. Common Layer Interface (CLI)

The CLI format was developed in a Brite EuRam project (Basic Research in Industrial Technologies for Europe/European Research on Advanced Materials) with the support of major European car manufacturers. It is a universal format for the input of geometry data to model fabrication systems based on RP&M. The CLI format is intended as a simple, efficient, and unambiguous format for data input to all 3D layer manufacturing systems, based on a two-and-a-half-dimensional layer representation. It is meant as a vendor-independent format for layer-by-layer manufacturing technologies.

The CLI file can be in binary or ASCII format. In a CLI format, the part is built by a succession of layer descriptions. The geometry part of the file is organized in layers in ascending order. Every layer is the volume between two parallel slices, and is defined by its thickness, a set of contours, and hatches (optically). Contours represent the boundaries of solid material within a layer, and are defined by polylines.

The CLI format has two kinds of entities. One is the polyline. A polyline is defined by a set of vertex points (x, y), connected contiguously in listed order by straight line segments. The polylines are closed, which means that they have a unique sense, either clockwise or counterclockwise. This sense is used in the CLI format to state whether a polyline is on the outside of the part or surrounding a hole in the part. Counterclockwise polylines surround the part, whereas clockwise polylines surround holes. This allows correct directions for beam offsetting.

The other is the hatch, which is a set of independent straight lines, each defined by one start point and one end point. One of the purposes of the hatch is to distinguish between the inside and outside of the part. The other is that hatches and open polylines are used to define support structures or filling structures, which are necessary for some 3D layer manufacturing systems like SLA, to obtain a solid model.

The advantages of the CLI format are presented as follows:

1. Since the CLI format only supports polyline entities, it is a simpler format than the HP/GL format.
2. The slicing step can be avoided in some applications.
3. Errors in layer information are much easier to correct than those in 3D information. Automated recovery procedures can be used, if required; and editing is also not difficult.

4. It is independent of vendors or fabrication machines, and requires only a simple conversion to the vendor-specific internal data structure of the machine.

The disadvantages of the CLI format include the following:

1. The CLI format only has the capability of producing polylines for the outline of a slice.
2. Although the real outline of the part is obtained by reducing the curve to segments of straight lines, the advantage over the STL format is lost.

The CLI format also includes layer information like the HP/GL format. However, the CLI format only has polyline entities, while HP/GL supports arcs, lines, and other entities. The CLI format is simpler than the HP/GL format and has been used by several RP&M systems. It is envisioned that the CLI format may become an industrial standard like STL.

D. Rapid Prototyping Interface (RPI)

The Rapid Prototyping Interface (RPI) format was designed by the Rensselaer Design Research Centre at Rensselaer Polytechnic [39,41]. It can be derived from currently accepted STL format data. The RPI format is capable of representing faceted solids, but it includes additional information about the facet topology. Topological information is maintained by representing each faceted solid entity with indexed lists of vertices, edges, and faces. Instead of explicitly specifying the vertex coordinates for each facet, a facet can refer to them by index numbers. This contributes to the goal of overall redundant information reduction.

The format is developed in ASCII to facilitate cross-platform data exchange and debugging. A RPI format file is composed of a collection of entities, each of which internally defines the data it contains. Each entity conforms to the syntax defined by the syntax diagram as shown in Fig. 31. Each entity is composed of an entity name, a record count, a schema definition, a schema termination symbol, and the corresponding data. The data are logically subdivided into records which are made up of fields. Each record corresponds to one variable type in the type definition.

The advantages of the RPI format include the following:

1. Topological information is added in the RPI format. As the result of this, flexibility is achieved. It allows users to balance storage and processing cost.
2. Redundancy in the STL is removed and the size of the file is reduced.
3. Format extensibility is made possible by interleaving the format schema with data as shown in Fig. 31. New entities can be added to the format and new variables can also be added to existing entities.
4. Representation of CSG primitives is provided, as are capabilities of representing multiple instances of both faceted and CSG solids.

The disadvantages of the RPI format are the following:

1. An interpreter which processes a format as flexible and extensible as the RPI format is more complex than that for the STL format.

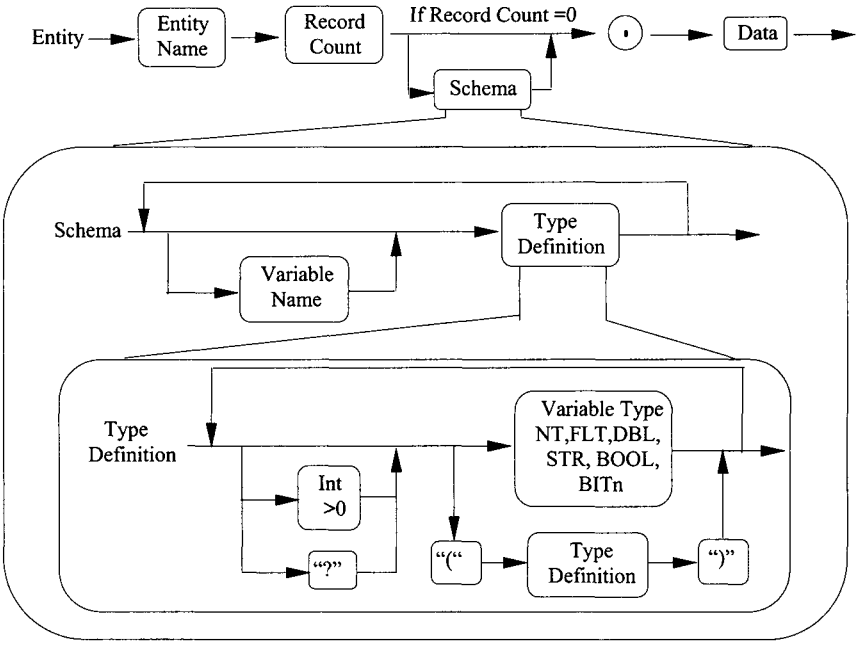


FIGURE 31 RPI format entity syntax diagram.

2. Surface patches suitable for solid approximation cannot be identified in the RPI format.

The RPI format offers a number of features unavailable in the STL format. The format can represent CSG primitive models as well as faceted models. Both can be handled with Boolean operators like union, intersection, and difference. Provisions for solid translation and multiple instancing are also provided. Process parameters, such as process types, scan methods, materials, and even machine operator instructions, can be included in the file. Faceted models are more efficiently represented as redundancy is reduced. The flexible format definition allows storage and processing cost to be balanced.

E. Layer Exchange ASCII Format (LEAF)

The LEAF, for Layer Exchange ASCII Format, was generated by the Helsinki University of Technology [42]. To describe this data model, LEAF borrows concepts from the object-oriented paradigm. At the top level, there is an object called the LMT-file (Layer Manufacture Technology file) that contains parts which in turn are composed of other parts. Ultimately, layers are composed of 2D primitives, and currently the only ones that are planned for implementation are polylines.

For example, an object of a given class is created. The object classes are organized in a simple tree as shown in Fig. 32. Attached to each object class is a collection of properties. A particular instance of an object specifies the values for each property. Objects inherit properties from their parents. In LEAF, the geometry of an object is simply one among several other properties.

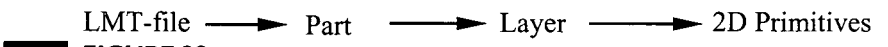


FIGURE 32 An object tree in LEAF.

In this example, the **object** is a LMT-file. It contains exactly one child, the object **P1**. **P1** is a combination of two parts, one of which is a support structure and the other is **P2**, again a combination of two others. The objects at the leaves of the tree—**P3**, **P4**, and **S**—must have been, evidently, sliced with the same z values so that the required operations, in this case **or** and **binary-or**, can be performed and the layers of **P1** and **P2** are constructed.

In LEAF, some properties, like **support-structure** and **open**, can also be attached to layers or even polyline objects, allowing the sender to represent the original model and the support structures as one single part. In Fig. 33, all parts inherit the properties of **object**, their ultimate parent. Likewise, all layers of the object **S** inherit the open property, indicating that the contours in the layers are always interpreted as open, even if they are geometrically closed.

Advantages of the LEAF format include the following:

1. It is easy to be implemented and used.
2. It is not ambiguous.
3. It allows for data compression and for a human-readable representation.
4. It is both CAD system and LMT process independent.
5. Slices of CSG models can be represented almost directly in LEAF.
6. The part representing the support structures can be easily separated from the original part.

The disadvantages of the LEAF format are the following:

1. A new interpreter is needed for connecting the 3D layer manufacturing systems.
2. The structure of the format is more complicated than that of the STL format.
3. The STL format cannot be translated into this format.

```
(LMT-file
(name Object) (radix 85) (units 1mm) ...
(Part (name P1) ...
(binary-or (Part (name S) (support-structure) (open) ...
                 (Layer ...))
           (Part (name P2) ...
                 (or (Part   (name P3)) ...
                     (Layer (name ...) (polyline ...)))
                     (Part   (name P4) ...
                     (Layer (name P4_L1) (polyline ...)))
                 )
            )
)))
```

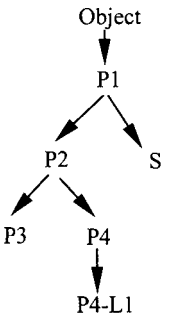


FIGURE 33 Instance tree in LEAF.

The LEAF format is described in several levels, mainly at a logical level using a data model based on object-oriented concepts, and at a physical level using a LISP-like syntax as shown in Fig. 33. At the physical level, the syntax rules are specified by several translation phases. Thus defined, it allows one to choose at which level interaction with LEAF is desirable, and at each level there is a clear and easy-to-use interface. It is doubtful if LEAF currently supports the needs of all processes currently available but it is a step forward in that direction.

F. SLC Format

The SLC file format (3D Systems, 1994) was generated by 3D Systems for representing cross-sectional information [43]. It is a $2^1/_2$D contour representation and also can be used to represent a CAD model. It consists of successive cross sections taken at ascending Z intervals in which solid material is represented by interior and boundary polylines. SLC data can be generated from various sources, either by conversion from solid or surface models or more directly from systems that produce data arranged in layers, such as from CT scanners.

The SLC format only contains the polyline that is an ordered list of X–Y vertex points connected continuously by each successive line segment. The polyline must be closed whereby the last point must equal the first point in the vertex list.

One of the strengths of the SLC format is that it is a simple representation of the solid object. In addition, the SLC format is directly accepted by rapid prototyping and manufacturing systems and does not need to be sliced in some cases.

However, the major weakness of the SLC format is that it can only approximately represent solid objects.

Through the comparison of several currently existing neutral formats and several proposed formats used for RP&M systems, it is clear that each format has its limitations in different aspects. Because of these, many efforts for improving the information process in RP&M and also for catching up with the latest development in engineering have been carried out. The next section will introduce these explorations, including solid interchange format, virtual reality modeling language, and volumetric modeling to support RP&M.

V. SOLID INTERCHANGE FORMAT (SIF): THE FUTURE INTERFACE

Several current RP&M research efforts are focused on the development of a future alternative data format for addressing the shortcomings of STL and for enabling data transfer for future advanced rapid manufacturing capabilities. This alternative data transfer mechanism is referred to as the Solid Interchange Format (SIF). To date, the SIF is a research topic only, with no current consensus or commercial implementation. Various proposals for possible SIF requirements, content, structure, data format, etc. have been submitted for consideration within the RP&M research community. There are a lot of universities contributing to the discussion of alternative data formats and development of SIF.

The solid interchange format will be based upon a precise, mathematical representation of the 3D part geometry. In addition to addressing the current

limitations of STL, the data representation capabilities of SIF must accommodate expected capabilities of future RP&M systems, including the use of multiple materials, gradient material properties, part color specification, nonplanar build layers, explicit fiber directions, surface roughness, tolerance, and embedded foreign components. Also SIF is designed to have strong extensibility and additional annotation capabilities.

As with the proposed improvements to the STL-based data transfer, development and acceptance of an alternative RP&M data transfer mechanism will require support of both RP&M systems developers and CAD vendors. Modifications would be required within both CAD and RP&M vendor products to implement a new data interface. The proposed approach is to standardize an industry-consensus SIF through the ANSI/ISO process when the technical concepts have matured and RP&M industry support is formed. The basis for this representation might be provided by the ASIC save file format, since ASIC has been incorporated as the geometry engine for many commercially available geometric modeling systems.

Although it has been announced to be under development and liable to change, contents of the SIF/SFF (SFF here represents RP&M) already include four major categories: lexical conventions, grammar, semantics, and example, whose detail is available on the Web site of http://http.cs.berkeley.edu/~jordans/sif/SIF_SFF.html.

Another language, L-SIF (Layered Solid Interchange Format), can be developed to describe $2\frac{1}{2}$D layers. Based on this standard, a slicer for generating the layered description from the 3D description can be developed. Other translators might be developed to perform certain transformations on layer-based data such as that obtained from laser digitizing measurements, CT, and MRI.

VI. VIRTUAL REALITY AND RP&M

A. Virtual Prototype and Rapid Prototype

Virtual reality (VR), until recently the preserve of the research labs, is gaining acceptance as a general potential tool. VR allows users to see and explore new products, plans, or concepts long before they exist in reality. It is likely that the uses for VR will coincide with applications that rapid prototyping systems have already been used for. Gibson *et al.* [44] have investigated both VR and PRM and concluded that it is a more efficient route for product development to combine each advantage. The primary use for a rapid prototyping system is not therefore in the qualitative assessment phase of product development. Also much more efforts are on making tooling processes to produce test parts and for short production runs. This is a much more important role for this technology to fill. Competitive marketing policies still dictate that physical models be created for purposes like tendering and user evaluation. VR, with its capacity to model real life, provides a practical replacement for rapid prototyping in this sense. A virtual prototype in VR has the potential of fulfilling at least four main functions, like visualization, verification, iteration, and optimization. There is no possibility of VR fulfilling the most important function, that of fabrication. The ideal product development environment is therefore a rapid prototyping

base supported by CAD systems to supply the engineering detail while VR will be linked to the CAD systems for the consideration of aesthetic, communication and optimization.

SolidView, a software package for RP&M, designed to facilitate the preparation of STL files for RP&M, also contains the facilitation of 3D communication through the use of virtual prototypes. SoildView's viewing, measuring, and annotating capabilities allow the user to create an electronic design review using a virtual prototype. In conjunction with STL files, since each view has its own set of measurement and annotations, users can literally "walk around" the design, showing areas of change or special concern. Also furthermore, it is possible to share the virtual prototype, in the form of a specific file, to anyone on the Internet.

B. Virtual Reality Modeling Language (VRML)

The Virtual Reality Modeling Language (VRML) is a language for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia. VRML may be used in a variety of application areas such as engineering and scientific visualization, multimedia presentations, entertainment and educational titles, Web pages, and shared virtual worlds. All aspects of virtual world display, interaction, and internetworking can be specified using VRML. It is the intention of its designers that VRML become the standard language for interactive simulation within the World Wide Web. VRML is based on the Open Inventor ASCII File Format from Silicon Graphics, Inc., which supports descriptions of computer graphics 3D scenes with polygonally rendered objects, lighting, materials, ambient properties, and realism effects. The first version of VRML allows for the creation of virtual worlds with limited interactive behavior. These worlds can contain objects that have hyperlinks to other worlds or data objects. When the user selects a link to a VRML document from within a correctly configured World Wide Web browser, a VRML viewer is launched for navigating and visualizing the Web. Future versions of VRML will allow for richer behaviors, including animation, motion physics, and real time multiuser interaction.

The Bremen Institute for Industrial Technology and Applied Work Science (BIBA) has opened a Web dialog on replacing the STL format with the Virtual Reality Modeling Language (VRML) format (http://www.biba.uni-bremen.de/users/bau/s2v.html). VRML will help to do the following:

- ***Improve the communication and information exchange in the process chain.*** As the different steps involved in the RP&M process chain are often done by different companies and even to meet different requirements, there is a need of communication and sharing information among different sites. By using a standard like VRML, which is not restricted to RP, many standard software tools, mostly independent from a specific hardware platform, are available, mostly as shareware.
- ***Lower the barrier for RP & M application.*** VRML can be used as a replacement for STL with a slight barrier. It has a node structure. When they are ever needed, RP-specific subnodes can easily be defined. By using VRML, all

can be done with one standard. You do not have to buy several costly interfaces to your CAD package, or use a conversion service where you are not even sure whether you get it right the first time.

- *Open new markets.* VRML reaches the consumer market and therefore has enormous power in contrast to STL. When RP&M apply VRML right now, we can profit from developments done by others. As concept modelers like 3D Systems Actua and low- end "3D printers" appear, VRML opens the mass market to them.
- *VRML files are more compact than STL.* This is just because it uses a list of numbered points followed by a list of triangles (three numbers, order indicates surface normal direction). STL has a large overhead: For each triangle, not only are all 3D points given, the surface normal vector is included, too. It also makes verification and correction easier.

Advocates of replacing STL point out that VRML deals with more issues, including 3D extension of the World Wide Web and future 3D telecommunication and networking standards that could also become the standard interface for all design and manufacturing activities. This format will be applied for storing and viewing with a 3D Web browser before printing for telemanufacturing. Advocates claim that the adoption of VRML would make RP&M more accessible globally, but much is still left to do. VRML today provides a geometry-viewing capability but has not addressed engineering needs such as the creation of very complex geometries and the ability to use it in other analyses. Future versions of VRML are expected to include NURBS, but today, such geometry creates a heavy processing load. Additionally, software based on the VRML format is necessarily developed to realize the special information process for RP&M, such as support generation, slicing, and hatching.

VII. VOLUMETRIC MODELING FOR RP&M

Conventional solid modeling has focused on developing models of objects (known as solid models) based on their geometry and topology. These models do not possess material information and are homogeneous. However, due to new developments in the field of CAD/CAM (optimal design using homogenization, layered manufacturing, etc.), it is becoming increasingly important to model heterogeneous objects (objects with varying material/density distribution and microstructures). Preliminary work has been completed toward modeling objects made of a finite number of materials (objects with discretely varying material distribution) and objects composed of functionally gradient materials. Nowadays, solid models produced by most of the commercial CAD systems assume an homogeneous interior. RP&M technology has the exciting potential of prototyping solids with an inhomogeneous interior and/or a microstructured interior.

Furthermore, it will raise some interesting proposals while comparing the nature of geometric modeling (CAD) and physical modeling (manufacturing). As an incremental forming process, rapid prototyping and manufacturing technology fundamentally adds material piece-by-piece in order to build up shapes.

Unfortunately, neither solid modeling nor surface modeling has such a character as to analogize the additional forming process, even in principle. As a result, it unnecessarily increases the complexity of information processing by introducing slice and scanning vector generation, which can be regarded as an inverse procedure for stacking and layer generating in the building process.

When these flaws in the commonly used modeling systems are considered, it is apt to introduce a voxel-based modeling method for the RP&M field. Actually, voxel-based modeling is not a new concept in volume graphics, where it has another name, volumetric modeling. Kaufuman *et al.* [45] proposed that graphics is ready to make a paradigm shift from 2D raster graphics to 3D volume graphics with implications similar to those of the shift from vector to raster graphics. Volume graphics, voxelization, and volume rendering have attracted considerable research recently. The term "voxel" represents a volume element in volume graphics, just like the term "pixel" denotes a picture element in raster graphics. Typically, the volumetric data set is represented as a 3D discrete regular grid of voxels and commonly stored in a volume buffer, which is a large 3D array of voxels. A voxel is the cubic unit of volume centered at the integral grid point. Each voxel has numeric values associated with it, which represent some measurable properties or independent variables (e.g., color, opacity, density, material, coverage proportion, refractive index, even velocity, strength, and time) of the real object.

Unlike solid or surface modeling, volumetric modeling in discrete form makes it close to the idea of the piece-by-piece building process used in RP&M. In image-based systems, like SGC, successive layers of the part under construction are generated by the use of masks that either allow a light source to solidify a photopolymer under the exposed regions or deposit material on the exposed areas of the mask. Each mask is the image of the object's cross section, which is easily generated by taking out all the voxels that have the same Z-axis coordinate value as that desired. Although much of the current installed RP&M systems are vector-based, in which the sequential formation of the contours is by scanning the object's cross section and hatching is needed to obtain the interiors, image-based systems will dominate the market in the long term due to the perks of faster speed and independence of objects' geometric complexity. Furthermore, generation of scanning vectors from an image is still feasible if necessary.

It is the biggest disadvantage that a typical volume buffer occupies a large amount of memory. For example, for a moderate resolution of $512 \times 512 \times 512$ the volume buffer consists of more than 10^8 voxels. Even if we allocate only one byte per voxel, 128 Mbytes will be required. However, since computer memories are significantly decreasing in price and increasing in their compactness and speed, such large memories are becoming more and more feasible. Another drawback is the loss of geometric information in the volumetric model. A voxel-based object is only a discrete approximation of the original continuous object where the properties of voxels determine the object. In voxel-based models, a discrete shading method for estimating the normal from the context of the models is employed [46].

Chandru and Manohar [47] have proposed a system named G-WoRP, a geometric workbench for rapid prototyping, in which the voxel representation scheme (V-rep) provides an efficient interface among the various modules.

Several problems that are difficult using conventional geometry-based approaches have a simple solution using voxel models. These include estimation of mass properties, interference detection, tolerance calculation, and implementation of CSG operation. Further, voxel-based models permit the designer to analyze the object and modify it at the voxel level, leading to the design of custom composites of arbitrary topology. The generation of slices is made simple, and reverse engineering is greatly facilitated.

Presently, there are two approaches to generating an object in volumetric modeled data. The first source of volume data can be produced from a geometrical model, either closed surface or solid. The voxels are used to fill up the interior of the object with exact size and correct properties according to the accurate specification. It is obvious that the more voxels that are adopted, the less the presentation error is. Of course, increasing the number of voxels may need more space for storing and, in turn, cost more time to deal with. The second measure for building a voxel-based model is much like the process called reverse engineering with many examples of successful applications. Volumetric data sets are generated from medical imaging (e.g., CT, MRI, and ultrasonography), biology (e.g., confocal microscopy), geoscience (e.g., electron seismic measurements), industry (i.e., industrial CT inspection), and molecular systems (e.g., electron density maps) [47].

REFERENCES

1. Peter, P. S. *Data Structures for Engineering Software.* Computational Mechanics, USA, 1993.
2. Wholes, T. Rapid prototyping and tooling state of the industry. 1998 Worldwide Progress Report, RPA of SME, Michigan, 1998.
3. Johnson, J. L. *Principles of Computer Automated Fabrication.* Palatino Press, Irvine, CA, 1994.
4. Hull, C. W. Apparatus for production of three-dimensional objects by steorolithography. U.S. Patent, 4,575,330, 1986.
5. Jacobs, P. F. *Rapid Prototyping & Manufacturing.* Society of Manufacturing Engineers, 1992.
6. Bourell, D. L., Beaman, J. J., Marcus, H. L., and Barlow, J. W. Solid freeform fabrication—An advanced manufacturing approach. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 12–14, 1991, pp. 1–7.
7. Levi, H. Accurate rapid prototyping by the solid ground curing technology. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 12–14, 1991, pp. 110–114.
8. Sachs, E., Cima M., Cornie, J. *et al.* Three dimensional printing: Rapid tooling and prototypes directly from CAD representation. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 6–8, 1990, pp. 52–64.
9. Lee, S. J., Sachs, E., and Cima, M. Powder layer deposition accuracy in powder based rapid prototyping. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 9–11, 1993, pp. 223–234.
10. Nutt, K. Selective laser sintering as a rapid prototyping and manufacturing technique. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 12–14, 1991, pp. 131–137.
11. Feygin M., and Hsieh, B. Laminated object manufacturing (LOM): A simple process. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, Texas, August 12–14, 1991, pp. 123–130.
12. Richardson, K. E. The production of wax models by the ballistic particle manufacturing process. In *Proceedings of the 2nd International Conference on Rapid Prototyping,* Dayton, OH, June 23–26, 1991, pp. 15–20.
13. Kochan, D. *Solid Freeform Manufacturing.* Elsevier, Amsterdam, 1993.

14. Greulich, M., Greul, M., and Pintat, T. Fast, functional prototypes via multiphase jet solidification. *Rapid Prototyping J.* 1(1):20–25, 1995.
15. Reed, K., Harrvd, D., and Conroy, W. *Initial Graphics Exchange Specification (IGES)* version 5.0. CAD-CAM Data Exchange Technical Center, 1998.
16. Li, J. H. Improving stereolithography parts quality—Practical solutions. In *Proceedings of the 3rd International Conference on Rapid Prototyping,* Dayton, OH, June 7–10, 1992, pp. 171–179.
17. Famieson R., and Hacker, H. Direct slicing of CAD models for rapid prototyping. *Rapid Prototyping J.* 1(2): 4–12, 1995.
18. Owen, J. *STEP: An Introduction.* Information Geometers, 1993.
19. Bloor, S., Brown, J., Dolenc, A., Owen J., and Steger, W. Data exchange for rapid prototyping, summary of EARP investigation. In *Presented at Rapid Prototyping and Manufacturing Research Forum,* University of Waraick, Coventry, October, 1994.
20. Swaelens B., and Kruth, J. P. Medical applications of rapid prototyping techniques. In *Proceedings of 4th International Conference on Rapid Prototyping,* Dayton, OH, June 14–17, 1993, pp. 107–120.
21. Vancraen, W., Swawlwns, B., and Pauwels, J. Contour interfacing in rapid prototyping—Tools that make it work. In *Proceedings of the 3rd International Conference on Rapid Prototyping and Manufacturing,* Dayton, OH, June 7–10, 1994, pp. 25–33.
22. Donahue, R. J. CAD model and alternative methods of information transfer for rapid prototyping systems. In *Proceedings of 2nd Internation Conference on Rapid Prototyping,* Dayton, OH, June 23–26, 1991, pp. 217–235.
23. Bohn J. H., and Wozny, M. J. Automatic CAD-model repair: Shell-closure. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 3–5, 1992, pp. 86–94.
24. Makela, I., and Dolenc, A. Some efficient for correcting triangulated models. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 9–11, 1993, pp. 126–132.
25. Leong, K. F., Chua, C. K., and Ng, Y. M. A study of stereolithography file errors and repair. Part 1. Generic Solution. *Int. J. Adv. Manufact. Technol.* (12):407–414, 1996.
26. Leong, K. F., Chua, C. K., and Ng, Y. M. A study of stereolithography file errors and repair. Part 2. Special Cases. *Int. J. Adv. Manufac. Technol.* (12):415–422, 1996.
27. Chua, C. K., Gan, G. K., and Tong, M. Interface between CAD and rapid prototyping systems. Part 2: LMI—An improved interface. *Int. J. Adv. Manufact. Technol.* 13(8):571–576, 1997.
28. Chua, C. K., Gan, G. K., and Tong M. Interface between CAD and rapid prototyping systems. Part 1: A Study of existing interfaces, *Int. J. Adv. Manufact. Technol.* 13(8):566–570, 1997.
29. Weiler, K. J. Topology, as a framework for solid modeling. In *Proceedings of Graphic Interface '84,* Ottowa, 1984.
30. Mortenson, M. E. *Geometric Modeling,* Wiley, New York, 1985.
31. Guduri, S., Crawford, R. H., and Beaman, J. J. A method to generate exact contour files for solid freeform fabrication. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 6–8, 1992, pp. 95–101.
32. Guduri, S., Crawford, R. H., and Beaman, J. J. Direct generation of contour files from constructive solid geometry representations. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 9–11, 1993, pp. 291–302.
33. Rock, S. J., and Wozny, M. J. Utilizing topological information to increase scan vector generation efficiency. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 12–14, 1991, pp. 28–36.
34. Crawford, R. H., Das S., and Beaman, J. J. Software testbed for selective laser sintering. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August, 12–14, 1991, pp. 21–27.
35. Dolenc, A., and Makela, I. Slicing procedures for layered manufacturing techniques. *Comput. Aided Design* 26(2):119–126, 1994.
36. Tyberg, J., and Bohn, J. H. Local adaptive slicing. *Rapid Prototyping J.* 4(3):119–127, 1998.
37. Wu, Y-J. E., and Beaman, J. J., Contour following for scanning control in SFF Application: Control trajectory planning. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, August 6–8, 1990, pp. 126–134.
38. Beaman, J. J. *Solid Freeform Fabrication: A New Direction in Manufacturing: With Research and Applications in Thermal Laser Processing.* Kluwer Academic, Dordrecht, 1997.

39. Wozny, M. J. Systems issues in solid freeform fabrication. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, Aug. 3–5, 1992, pp. 1–15.
40. Du, Z. H. *The Research and Development on Patternless Casting Mold Manufacturing Directly Driven by CAD Model.* Tsinghua University, Ph.D dissertation, Beijing, China, 1998.
41. Rock, S. J., and Wozny, M. J. A flexible file format for solid freeform fabrication. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, Texas, August, 6–8, 1991, pp. 155–160.
42. Dolenc, A., and Melela, I. Leaf: A data exchanger format for LMT processes. In *Proceedings of the 3rd International Conference on Rapid Prototyping,* Dayton, OH, June 7–10, 1992, pp. 4–12.
43. 3D Systems, Inc., SLC File Specification, 3D System Inc. Valencia, VA, 1994.
44. Gibson, I., Brown, D., Cobb, S., and Eastgate, R. Virtual reality and rapid prototyping: Conflicting or complimentary. In *Proceedings of Solid Freeform Fabrication Symposium,* Austin, TX, Aug 9–11, 1993, pp. 113–120.
45. Kaufman, A., Cohen, D., and Yagel, R. Volume graphics. *IEEE Computer* **26**(7):51–64, 1993.
46. Kaufman, A. *Volume Visualization.* IEEE Computer Society Press, Los Alamitos, CA, 1990.
47. Chandru, V., and Manohar, S. G-WoRP: A geometric workbench for rapid prototyping. In *Proceedings of the ASME International Mechanical Engineering Congress,* 1994.

12

DATABASE SYSTEMS IN MANUFACTURING RESOURCE PLANNING

M. AHSAN AKHTAR HASIN

Industrial Systems Engineering, Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

P. C. PANDEY

Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

I. INTRODUCTION

The overall manufacturing system is composed of complex functions and activities. Most of the functions and activities are interrelated in some way. For

Database and Data Communication Network Systems, Vol. 2

production, a firm needs the following things [1,14,16]:

1. A variety of processes, machinery, equipment, labor, material, etc.
2. Relevant planning functions, which plan for the above items, in order to make use of them as efficiently and profitably as possible, and
3. Administrative functions as support activity.

To be profitable, not only are good machinery, equipment, or top quality materials necessary, but also good plans are a must. The company must efficiently and properly plan for the materials and resources, in order to ensure production of the right goods and components, in right amounts, at a right time, at an acceptable level of quality, and as economically as possible. For this purpose, an integrated resource planning system with the following major objectives is necessary: Efficient use of resources and materials, and good customer service.

A good materials and resource plan can contribute much to a company's performance, and thereby, profit. This truth has lead toward the development of the popularly known Manufacturing Resource Planning (MRPII), an integrated materials and resource planning system. Nowadays, MRPII is a very known and popular, though simple, tool, in the field of industrial materials management.

This chapter deals with the applications of database techniques in the MRPII system. Section II of the chapter discusses the MRPII concepts and its planning procedure, the next part (Section III) presents the requirements of data elements in the system, Section IV discusses applications of relational database management system, and the last part (Section V) deals with the applications of the object-oriented technique in the MRPII system.

II. MRPII CONCEPTS AND PLANNING PROCEDURE

A. Manufacturing Resource Planning (MRPII): What It Is

The Manufacturing Resource Planning, popularly known as MRPII, is basically an integrated materials management system. It has been defined by APICS (American Production and Inventory Control Society) as:

> A method for the effective planning of all the resources of manufacturing company. Ideally it addresses operational planning in units, financial planning in dollars, and has a simulation capability to answer 'what if' questions. It is made up of a variety of functions, each linked together: Business planning, Production planning, Master Production Scheduling (MPS), Material Requirements Planning (MRP), Capacity Requirements Planning (CRP), and execution systems for capacity and priority. Outputs from these systems would be integrated with financial reports, such as, the business plan, purchase commitment report, shipping budget, inventory production in dollars, etc. Manufacturing Resource Planning is a direct outgrowth and extension of MRP. Often referred to as MRPII (of closed loop MRP). [16, p. 228].

As defined above, the MRPII is a system that integrates several planning and execution steps, such as MPS, MRP, CRP, PAC and purchasing with the support

from inventory, product structure of the goods, and other functions to provide necessary feedback in the earlier steps in a closed loop, in order to generate amount and timings of purchasing and manufacturing.

It is a computer-based materials planning system, sometimes also known as production and inventory planning and control system. This system introduced a shift from traditional two-bin or periodic inventory control policy to a time-phased discrete inventory policy.

The "closed loop MRP" provides information feedback that leads to the capability of plan adjustments and regeneration. The MRPII system provides some additional facilities, such as financial and accounting.

In the 1960s, a main-frame computer was necessary to run an MRPII system, but now, a Window-based PC or UNIX-based work station is sufficient to perform the job [14].

B. Hierarchical Manufacturing Planning and Control

In both short and long runs of plan generation and execution, the system must balance the needs against its capacity, at several hierarchical levels, with differing levels of information details and time spans. For instance, in the case of long-range planning decisions, the plan needs to identify overall business targets, possible areas of investment, market segments, and amount of investment. In the next stage, the mid-range plan, it should identify, in more detail, each product type with its amount in a specific time period. In the shortest version of the plan, it must prepare a plan that can show exact timings of production start and finish, detailed production schedule, resource allocation, etc. [1].

There are five levels in the overall manufacturing planning and control system, as shown in Fig. 1. The level of detail of plans increases from general categories to specific components, with a decrease in time span (planning horizon)

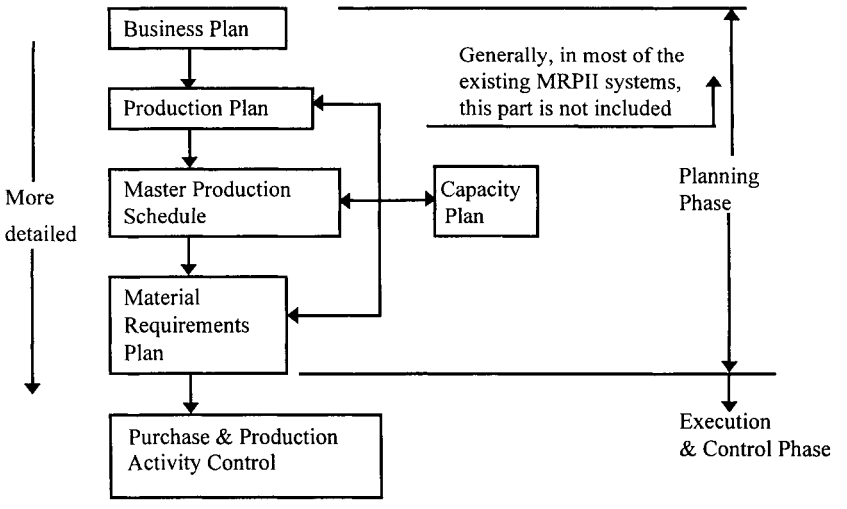


FIGURE 1 Levels of manufacturing planning.

from years to days, when we gradually move from top to bottom. At any level, the target is to find out what, when, and how much of a product or component to produce and/or purchase. Basically, a commercial MRPII system starts functionality from a *master production schedule* (MPS), which is built from two different functions of production plan, namely forecast and customer order.

The MRPII system begins with the identification of each component through a product structure, and then finds out the requirements along with the timings of all those components/subassemblies, based on forecast or customer order, in accordance with available stock on-hand. This misses a major requirement, which is the plans are not generated in accordance with available capacity. During order generation, to fulfill the requirements or production volume, it cannot consider the limitation of capacity of the plant. That is why the system is termed as an infinite capacity planning system.

Some of the major characteristics of this system are as follows:

1. The MRPII system is mostly applicable to make-to-stock, discrete, batch-oriented items. However, job shop, rate-based, and process manufacturing can also be accommodated with variants or enhancement.
2. The MRPII system is divided into separate modules, which have good interaction while preparing materials orders. The main modules are *forecast, bill of materials* (BOM) or *product structure,* MPS, *inventory, control, shop floor control* (SFC) or *production activity control* (PAC), *purchasing, sales* and *order processing, capacity management,* etc. The details of these modules or functions are discussed later.
3. The demand for finished goods, or end items, follows independent inventory, policies, like *economic order quantity* (EOQ), *re-order point* (ROP), or others, whereas demand for subassemblies, components, or raw materials is dependent on the demand for finished goods. The demand for end items are arranged in an MPS. The dependent demand can be calculated from this MPS by time phasing, and is of lumpy type. This calculation is managed by the *material requirements planning* (MRP) module of an overall MRPII system.
4. MRP is part of an overall resource planning (MRPII) system. The program, which generates the materials plan (i.e. orders), is known as the BOM explosion program. Since an MRPII system can prepare plans for all manufacturing resources, such as manpower and machines, including materials, it is popularly known as a *resource planning* system. The generation of a materials plan in accordance with capacity constraint is still not possible.
5. The orders are scheduled based on due dates and an estimated lead time. From the due dates, the order release dates are obtained by going backward, equal to lead time (known as lead time offsetting), in the production calendar. This scheduling process is known as backward scheduling.
6. An MRPII system is driven by MPS, which is derived from forecast, or *customer order,* or a *distribution requirements planning* (DRP) system.

The general structure of an MRPII system is shown in Fig. 2.

The MRPII system is modular, based on the functions/departments it serves. These modules share the manufacturing database, in order to generate combined materials plans for each department. The major modules are described below with their functionality.

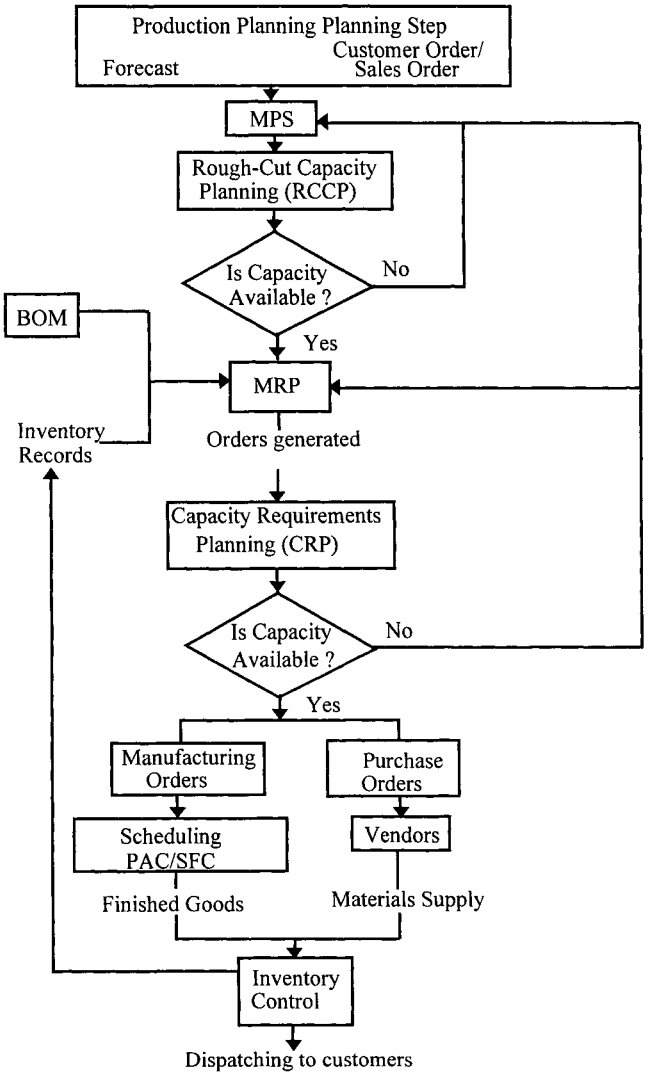


FIGURE 2 The closed loop system.

1. Bill of Materials

The *bill of materials* (BOM) of a product lists all the components, and sometimes the resources (such as labor), required to build a product. The BOM enables the planner to identify not only the components, but also their relationship in the final assembly.

There are several kinds of BOMs to facilitate different purposes. The most commonly used BOM is a tree-type structure. An example of a BOM of a product, a wooden table, is shown is Fig. 3. The major aspects of this BOM are explained as follows: The BOM shows a multilevel product structure, where components are arranged according to parent–component relationships. The part identification (Part ID) numbers are shown in parentheses just below the part name, inside the box. This information is necessary during explosion to

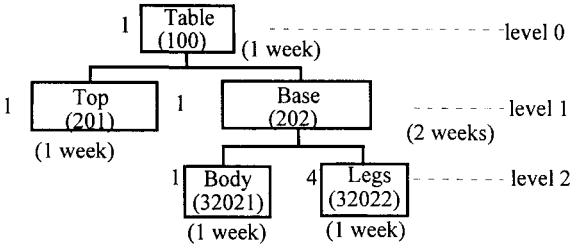


FIGURE 3 Tree-type product structure (BOM).

find out the requirements of the total amounts of the individual parts. The contents in parentheses, outside of the box, show the lead times in weeks. For manufacturing components, it is the manufacturing lead time, whereas it is the purchasing lead time for the purchasing component. A critical path is the line in the BOM that has the longest cumulative lead time. Here, it is 4 weeks, along Table–Base–Legs. This information is necessary during MRP explosion, to find out the total lead time required to manufacture a batch of products (Table). The order release dates for the item are found by subtracting the lead time from the order due date. This subtraction process is known as the *lead time offsetting* for *backward scheduling*. The numbers on the left-hand side of the box show the quantity required per (Q/per) assembly, which means number of components necessary to build one unit of the parent item.

The item at level zero is the table which is known as the *finished good,* or *end item,* and is sold to the customer. This is planned for production in the MPS and is said to have an "independent demand." Below this are the subassemblies and the components required to manufacture a table.

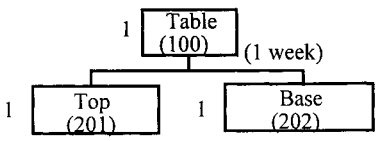
To facilitate easy display of this graphical BOM on a computer screen, it is generally converted to another format, known as indented BOM. The indented BOM of the Table is shown in Table 1.

The BOM in Fig. 3 is known as a multilevel BOM. When a subassembly is shown only one level down, then it is known as a single-level BOM. A single-level BOM for Table (Part ID 100) would be as shown in Fig. 4, where it can be said that "a Table (100) comprises two components, a Top (201) and a Base (202)," whereas the general expression of a "where-used" format of the BOM is the reverse presentation of that. For example, it can be said that "the Base (202) is an item, which is used to make one Table (100)."

TABLE 1 The Indented BOM

Manufacturing Bill of Material for Table, Part ID No. 100

Part ID no.	Part name	Quantity per assembly
100	Table	1
202	Base	1
32021	Body	1
32022	Legs	4
201	Top	1



Tree Structure

Table (100)
Top (201)
Base (202)

Indented format

FIGURE 4 Single-level BOM.

There are several other variants of BOM presentation schemes [1], which are not elaborated here, as this chapter is not intended to include all possible aspects of the MRPII system.

2. Master Production Scheduling

The MPS is the next step in production planning. An estimation is done based on forecasting and/or customer orders/sales orders to identify the demand for a specific time horizon, which is transformed into an MPS. It is an input to the MRP explosion program, which translates the demand for end items to demand for individual components.

To resolve the differences between the MPS and the available production capacity, manual trial and error adjustment is applied. This process is called *rough-cut capacity planning* (RCCP), which is discussed later on.

In case of a make-to-stock item, the MPS is developed based on forecast. The completed finished goods are stored after production, and then sold to the customer from the stockroom. Any standard product, such as electronic goods (e.g., television sets), is of such kind. In case of make-to-order products, the finished goods are produced only when customers place orders. An MPS is developed using real customer orders, but input raw materials and some subassemblies can be procured and stored in advance based on forecast. This is known as job shop, which produces custom-tailored products, such as furniture. In case of assemble-to-order, the major subassemblies are produced in advance, based on forecast, and are kept in the stockroom. The finished goods are assembled from those stocked subassemblies, based on a *final assembly schedule* (FAS).

The MPS contains information on which, how much, and when to complete the production of finished goods. Generally, MPS is developed on a weekly or sometimes daily basis, known as a time bucket, with a total planning horizon at least equal to the critical path time of the product. The planning horizon is the time period for which an MPS is produced. It is done for a period of 3 months to 1 year.

Suppose that the historical demand proportion for two types of Tables, Office and House Table, is 50:50%. Table 2 shows a plan for a weekly time

TABLE 2 An MPS

	Weeks											
Product	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
House	100	100	100	100	110	110	110	110	75	75	75	75
Office	100	100	100	100	110	110	110	110	75	75	75	75

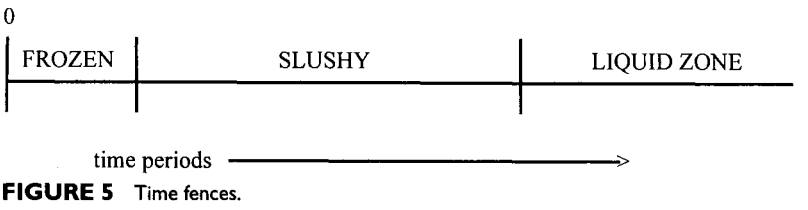


FIGURE 5 Time fences.

bucket and a planning horizon of three months. It is assumed that each month is comprises four weeks.

This MPS is still at the preliminary level, which requires adjustment in terms of capacity. The capacity planning section is given later on. Some terminology related to MPS are as follows

Delivery Promise. It is an amount not booked by any customer, and thus, is available to promise to customers. A delivery promise can be made on the *available to promise* (ATP) amount, where ATP = Beginning inventory + scheduled receipts − actual orders scheduled before next scheduled receipt.

Scheduled receipts. It is the order for which delivery is scheduled, but not yet fulfilled.

Time fences. As discussed earlier, the planning horizon is the time period, generally from 3 months to 1 year, for which the MPS is prepared. This may again be divided into separate zones, depending upon the level of certainty, as shown in Fig. 5 [1]. For the frozen zone, capacity and materials are committed to specific orders. Generally, no changes are allowed in this time period, thus is flagged as "Firm Planned." In the slushy zone, capacity and materials are committed to a reduced extent. Changes are allowed. The liquid zone is completely uncertain. No materials and capacity are committed for this time period [1].

3. Inventory Control

Inventories are those materials and supplies carried on hand by a business organization either for sale or to provide input to the production process [1]. There may be three types of inventories in a production company: Raw materials and subassemblies are materials purchased from the vendors, for use as inputs in the production process; work-in-process (WIP) inventories are partially processed materials in the production line; and finished goods are completed finished goods ready for sale to customers.

The inventory management is responsible for planning and controlling inventory. During planning, it decides the amount to be purchased or manufactured in one batch, known as lot size, and its timings.

It is necessary to classify the inventoried items in terms of their values based on Pareto analysis. This technique is known as ABC classification, which says that a relatively few items in number often constitute a large part in the total in terms of monetary value. It is usually found that the relationship between percentage dollar values of the items and corresponding numbers of items follow a pattern as shown in Fig. 6.

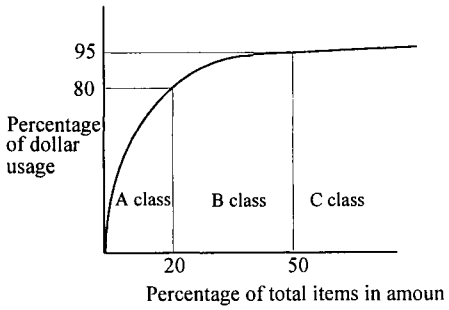


FIGURE 6 ABC Pareto curve.

A class items. Single-order inventory policy, and the Wagner–Whitin algorithm can be used to find out a refined lot size, or lot-for-lot (MRP) purchasing/production can be followed.

B class items. In this case, lot-for-lot (MRP) or a fixed lot size, obtained using EOQ or EPQ (*economic production quantity*), or the least total cost method can be followed.

C class items: In this case, bulk purchasing, fixed lot size, or the EOQ/two-bin system/periodic review system can be followed.

The MPS (or the finished good) is based on an independent inventory policy, whereas the MRP (or the lower-level components in the BOM of the finished good) is based on a dependent policy.

Independent policies. These policies find out the ordering size (volume) and timings of the finished goods. The most widely used policies in an MRPII system are EOQ/EPQ, order point system, lot-for-lot (MRP), etc.

Dependent policies. The components, raw materials, and subassemblies, which act as part of the finished good, do not have an independent demand of their own. The demand of these components depends on the demand of the finished goods. For example, the leg of a table is said to have dependent demand.

Lot-sizing rules, which are commonly used by MRPII professionals, are discussed below.

Lot-for-lot. Order size is exactly equal to the requirement in a time bucket. This is also known as MRP lot size.

Fixed lot. Irrespective of the requirement, a fixed lot or multiples of a lot are ordered to achieve minimum cost. EOQ, the most commonly used rule to determine a fixed lot, is discussed below.

The EOQ (or EPQ) formula tries to determine a lot size that offers minimum cost in terms of the above cost elements. Suppose that for Top of the Table, the order quantity is 400 units and the usage rate is 200 units per week, then Fig. 7 shows its inventory levels with time.

If the economic purchase quantity is Q (EOQ) per order, when each order costs Q for ordering an item, the item has an annual demand rate of D, and the unit purchase price is P at i percentage to carry the materials inside the

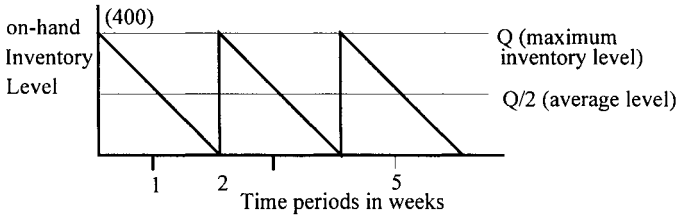


FIGURE 7 Inventory usage characteristics.

stockroom (i.e., inventory carrying, or holding cost), then

$$Q = \sqrt{\frac{2OD}{iP}}. \tag{1}$$

For example, if the purchasing price (P) of Top a Table is \$10/unit, the ordering cost (Q) is \$10 per order, the cost of carrying the inventory is estimated to be 20% (i.e., $i = 0.20$), and it has an annual demand of 10,000 units, then EOQ would be

$$Q = \sqrt{\frac{2 \times 10 \times 10000}{0.20 \times 10}} = 316 \text{ units.}$$

This amount is used as a fixed lot size in material requirements planning.

Although there are different variants of this EOQ policy, and different other policies as well, we will not discuss them all in this chapter.

4. Material Requirements Planning

The material requirements planning (MRP) prepares a schedule for the dependent components and raw materials in the BOM. On the basis of lead time for each component, it computes the timings and amount of these components. This process is known as *time phasing*. The MRP, having inputs from MPS, inventory, BOM, and open orders, finds the following:

1. *What items to order:* The process that identifies the items one by one, by going through the BOM level by level, is called MRP explosion.
2. *When to order:* From the target due date of requirements of the item, the calculation process that finds the order release dates, by calculating backward on the calendar, equal to lead time, is known as *lead time offsetting*. This scheduling process is called *backward scheduling*. Lead time is the time required to purchase or manufacture an item. This includes planning and order processing time, transportation time, materials inspection time, machine setup time, run or processing time, and queuing and moving time.
3. *How much to order.* This is calculated based on the amount in an MPS, on-hand inventory, and open order information. The MRP explosion program calculates this.

In some cases, the MRP should be able to regenerate the orders to satisfy any unforeseen situations on the shop floor, or in some other logistics functions.

Some terminology related to MRP calculation are discussed below [1].

Planned order. The orders generated by MRP explosion program and decided for execution are called planned orders. The order due date is termed as

the *planned order receipt* date, and by lead time offsetting, the date on which the order should be released for execution is known as the *Planned Order Release* date.

Scheduled receipts. Scheduled receipts are orders placed on manufacturing or purchasing and represent a commitment and capacity at work centers allocated. Until the order is completed (closed), this order is termed an *open order.*

Gross and net requirement. Gross requirement is the total amount required for an item, computed from MPS, without considering on-hand inventory or open orders. Net requirements are calculated as

$$\text{Net requirements} = \text{Gross requirements} - \text{scheduled receipts} - \text{on-hand inventory.} \quad (2)$$

Table 3 shows the material requirements plan for the components of the finished good, Wooden Table.

TABLE 3 MRP Computation

Low level code	Item name		Past	Weeks 1	2	3	4	5	6
0	Table	Gross requirement							70
		Scheduled receipt							
		On-hand	40	40	40	40	40	40	0
		Net requirement							30
		Planned order receipt							30
		Planned order release						30	
1	Top	Gross requirement						30	
		Scheduled receipt							
		On-hand	10	10	10	10	10	0	
		Net requirement						20	
		Planned order receipt						20	
		Planned Order Release					20		
1	Base	Gross requirement						30	
		Scheduled receipt							
		On-hand	10	10	10	10	10	0	
		Net requirement						20	
		Planned order receipt						20	
		Planned order release				20			
2	Body	Gross requirement				20			
		Scheduled receipt							
		On-hand	0	0	0	0			
		Net requirement				20			
		Planned order receipt				20			
		Planned order release			20				
2	Legs	Gross requirement				80			
		Scheduled receipt			70				
		On-hand	0	0		0			
		Net requirement				10			
		Planned order receipt				10			
		Planned order release			10				

TABLE 4 MRP for any Component, Lead Time 3 Weeks, Lot-for-Lot Lot Sizing

	Periods, weekly time bucket								
	PD	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Gross requirements		4	10	18	0	10	8	0	15
Scheduled receipts			20						
Projected on-hand	20	16	26	8	8	0	0	0	0
Net requirements						2	8		15
Planned order receipts						2	8		15
Planned order release			2	8		15			

Note. PD means past date/due.

The procedure of calculation is as follows:

1. At level zero, the amount specified in the MPS is the gross requirement for Table. The net requirement is computed using Eq. (2). This is the planned order receipt. By offsetting the lead time of one week, the release time is found.
2. Next, at level 1, for Top and Base, the quantity per assembly is one in each case. Based on the planned order release amounts and time of Table, the gross requirements for both of them are found to be 30 on week 5. This means that the planned order release time and amount of a component at any level becomes the gross requirements for the next level component, when multiplied by the quantity per assembly, and offset by the lead time.
3. The computation continues down to level 2, and the same procedure applies.

In this example, the lot-for-lot lot sizing rule has been used. Examples of some other widely used rules are demonstrated in Tables 4 and 5.

Where it can be seen that the MRPs are different for the same MPS, depending upon the lot size rule.

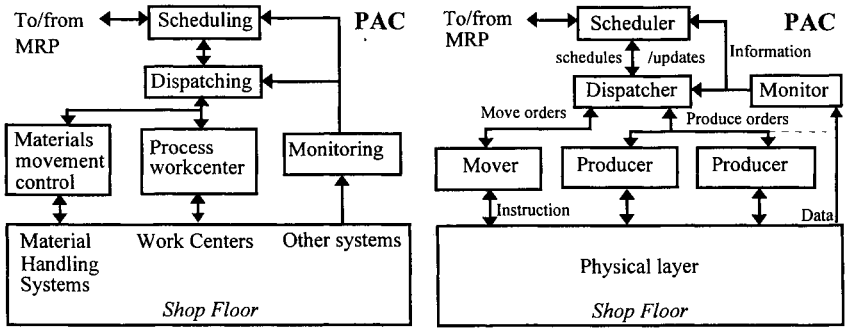
There are several other variants of these MRP policies, which we do not intend to discuss in detail in this chapter [1,15,45].

Recent computerized MRPII systems have the capability of regenerating the planned orders in accordance with the changes in MPS. Two such procedures

TABLE 5 MRP, Lead Time 3 Weeks, Fixed Lot Size of 20 Units

	Periods, weekly time bucket								
	PD	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Gross requirements		4	10	18	0	10	8	0	15
Scheduled receipts			20						
Projected on-hand	20	16	26	8	8	18	10	10	15
Net requirements						2			5
Planned order receipts						20			20
Planned order release			20			20			

Note. PD means past date/due.



PAC Functional elements PAC Functional model

FIGURE 8 PAC System (Ref. 4, copyright Taylor & Francis).

are known: (i) *Net change MRP*, a procedure where only the changed orders are recalculated while others remain unchanged, and (ii) *regenerate MRP systems*, where all the orders are recalculated, in case of any change.

5. Production Activity Control

Production activity control (PAC), or alternatively known as *shop floor control* (SFC), describes the principles and techniques of planning and controling the production during execution of manufacturing orders on the production floor [1,4]. It supports the following major functions:

- Scheduling,
- Dispatching,
- Monitoring,
- Control,
- Capacity management, and
- Physical materials arrangements.

The PAC functional elements and its model are shown in Fig. 8. After approval of the MRP generated orders, they are released as either purchasing orders for those components purchased from the vendors or manufacturing orders for those components manufactured on the shop floor. While the purchasing orders go to the purchasing department as *purchase request*, or *purchase requisition* (PR), the manufacturing orders are released to the shop floor.

One of the most important tasks of shop floor control is to assign priority to jobs, in order to prepare a *dispatching list* [1,4]. The dispatching list arranges the jobs in order to be processed at each work center according to a certain priority. A typical dispatch list is shown below:

Dispatch list

Work center: 5 **Work center ID: Assembly center**

Today: 16/9/98

Job No.	Part ID	Amount	Due date	Run time (hours)	Start date	Finish date
3151	1400	100	27/09/98	32.3	23/09/98	27/09/98
3156	1500	50	30/09/98	30.0	26/09/98	30/09/98
4134	1300	60	10/10/98	40.5	01/10/98	05/10/98

TABLE 6 Dispatching List in Accordance with the Priority Rules

Jobs	Number of days remained until due	Operation days required	Rank	Priority rules	
				EDD	CR
1000	9	7	1st	1001(2)	1001 (0.50)
1001	2	4	2nd	1010 (5)	1010 (0.83)
1003	8	5	3rd	1009 (7)	1000 (1.29)
1009	7	3	4th	1003 (8)	1003 (1.60)
1010	5	6	5th	1000 (9)	1009 (2.33)

There are several priority rules. The two most common rules are (i) *earliest due date* (EDD), where the job are arranged in a sequence of due dates, and (ii) *critical ratio* (CR), which is an index as

$$\text{CR} = \frac{\text{Due date} - \text{Present date}}{\text{Lead time Remaining}} = \frac{\text{Actual time remaining}}{\text{lead time remaining}}.$$

The priority schedules, prepared in accordance with the above two rules, are shown in Table 6.

Recently, information integration has gained momentum, leading toward CIM. The ESPRIT (European Strategic Program for Research in Information Technology) project is an attempt toward that. Out of several islands of automation in manufacturing, possibly PAC [4] is the best module, which requires integration of the most heterogeneous hardware systems. The problem is aggravated because of the closed architecture of CNC (*computer numerical control*) machines and other computer-controlled equipment, and nonimplementation of common 7-layer OSI communication protocol, such as MAP/TOP (Manufacturing Automation Protocol/Technical and Office Protocol). Additionally, PAC is highly a dynamic module, which goes through physical and logical changes frequently, because of changes in customer requirements, changes of equipment, disturbances, changes in product types, etc.

6. Capacity Management

Capacity management is responsible for determining the capacity needed to achieve the priority plan by planning and controlling the available capacity, and arranging for alternatives [1]. It verifies the feasibility of a materials plan, in varying the details at different levels, as shown in Figs. 9 and 10, in terms of resource capacity, where resources are the machines and equipment, and manpower. The capacity is measured in terms of man or machine hours. The capacity problem is still solved by the manual trial-and-error method.

Resource planning (RP) is concerned with long-range (say, for 2 or more years) capacity requirements against production planning. It roughly estimates the gross requirements of labour and machine hours for all or a group of products, for several years of operations. *Rough-cut capacity planning* (RCCP) includes a more detailed capacity requirements plan for individual products, over all weeks, and excludes the detailed shop floor scheduling and real time or unforeseen events. The *capacity requirements planning* (CRP), being the most

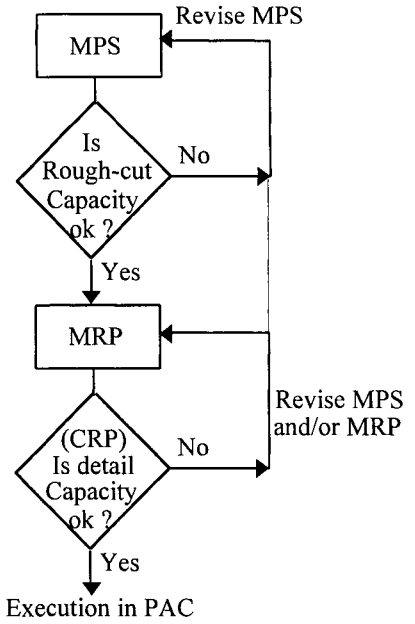


FIGURE 9 Capacity planning method.

detailed capacity planning stage, prepares detailed shop schedules, identifies alternative routing, etc. Generally, RP is not included here, and CRP is not supported by the MRPII systems. So, the following sections discuss only the RCCP method.

During RCCP, the work hours required to perform operations on a batch are verified against the available work hours during the same time period. If the available work hours are greater than or equal to the required hours, the MPS is accepted and approved. If not, then adjustments either to the MPS or to resource management is necessary, by the trial-and-error method.

For visual realization of overload or underload situations, load profiles are used for each work center and time period, and then necessary adjustments are done. For example, suppose that both House- and Office-type tables require the final painting work center, with the data provided in Table 7.

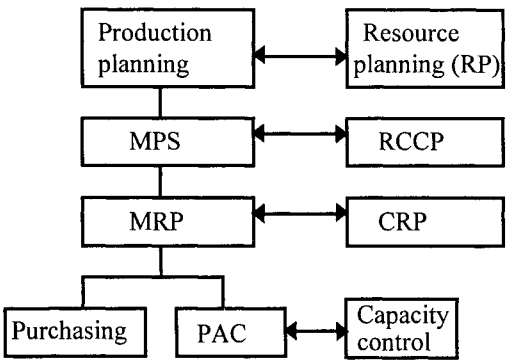


FIGURE 10 Capacity planning hierarchy.

TABLE 7 Required Man Minutes in Period 1

Table type	Production vol. (period 1)	Painting station		
		Run time (min.)	Setup time (min.)	Total time (min.)
House	120	12	60	1500
Office	120	10	60	1260

Note. Total time required: 2760 minutes.

Similarly, suppose that for period 2, total required work minutes is 2000 minutes.

Against the above required work minutes, the available work minutes (man minutes) at that station is calculated as 1 worker, 5 working days per week, 1 shift per day, 8 hours per shift, 60 minutes per hour. Thus, the available man-minutes in periods 1 and 2 = $(5 \times 1 \times 8 \times 60) = 2400$ minutes.

Now, the load profile can be generated as shown in Fig. 11, where it is seen that the required capacity does not match the available capacity in each period. Some manual forward and backward adjustment may, thus, be necessary to keep the required capacity within the available capacity.

A shop calendar is maintained for finding the available work hours. With the assumption that each week comprises 40 work hours (excluding the provision of overtime), Table 8 is an example of a typical shop calendar.

7. Other Modules

Several other modules take part as "supporting modules" in the MRPII system, although they are not directly linked to the MRP explosion algorithm. They provide data as input only. Several modules, such as purchasing, sales, forecasting, accounting, costing, physical inventory verification, and distribution planning, exist within the MRPII system. As these modules do not take part in the algorithm directly, these are not discussed in detail, rather only the data elements that are input from these modules to the closed-loop MRP calculation are given in Section III of this chapter.

8. Operating Reports and Report Writers

Although, in manufacturing, a consolidated report has enormous necessity for management review and decision making, the users of MRPII, on many

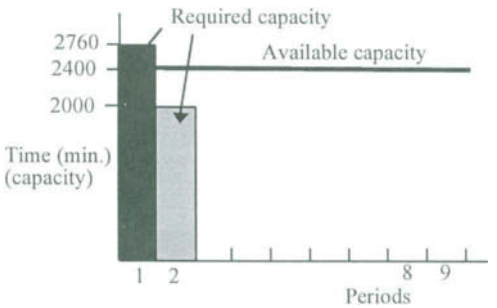


FIGURE 11 Load profile at painting station.

TABLE 8 An Example of a Shop Calendar

September 19xx													
Sunday		Monday		Tuesday		Wednesday		Thursday		Friday		Saturday	
										1	8	2	Nil
3	Nil	4	8	5	8	6	8	7	8	8	8	9	Nil
10	Nil	11	8	12	8	13	8	14	8	15	8	16	Nil
17	Nil	18	8	19	8	20	8	21	8	22	8	23	Nil
24	Nil	25	8	26	8	27	8	28	8	29	8	30	Nil

Note. Notation of calendar: Date | Hours available

occasions, neglect such reports simply because a typical MRPII produces so many reports that the need for additional ones seems to be limited. Moreover, the manufacturing databases are large and highly interrelated, which force the users to spend an enormous amount of computer time and hence, money too, on the production reports that cut across the subsystem boundaries [27,28].

An information center and ad hoc inquiry support task is normally required, and supported too in many MRPII systems, to satisfy special information needs. The problem again is the large and ever-increasing volume of manufacturing data collected and stored in large numbers of files in an MRPII system that must be processed for report generation. The problems of inquiry time, and thus cost, may be reduced in systems that provide summary databases or integrated inquiry and reporting facilities [28]. For example, Max Report Writer is an intelligent tool to help the user analyze information, stored in MRPII, in order to create ad hoc queries and reports. This has the capability to produce as much, or as little, information from the large manufacturing database as the user wants, without further programming [27].

III. DATA ELEMENT REQUIREMENTS IN THE MRPII SYSTEM

A. Database of the MRPII System

The center of an MRPII system is its large manufacturing planning and supporting database. The database may sometimes become so large that during implementation, only developing the database and verifying its accuracy may consume up to two-thirds of the total implementation time. A high degree of accuracy for all main data elements is a must for successful implementation. Many implementation attempts failed because of wrong plan generation from inaccurate databases. Additionally, once the system goes into operation, it becomes essential and a must to update the database on time. Otherwise, duplicate orders may be generated, wrong and false commitments to customers may be made, out-of-stock inventory situations may arise, etc.

In the earlier systems, the files of each distinct manufacturing and planning function were kept separately. As a result, when one file was changed, either the corresponding changes were not made in other functional files or duplicate efforts were necessary to keep all files consistent. As the functions of current

computer-based MRPII systems are highly integrated, they maintain a single set of files. The appointed users can retrieve the information either as display or printout through standard reports. Numerous such built-in reports can be found in MRPII commercial packages, which in the background, uses a standard query system for the inquiries of the users.

Although the arrangement of data elements may vary from system to system, a typical system contains the following files:

Data input files
- Master data files
- Supporting data files
- Transaction (On-going/continuous) data file

Output files
- Manufacturing/purchasing order files
- Status (of different departmental) data files

1. Master Data Files

There are four master data files: *item master file* (IMF), *BOM file, work center file,* and *routing file.*

Item Master File

This is also known as the *part master record* (PMR) file, which contains the records of all parts/components, and even resources, required to build products. This file allows the planner to enter into the system the necessary part information, which may be categorized as the following:

Basic part information:

Part identification. A unique alphanumeric number which is used as a *key* to records. The ID may contain some group technology concepts in order to incorporate some feature qualifiers, such as design and manufacturing attributes, according to a similarity of features available among several components.

Descriptions. A lengthy character field to describe the name of the part, corresponding to the part ID.

User code. Although the above two items are common to any user, sometimes the users may differ as to which items they are interested in. For example, the accounting department may want to include packaging materials inside the BOM such that the cost of the packages may be automatically taken into account while calculating the total production cost of the product, whereas the design and engineering department is generally not interested in defining the packaging materials in the product design and BOM [45]. Similarly, the planning and the shop floor departments are generally interested in putting the tools as resources inside the BOM to facilitate capacity planning, but as it is not a part of the product design, the design department is not interested to include it. By having a user code with different levels and degrees of access, the needs of various interested groups can be served. The reports are also displayed or printed as per user code and other requirements.

Date added and obsoleted. The planner may be interested to keep records as to when the part was defined the first time, and afterward when the part would no longer be necessary for any reason, such as design change.

Several other items of information, sometimes tailored, may be necessary for some organizations not elaborated here [27,45].

Design and engineering information:

Drawing number. Necessary to relate the part ID to its drawing number.
Engineering status. Necessary to track the part in its complete life cycle.

In commercial packages, several other items of information are available as fields in the system database, such as units of measure (UOM), engineering change number, and revision number [27,45].

Low-level code. Level refers to the position where a part fits in the complete product structure. The recent MRPII software has the capability of determining the level of a component through a computation, known as *low-level coding,* specially when the same component appears at different levels in different products. It is not necessary for the planner to specify this; rather the software does this during data processing. In regenerative MRP systems, if explosion processing is simply done by calculating along the path through BOM hierarchies, it would, as a result, replan common components several times over. Here the low-level coding, a data processing mechanism, performs its role. It assigns each component a code that designates the lowest level of the component in any BOM on which it is found. The reason behind this is that MRP explosion processing is done level by level, from bottom to top in a tree, taking into account the total requirements of a component residing at a level [4].

Planning information:

Part type code. An essential field that determines whether a part is an in-house-built part or a purchased part to be supplied by a vendor, as well as whether a part is to be included as a finished good in the MPS or to be planned using ROP or another inventory policy.

In commercial packages, several other items of information are available as fields in the system database, such as manufacturer's part number, planned yield and scrap, etc. [27,45].

Bill of Materials File

This file describes the product structure, which is composed of parts described in the PMR. The records in this file each contain one parent–component relationship, with additional data, such as quantity per assembly, quantity code, BOM-type code, BOM level code, effectivity date, alternate part, lead time for assembly and lead time offset [27,45]. The search procedure for a component in the BOM, along with other information for the part, is explained later in this section.

Work Center File

The *work center* (WC) file and the *shop routing* file may reside under the PAC module. A WC is a single machine or a place, or a group of machines or places, that perform a particular operation on the shop floor. This is required to plan capacity and production schedules. Each WC must be defined by the required information, and must be identified by a unique number, known as

the *WC identifier,* which is used as a key, with associated detailed description. Some other information (or data fields) that are necessary in this file include location, number of machines, number of workers, utilization rate, work hours assigned and available, smoothing factor, allowable queue, and overhead and labor rates (for costing) [27,45].

Shop Routing File

The shop routing describes the sequence of operations, with required WCs and tools, for producing a product. The routing should have an identifying number distinct from the number of the part being manufactured. This allows different parts to have the same shop routing [45]. A typical shop routing file may contain the following information (data elements):

Part ID (to link a routing to a particular part), operation sequence number, work center ID and descriptions, operation type (e.g., unit operation or batch operation), tool name and number, run time, setup time, planned scrap, etc. [27].

A standard shop routing is saved for a part. However, there are times when a particular manufacturing order may require some deviations from the standard one, due to specific reasons. In such a case, changes can be made in the standard file and then saved as an "order routing file," which indicates its application to a particular order only, whereas the standard file remains unchanged [27].

2. Supporting Data Files

Several other files contribute to the above master files to complete the system, depending upon the system's capability. Some major files are discussed, with their functionality and major data elements, below.

Cost Data File

This is used to input and review the cost-related data of a part, defined in IMF. Some major data elements are part ID (a key to IMF), part-type code, cost-type code (manually or automatically calculated), accounting-type code, unit of measure, costing date, direct material cost, direct labor cost, material and labor burdens, etc. [27]. Sometimes, a key to the *general ledger* (GL) accounting system is included in this file. Otherwise, a separate file that acts as an interface between closedloop MRP and a financial and cost accounting system/module may be introduced. The accounting system is also linked to different transaction files, such as materials purchase and sales, to exchange accounting data from/to accounts receivable (A/R) and accounts payable (A/P) files under an accounting system/module.

Inventory/Stock Files

There are two files associated with inventory control. The *stockroom data file* maintains information on available stockrooms in the company. Its data elements are a unique stockroom identification number and name, location, a code to flag whether it is a nettable (a permanent storage for regular transactions against usage) or nonnettable (a location for temporary storage, such as an inspection storage or a materials review board) stock, size/floor space, etc. The *inventory part data file* is necessary for linking part/component

data in IMF to stockrooms where it is stored. It identifies the location (bin/shelve/rack/row/column) of parts in stockrooms (stockroom ID of stockroom data file), along with other inventory control information, such as lot size policy, order size, safety stock, lead time, physical cycle count code, and shelf life.

Purchase Files

These files link the part/component in IMF to its vendors, and other purchasing data/information. Several files may exist in a MRPII system to maintain these data/information. The *purchase part data file* maintains the basic purchasing information of a part if it is a purchased part. The data elements that may be found in this file are part ID (from IMF), purchasing lead times, purchaser's ID, UOM, etc. The *vendor data file* keeps the records, such as vendor ID and name, address, and FOB points, of all possible vendors/suppliers of input materials for the company. The *vendor part data file* identifies the relationships between parts/components and their suppliers (i.e., it shows what part can be purchased from what vendor, which means it basically provides a linkage between purchase part data file and vendor data file). It also provides price break down information for a part.

Sales Files

Two major files can be found in a MRPII system. The *sales part data file* identifies the MPS parts to be sold to the customers. It includes part ID (from IMF), sales price break down, taxation code, and sales UOM. The *customer file* keeps the records of permanent, potential, and past customers for future reference.

MRP File

This file contains information regarding a part's lot size policy in order to determine the amount to manufacture or purchase, and the lead time for backward scheduling. Additionally, it may include the planner's ID and name. Other information in this file are basically fed from other pertinent files, as given above.

Capacity Planning File

Although the current MRPII systems are unable to manage the CRP process, some systems provide connectivity with a third party scheduling software. This is beyond the scope of discussion in this chapter. However, some commercial MRPII systems provide only a RCCP module with limited capability. A RCCP module may contain two files for this process. The *shop calendar file* maintains a yearly calendar marked with weekly and other holidays, in order to exclude those days during backward scheduling. A *capacity maintenance file* may contain data on planned utilization of each work center, planned idleness of a machine for maintenance, etc.

For permanent data maintenance (input, display, print) purposes, in addition to the above major files, several other secondary/optional files, distributed in the pertinent modules, may be found in a system. Some examples of such secondary modules that may be found in a system are *quoting* (for quotation in

bidding), *warranty tracking* (for efficient warranty process), *subcontracting* (for maintaining subcontract orders), *repetitive manufacturing* (in case of a repetitive environment, the module provides backflush capability against the received finished goods), *labor control* (to maintain labor/worker information), *management report writer* (to create summarized reports for management review), *lot tracking* (to track the products produced in a particular lot), etc. [27].

3. Transaction Data Files

In addition to the above files for permanent data maintenance, several other files are necessary for regular and ongoing continuous data inputs against all transactions. Some major transactional data files are discussed below along with their functionality.

Physical Inventory File

This file is utilized to store and retrieve information on a physical cycle count process. The file contains the records of inventory tag data, for use in cycle counts. Some important data elements in this file are tag number linked to part ID from IMF, quantity counted, name and ID of counting personnel, date of count, linked to stock ID from inventory module, etc.

Sales Order Entry File

Although the purchase orders are created automatically from the MRP explosion program, in case of urgency, the MRPII systems provide an additional file for entering a purchase order without any formal planning steps. This is the *unplanned purchase order file,* which includes part ID, vendor ID, line and delivery number, dates, quantity, price, etc.

To monitor and track the materials that are already purchased and waiting for reception or are in transit, a separate file, *reception record file* or *purchase order transit file,* is necessary to keep the records of amounts already arrived, the possible date of arrival, transportation mode, etc.

B. Data Storage and Retrieval in the MRPII System

In most of the traditional MRPII systems, the *record dictionary* and *pointer* [43] are used to identify and retrieve data, such as item/component/materials data from the BOM file, an example of a tree-type BOM of a Pen being given in Fig. 12, with the IMF, with selected fields, for the BOM being given in Table 9.

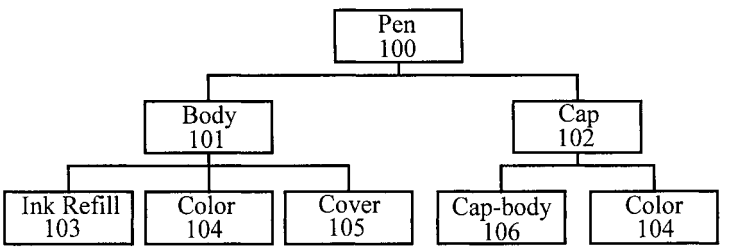


FIGURE 12 Tree presentation of BOM of pen.

TABLE 9 Item Master File (IMF) with Selected Fields

Record address	Part ID	Part name	Vendor, and (Mfg./Pur.)	First parent record	First component record
1	103	Ink Refill	TDCL (Pur.)		10
2	104	Color	TDCL (Pur.)		13
3	105	Cover	In-house (Mfg.)		16
4	102	Cap	In-house (Mfg.)	18	11
5	101	Body	In-house (Mfg.)	10	12
6	100	Pen	In-house (Mfg.)	12	
7	106	Cap-body	In-house (Mfg.)		15

As the part ID is a unique value for any component, it is used as the key to index and search data [43], as given in Table 10. The corresponding BOM file, with parent–component relationships and selected fields, is given in Table 11.

Now, if it is desired to find out the single-level BOM for subassembly Body (part ID 101) along with the information as to whether the component is a Purchased part or an inhouse Manufactured part, and if purchased, then the Vendor name, the record dictionary (Table 10) is searched. It is found that for this part (Body), the record address in IMF (Table 9) is 5. From Table 9, it is found that the pointer to the first parent record in BOM file (Table 11) is 10. When searched in the BOM file (Table 11) against the record number 10, it is found that the first component for Body (part ID 101) is 103 (Ink Refill). The pointer additionally indicates that the next component record can be found at address 13, which is the component with ID 104 (Color). The pointer at this record, in turn, indicates that the next component for this part is at record 16, where the component with ID 105 (Cover) is found. Thus, it is found that there are three components, with IDs 103, 104, and 105, for the parent part Body (ID 101). Additionally, the records at addresses 10, 13, and 16 give pointers to the IMF (Table 9) as 1, 2, and 3. When searched at record addresses 1, 2, and 3 in Table 9, the single-level BOM for subassembly Body (part ID 101) is established as shown in Table 12.

TABLE 10 Item Master, Indexed against Part ID (Key): Record Address Dictionary

Part ID (Key)	Record address
100	6
101	5
102	4
103	1
104	2
105	3
106	7

TABLE 11 BOM File with Pointers

Record No.	Parent (Part ID)	Component (Part ID)	Pointers		
			Component next where-used	Parent's next component	Component's IMF record addr.
12	100	101		11	5
11	100	102			4
10	101	103		13	1
13	101	104	18	16	2
16	101	105			3
18	102	104		15	2
15	102	106			7

C. Information Transaction in MRPII

As mentioned earlier, the MRPII system, which is modular in construction and can be mapped one-to-one with the disparate departments of an organization, integrates the departmental operations into a single processing function, with the flexible option of handling the information of each department either independently or in an integrated fashion. The following sections discuss the information subsystems of the major functions/departments in the overall MRPII system.

1. Order Processing

When the customer wants to buy, he/she can place the order with the sales department (corresponding: Sales module of MRPII). This becomes the *customer order* in the next step of the MPS module of MRPII. In the case of make-to-stock items, the forecast amount and type of products become the MPS. In the case of a hybrid-type demand system, the direct customer order and forecast are combined together to produce the MPS. Then going through the complete MRPII algorithm, the orders are ultimately fulfilled from the finished good stockroom (corresponding: *inventory control* (IC) module of MRPII), by optionally updating the (A/R) of the accounting module of MRPII. This is shown in Fig. 13.

2. Purchasing Subsystem

The purchasing subsystem consists of those activities necessary to acquire raw materials and components for production and other purposes (see Fig. 14).

TABLE 12 Single-Level BOM for Body (Part ID 101)

Part ID	Part name	Vendor, and (Mfg./Pur.)
103	Ink refill	TDCL (Pur.)
104	Color	TDCL (Pur.)
105	Cover	In-house (Mfg.)

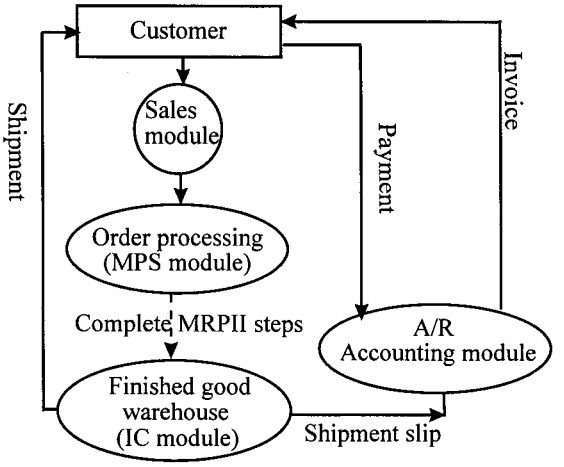


FIGURE 13 Order processing subsystem.

Generally, other requirements are fulfilled as "unplanned order" to combine with the regular manufacturing needs/orders, which are generated through the MRP explosion program. The later orders are known as "planned orders," which need approval in the computer system. All these are done in the MRP module of the MRPII system. The orders for purchasing are known as *purchase requests/requisitions* (PR) until those are assigned to particular vendors.

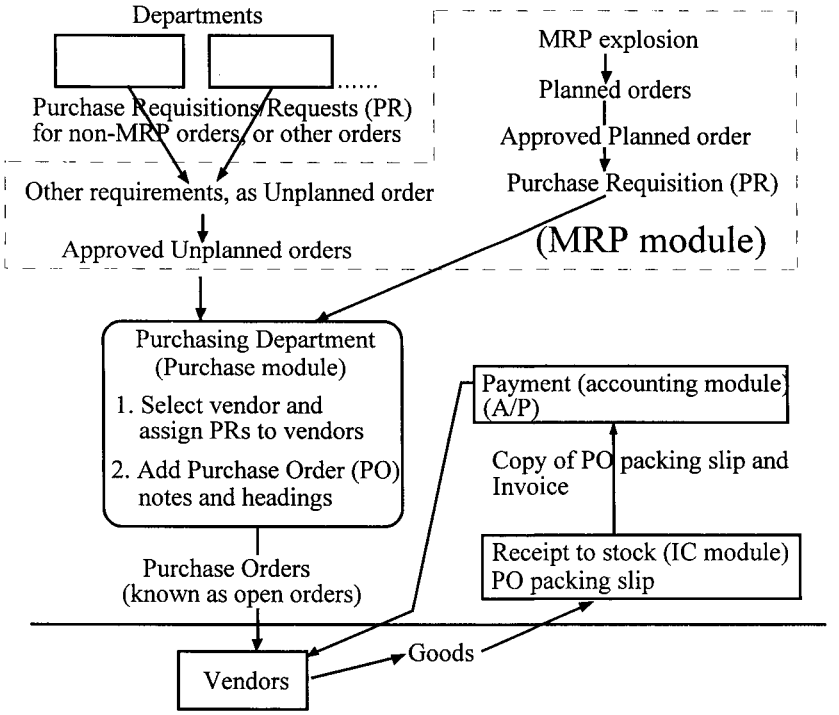


FIGURE 14 Purchasing function subsystem.

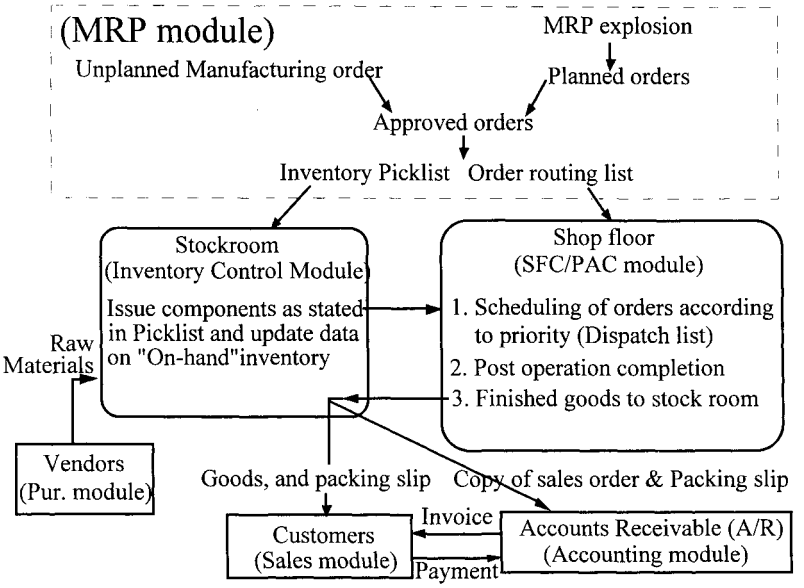


FIGURE 15 Manufacturing function subsystem.

A vendor database with all possible vendor lists are maintained, in the MRPII system, with their addresses, payment and shipment procedures, and possible price breakdowns. The complete process is known as *vendor scheduling*. Once a vendor is assigned to a particular PR and sent, the order status is changed from PR to *purchase order* (PO).

3. Manufacturing Subsystem

The manufacturing subsystem is responsible for generating and managing manufacturing orders (see Fig. 15). Generally, the orders are generated through the MRP explosion program, although some orders may be generated manually, as unplanned order. These orders need approval in the computer system. The manufacturing orders contain two vital lists, namely the *inventory pick list* and *shop routing list*. The inventory pick list lists all the raw materials and components needed to manufacture a particular product. The inventory people verify this list and then send the required materials to the shop floor through a menu item, "issue stock." At the same time, the shop routing list, which describes the requirements of operations, work centers, machines, tools, and sequence of operations to produce a particular product, is sent to the shop floor. After necessary prioritization, using some priority rules, a shop schedule is generated for maintaining the schedule within available capacity. Once production is complete for a particular order, the finished goods are sent to the *finished good stockroom* for shipment to the customers. When payment is also completed from the customer side, the accounting module describes the order as a "closed order".

The three functional subsystems described above are the most important subsystems, although there are several other functional subsystems in the complete MRPII system.

D. Early-Stage Information Systems in Manufacturing Planning

As different tasks in manufacturing planning became tedious, and need time-consuming iterative processes or trial-and-error solutions, there is no need to point out the importance of the role of computers in planning manufacturing resources, coupled with other related management functions. Over the years, several MISs (manufacturing information systems) have been developed for varied purposes, related to production resource planning, and in different details, basically to serve the same needs, an integrated resource planning for manufacturing. These MISs manage the information flow in parallel to resource flow, and establish functional connectivity between resources in different departments, where the resources may be machines and equipment, material, people, and money. The MRPII system is the most versatile of all such MISs.

The following section discusses some of the MISs [17] other than the MRPII system, since the MRPII system has been discussed in detail in earlier sections.

1. Computer-Aided Production Information System (CAPIS)

This system collects practical data through terminals located in work centers with on-line communication through (i) the production *scheduling and control* (SAC) system, to generate an optimum operation schedule, taking into account real shop conditions, which are then judged by the production planner, and (ii) *optimum seeking machining* (OSM), to know about the optimum working conditions of the machines and tools, taking into account the recorded basic engineering data from a file (see Fig. 16). This system, thus, focuses on generating operation schedules, based on machining and tool conditions, which is ultimately judged by people.

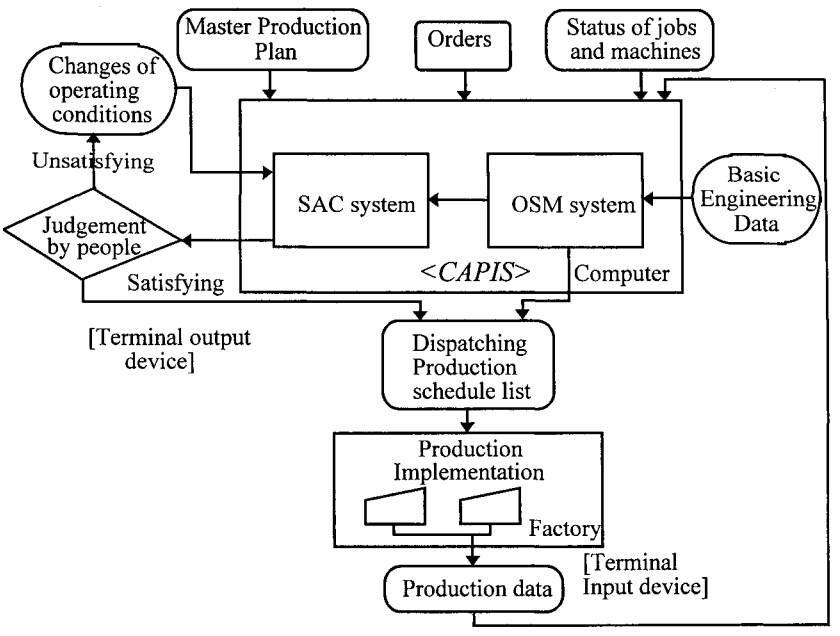


FIGURE 16 A general flow chart of the CAPIS system (Ref. 17, copyright Taylor & Francis).

The SAC performs simulations to find out the optimal schedule, which is then displayed at each work center by means of both visual display and/or printed instruction sheets.

It can be observed that although this on-line computer system is powerful in production scheduling, it lacks power in other functions of manufacturing, such as forecasting, linking forecast to MPS, which are complemented in the recent MRPII system.

2. Parts-Oriented Production Information System (POPIS)

This system is broader and one step closer to the MRPII system, in terms of functionality, than the CAPIS system. It consists of three modules—demand forecasting and production planning subsystem (Module I), parts production subsystem (Module II), and products assembly subsystem (Module III), all of which again consist of several submodules (see Fig. 17) [17].

The first module forecasts demand and sales, which is known as "primary information." It also manages data/information relating to the reception of

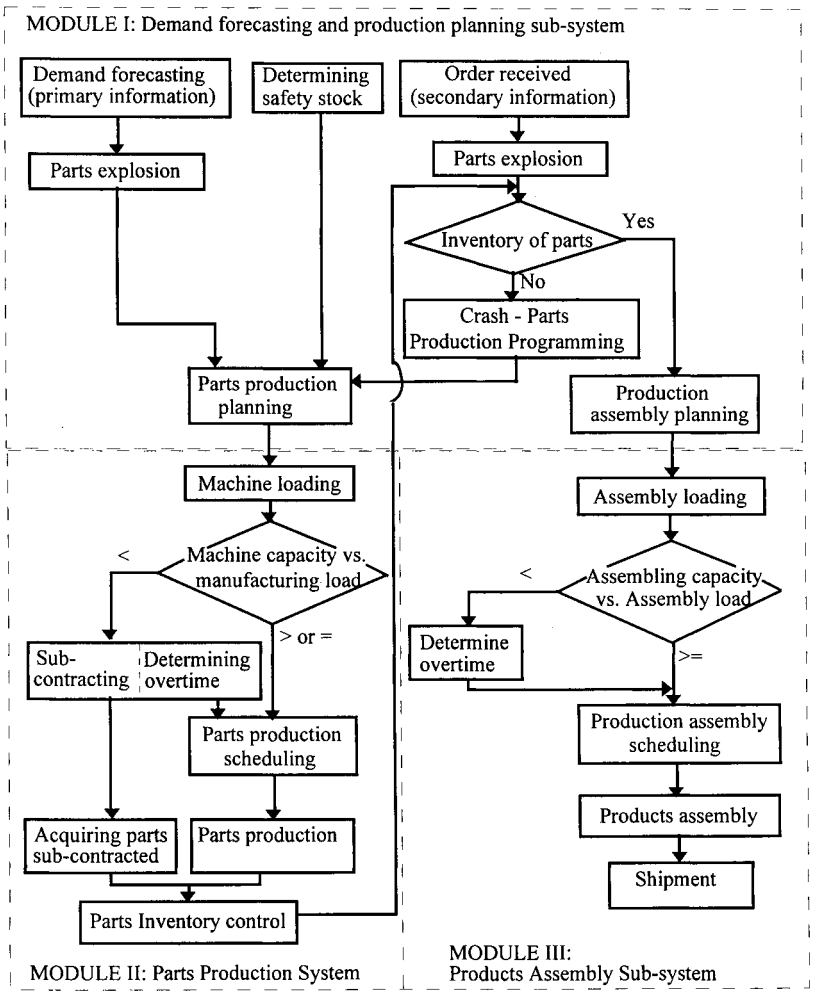


FIGURE 17 General flow chart of POPIS (Ref. 17, copyright Taylor & Francis).

orders, the determination of safety stock, and then the preparation of production plan, periodically, based on the primary information.

The second module, having input from the first module, prepares production schedule for the dependent parts (produced for stock) by properly sequencing them according to priority rules, and then manages the data/information of these produced parts along with the purchased parts as part of inventory control.

The third module, based on the information and data from the first module and materials along with information from the second module, prepares an assembly plan and executes the plan.

Although this computer-based information system has been proven to be useful for resource planning in manufacturing, it has not gained much popularity in today's manufacturing industries.

3. Communications-Oriented Production Information and Control System (COPICS)

This computer-integrated production information and management system was proposed by the International Business Machines Corporation as a concept of *management operating systems* (MOS) [17]. It deals with establishing a production plan based on forecast, and executing the plan as per capacity by purchasing raw materials and shipping the finished goods to customers. This is quite close to the idea of the current MRPII system.

This system introduced the idea of bill of materials of MRPII, from which an explosion is run to find out component requirements (see Fig. 18).

E. Information and Database Systems Centered Around MRPII

On several occasions, MRPII has been used as the center of a company-wide information system, sometimes leading toward the idea of *enterprise resource planning* (ERP), and *supply chain management* (SCM). Additionally, in several cases, the concepts of MRPII have been utilized to build customized information systems for companies. That means, these data and information systems can also be considered as a version of MRPII [13,18,23,25,37, 54].

Although argumental, it can be stated that MRPII evolved toward ERP with the inclusion of several facilities, like electronic commerce and electronic data interchange, and toward supply/demand chain management with the inclusion of a network of companies who trade among themselves, either as a supplier or as a buyer. MRPII plays the central role of managing the data/information of resources in this connection (Fig. 19) [23].

The limitations of the MRPII system, either as an independent system or as a part of a company-wide information system, have also been complemented on several occasions by integrating information systems of other islands of manufacturing (see Fig. 20) [18,25,36].

From a survey, Meyer [25] identified four clusters of information for integration for manufacturing planning. These are: (i) internally oriented administrative systems, e.g., inventory status, purchasing, and accounting, (ii) market-oriented administrative systems, e.g., order entry, sales, forecasting, and distribution, (iii) internally oriented technical control systems, e.g., shop floor

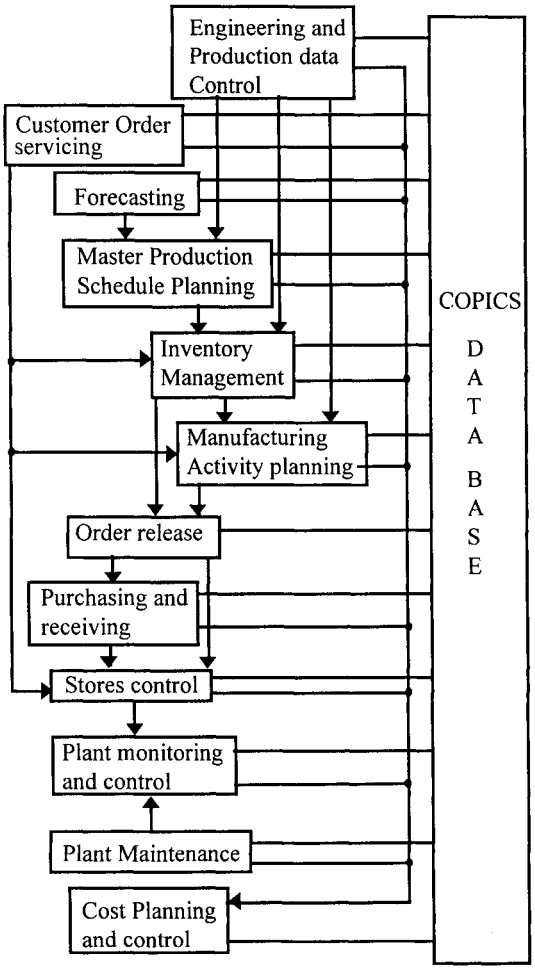


FIGURE 18 COPICS structure (Ref. 17, copyright Taylor & Francis).

control and quality reporting, and (iv) technical systems that link manufacturing to external groups and eventually the customer, e.g., design (CAD) and engineering. It can be noted that out of four clusters of information, three belong to the integrated MRPII system, where as only the last one is beyond the scope of MRPII. Meyer has shown this in a two-dimensional map, as seen in Fig. 21, where the islands of integration are linked with each other through a pivotal database, which is the MPS/MRP.

As opposed to the tradition of interfacing the databases of two systems, Hsu and Skevington [18] formulated a fundamental approach to the information integration of manufacturing enterprises, which includes MRPII as a key provider of data and information for several functions together for an enterprise. This approach entails the use of an intelligent metadatabase, which differs fundamentally from the strategies of both interfacing and a prevailing "super database management system." They applied the "feature" concept, traditionally known to be a technique for CAD/CAM integration, in the case of information modeling for all functions where MRPII takes a major part. This knowledge-

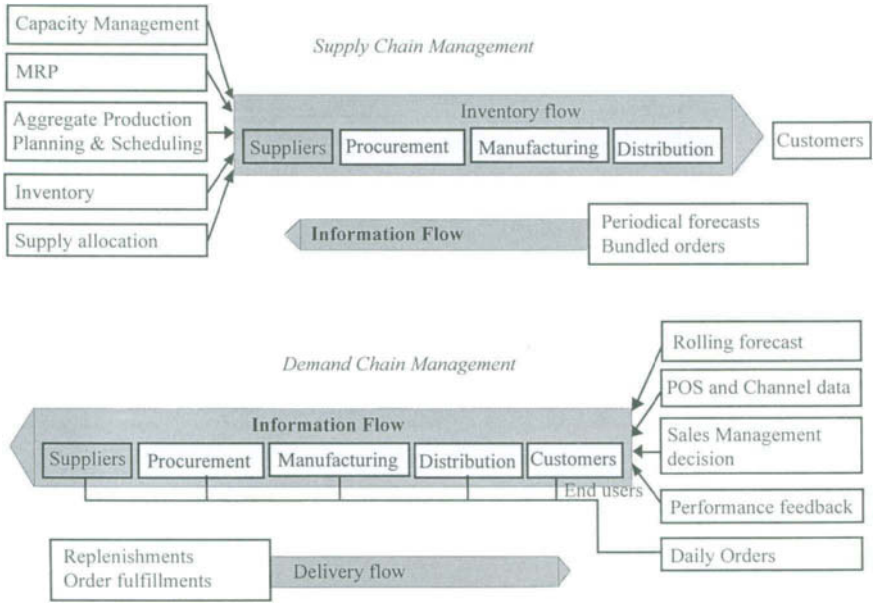


FIGURE 19 Information as driver for supply/demand chain (Ref. 23, copyright Taylor & Francis).

based enterprise information model has been termed as a metadatabase (see Fig. 22).

The authors also proposed an integrated system for computer-integrated manufacturing, specially between CAD/CAM, MRPII, and MRPII-related additional functions. The metadatabase serves as the core of the system residing in the kernel. The functional database, in conjunction with the metadatabase, constitutes the intelligent database system (DBS). The coordination is provided by a hierarchic control system. The application layer provides the system implementation for MRP.

As depicted in Fig. 23, the intelligent metadatabase plays the central role, which serves as an on-line repository of operational intelligence monitoring and supervises the flow of data across the system for real-time information.

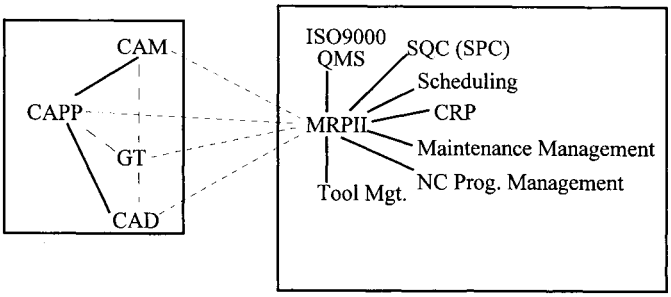


FIGURE 20 Systems Integration around MRPII. SQC, statistical quality control, includes SPC (statistical process control); QMS, quality management system, includes QA (quality assurance); CRP, capacity requirements planning; GT, group technology; CAPP, computer-aided process planning; CAD, computer-aided design; CAM, computer-aided manufacturing (Ref. 36, reproduced with permission from Inderscience Enterprises Limited).

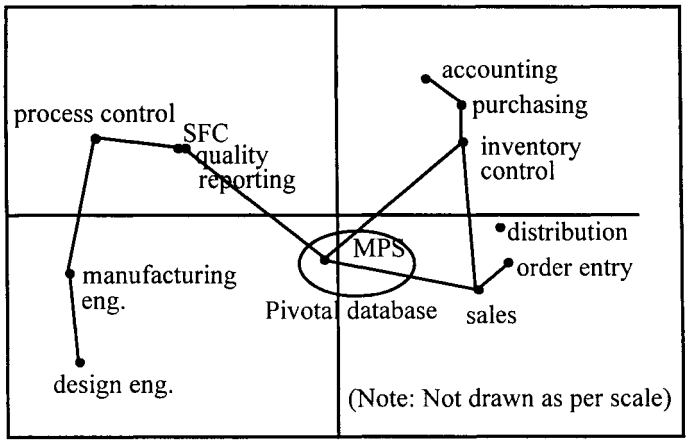


FIGURE 21 Integration around a pivotal database of MPS/MRP [25].

While the subsystems in this integrated database, such MRP, can function independently, its status and directories would be monitored by the metadatabase. The main difference in this system is that the conventional MRPII system's functional entities, such as orders and schedules, are converted to features, and the features and feature-based rules are then mapped onto data and knowledge representations. In this case, an object-oriented (or frame-based) representation for the feature hierarchy and dictionary is developed.

Harhalakis *et al.* [13] have also utilized the MRPII system's large database for further integration, to create an enhanced database system for a company. They are of the opinion that it is necessary to develop and share data from a variety of sources, specially the MRPII system. The authors have proposed the integration of MRPII with computer-aided design (CAD) and computer-aided process planning (CAPP), where they modeled and analyzed the system using Petri nets.

Since integrated information and databases have become the center of research in CIM, the MRPII system has been used to integrate CAD, CAPP,

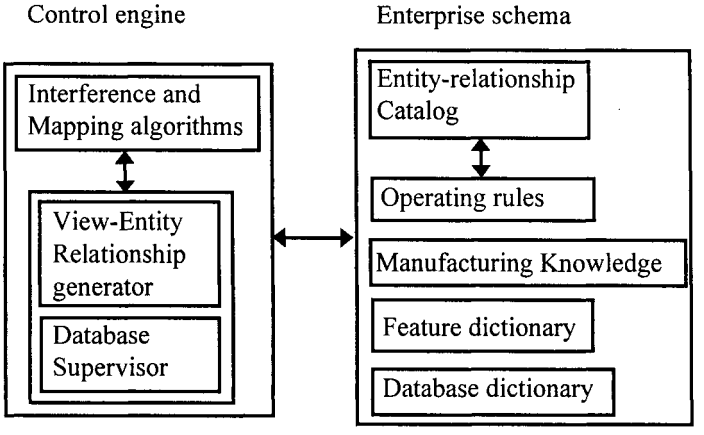


FIGURE 22 The metadatabase [18].

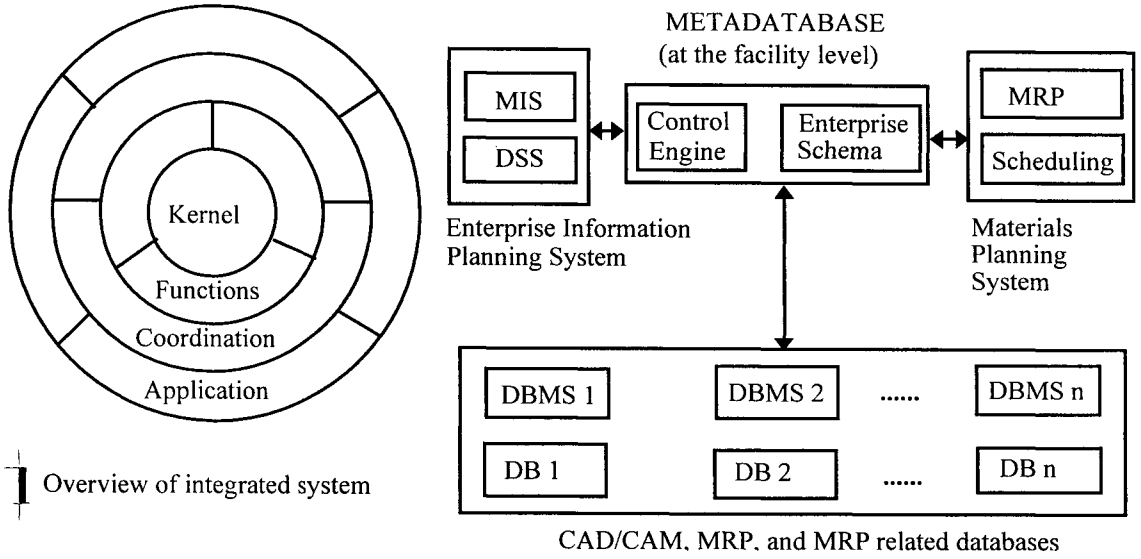


FIGURE 23 Information system for MRPII for integration [18].

TABLE 13 Common Data Records [13]

Part record

CAD	MRPII
Part number	Part number
Drawing number	Drawing number
Drawing size	Drawing size
BOM unit of measure	BOM unit of measure
—	Purchase inventory unit of measure
—	Unit of measure conversion factor
—	Source code
—	Standard cost
—	Lead time
Supersedes part no.	Supersedes part no.
Superseded by part number	Superseded by part number

Part revision record

CAD	MRPII
Part number	Part number
Revision level	Revision level
Effectivity start date	Effectivity start date
Effectivity end date	Effectivity end date
Status code	Status code
Drawing file name	—

Routing record

CAPP	MRPII
Routing number	Routing number
Part description	Part description
Unit of measure	Unit of measure
Operation number	Operation number
Operation description	Operation description
Work center ID no.	Work center ID no.
Setup time	Setup time
Machining time	—
Handling time	—
Run time	Run time
Feed	—
Speed	—
Depth of cut	—
Number of passes	—
—	Resource code
—	Begin date
—	End date
Status code	Status code

Work center record

MRPII	CAPP
ID number	ID number
Description	Description
Department	Department
Capacity (HR)	—
Rate code	—
Resource capacity	—
Dispatch horizon	—
Effectivity start date	Effectivity start date
Effectivity end date	Effectivity end date
Status code	Status code
—	Horse power
—	Speed range
—	Feed range
—	Work envelope
—	Accuracy
—	Tool change time
—	Feed change time
—	Speed change time
—	Table rotation time
—	Tool adjustment time
—	Rapid traverse rate

and MRPII through different DBMS approaches, as outlined by Harhalakis *et al.* [13]. As stated by the authors, the functional model of the proposed MRPII/BCAD/CAPP integrated system is based on the similarity of functions and the commonality of data among these three application modules. More specifically, Table 13 lists the data records the authors found common to the information subsystems [13].

To represent the functioning of the model, status codes for each entity are used to control the information flow and the status changes. The entities in question are part revision, routing, and work center. These are listed in Table 14.

TABLE 14 Status Codes [13]

Part revision status codes in MRPII, CAD, and CAPP	
MRPII	
R—"Released"	An active part, whose purchase or manufacture can be initiated in MRPII.
H—"Hold"	Under review, not to be used by MRPII.
CAD	
W—"Working"	At a conceptual or preliminary stage, prior to approval, and not transmittable to MRPII.
R—"Released"	An active part, whose design has been finalized and approved.
H—"Hold"	Under review, pending approval, possibly with a new revision level. The part should not be used by any system.
O—"Obsolete"	The part is obsolete.
CAPP	
W—"Working"	At a conceptual or preliminary stage, prior to approval.
R—"Released"	An active part, whose design has been finalized and approved.
H—"Hold"	Under review, pending approval, possibly with a new revision level.
O—"Obsolete"	The part is obsolete.
Routing status codes in MRPII and CAPP	
MRPII	
R—"Released"	An active routing, which is able to be passed down to the shop floor by MRPII.
H—"Hold"	Under review, not to be used by MRPII.
CAPP	
W—"Working"	At a conceptual or preliminary stage, prior to approval, and not transmittable to MRPII.
R—"Released"	An active routing, whose process design has been finalized and approved.
H—"Hold"	Under review, pending approval, possibly with a new revision level. The routing should not be used by any system.
O—"Obsolete"	The routing is obsolete.
Work center status codes in MRPII and CAPP	
MRPII	
R—"Released"	An active work center.
H—"Hold"	Not to be used by MRPII.
D—"Delete"	A work center deleted from the system.
CAPP	
W—"Working"	At a preliminary stage, work center details need to be entered in CAPP.
R—"Released"	An active work center, able to be used in process plans.
H—"Hold"	Under review, not to be used for process plans.
O—"Obsolete"	The work center is obsolete.

According to the authors [13], the Petri net theory has distinct advantages that enabled them to model an integrated database system around MRPII. The advantages are listed as: (i) its ability to model simultaneous events, (ii) its ability to represent conflicts, (iii) its potentiality to identify the status codes, using Petri net graphs, and (iv) its hierarchical modeling capability.

IV. APPLICATION OF RELATIONAL DATABASE MANAGEMENT TECHNIQUE IN MRPII

The past two decades have witnessed phenomenal change in the field of industrial engineering systems, which have expanded to include computer-based information. One tool regarding this is the *database management systems* (DBMS), particularly, relational DBMS. The relational data structure supported by a software is easier to conceptualize and manage some alternate data structures, and it allows greater flexibility in defining the data and their relationships [20].

Over the years, relational database techniques, as a tabular form, have been applied, and still now, are the most widely used in the MRPII system and its modules' design. The majority of existing commercial MRPII systems follows the relational approach. There have been reports of design efforts of MRPII and its modules using relational databases [29,35,54]. It must be stated that generating and managing, efficiently, the database of BOM is possibly the most difficult task among all the modules of MRPII. As such, the design of this module has attracted considerable attention from researchers.

Because of the large variety of products, which require several kinds of numerous resources for production, processing and response in a MRPII system become so high that ad hoc decision facilities become necessary, in place of the current situation where processing is done at a full level of detail all the time. The possibilities of abstraction and decomposition in a materials planning system, such as MRPII, becomes inevitable. Winter [54] proposed that a combination of metaplanning concepts with representation and processing capabilities provided by a relational DBMS is suitable for resolving the stated issue. Since a recent relational DBMS is capable of representing complex abstraction hierarchies, such databases can serve as an intelligent tool for the integration of different levels of information abstraction.

Hierarchical materials planning systems require the concepts of object abstraction. Abstraction can be applied to any object, such as machines, parts, or orders. The abstraction has five dimensions: aggregation, generalization, association, classification, and selection. A typical abstraction hierarchy is shown in Fig. 24 [54].

Winter proposed that objects, such as machines, manpower, and orders, as well as procedures that compute, govern, and link the objects, such as MPS, offsetting, and lot sizing, can be represented by data abstraction, and then represented by a relational view.

Although Winter [54] did not elaborate on the construction of abstract objects and their linkage methods through relational database management techniques, he presented the objects and procedures to show that they can be linked with relational techniques using different views of abstraction dimensions, such as aggregation, generalization, and selection.

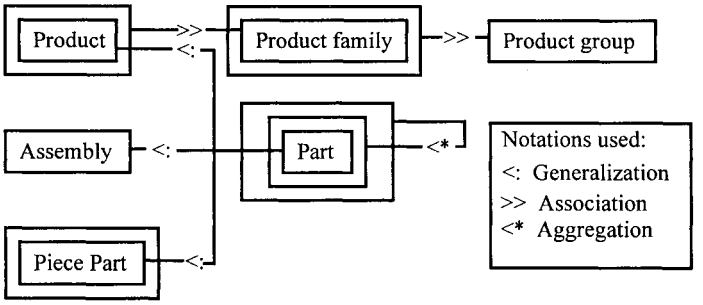


FIGURE 24 An overview of an abstraction hierarchy for a product structure (Ref. 54, reproduced with permission).

It is possible to represent all dimensions of abstraction in terms of relational views referring to physical relational tables or to other relational views. In order to explain this, it is assumed that piece parts are those parts, consumed by the production process, but are never outputs from a process (e.g., purchased parts, raw materials), subassemblies are those parts outputted from a process, but not the final product, and products are those parts not consumed by a process, but rather are the final outputs of a process. Figure 25 illustrates the aggregation view of subassembly, which is based on the "process" table, "BOM" table, and "piece part" table.

Figure 25 is graphically self-explanatory. The part ID of subassembly is obtained from that of the assembly process. The maximum lead times for the components (piece parts) required to build the subassembly are added together to find the lead time for a subassembly. The values of the components are multiplied by respective quantities to find the value of their subassembly. ABC classification of a subassembly is the minimum ABC position of its components [54].

The final product and its components can also be defined similarly. This way a complete aggregation hierarchy of BOM can be defined. Thus, it can be

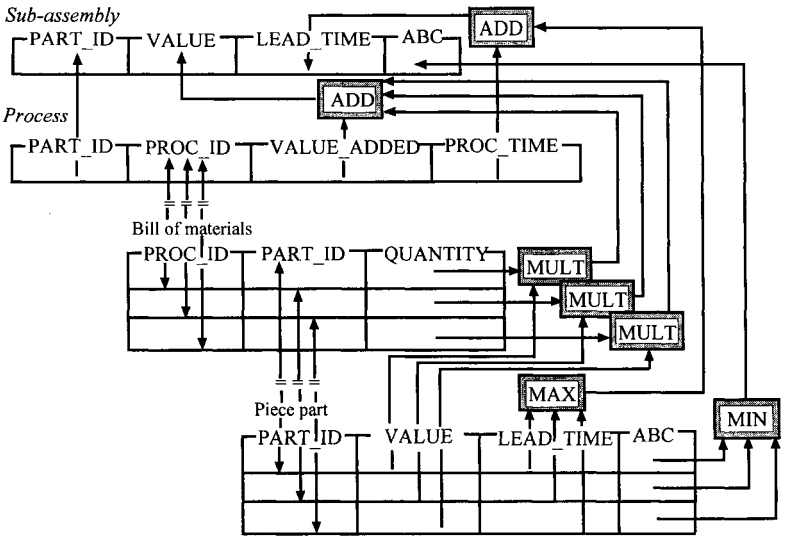


FIGURE 25 Aggregation view of subassembly (Ref. 54, reproduced with permission).

concluded that abstractions (of objects) can be represented as a relational view. Similarly, abstraction is possible in the case of many planning procedures.

Based on the fact that whenever a planning procedure can be represented by a series of relational operations that procedure can also be represented by a relational view, the Winter [54] illustrated how some MRP procedures can be represented as a relational view. The following discussion shows the netting procedure (which determines net requirements of materials or inventory) with a relational view.

A netting procedure requires a set of sequential operations (computations) using data from separate relational tables. Separate data tables for inventory and gross requirement can be created to perform those sequential computations (Fig. 26). In MRPII, net requirement represents the difference of amounts between current inventory on-hand (from the "inventory" table) and the gross requirement (from the "gross requirement" table). This has been shown as a relational view in Fig. 26. For each part, using an appropriate view, cumulative gross requirements should be found based on the gross requirements table. The amount of "inventory out" should be based on the on-hand inventory and cumulative gross requirement, whichever is least. (That is, if gross requirement is more than on-hand, an amount equal to gross requirement cannot be issued. In that case an amount equal to on-hand can be issued. On the other hand, if the on-hand amount is more than the cumulative gross requirement, then certainly an amount equal to the cumulative gross requirement should be released.) Whatever the abstraction level of parts, gross requirements, and inventory, the

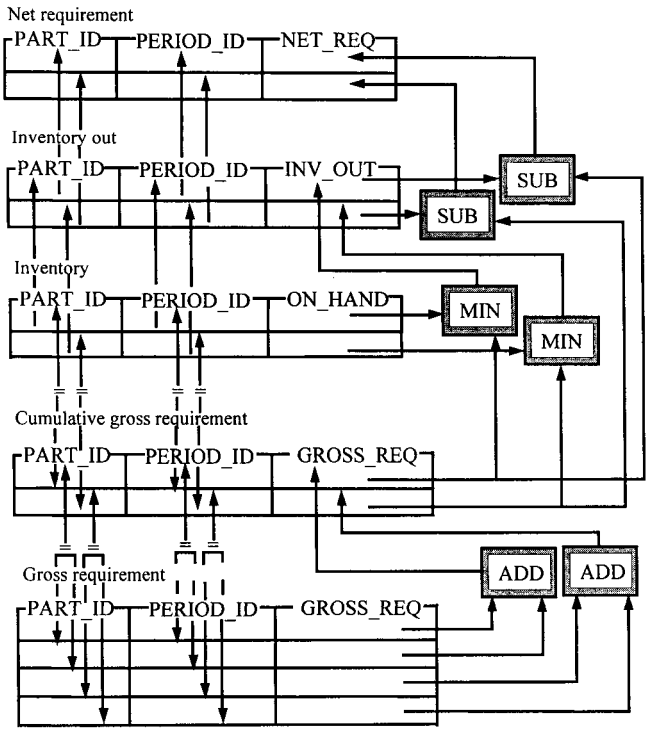


FIGURE 26 Relational view of net requirement computation (Ref. 54, reproduced with permission).

netting procedure is always feasible, and leads to net requirements at the same level of abstraction [54].

Similarly, several other MRPII procedures such as lot sizing, MRP explosion, and lead time offsetting, can also be represented in a relational view. Finally, a relational database system integrates the representation of abstraction hierarchies and planning procedures [54].

A prototype system, having database definition and processing in the relational DBMS Oracle V5. 1B, its SQL interface Oracle SQL*PLUS V2.1, and its C interface Oracle PRO*C V1.2, was developed. This system provides the facilities of normal materials planning, and additionally reusability and simulation. It is true that some of the complex planning procedures cannot be represented solely as a sequence of relational operations. Those were implemented in C programs. Database functions and relational operations were embedded in these procedural programs as high-level SQL commands. All base tables are loaded/unloaded via import/export procedures from a host-based MRP system that supports a hierarchical database model. The authors hoped that despite several drawbacks, continuing development of relational DBMS and the integration of database representation techniques and inferential processing will allow the application of the above concepts to more complex planning systems' design at a reasonable level of performance [54].

Olsen *et al.* [35] described the use of this approach in the case of customer-oriented products, which need a large number of variants. As a solution to the problem of managing a large number of product variants, if separate BOMs are created for each variant, the database would become so large that it would be impossible (and/or inefficient) to manage in the computer system. As a result, the authors suggested a generic structure of BOM. They designed this generic BOM using a relational database technique, but their approach is different from the traditional method of generic BOM. Earlier attempts to develop generic BOM have been directed toward either creating planning BOMs or improving the tabular structures only. As this technique has several limitations, including inflexibility, the authors proposed resolving it by the application of programming language notation [35], although there has been a report of the use of the object-oriented technique for this [7,8]. The authors, however, disagreed with the idea of object-oriented techniques for this purpose as it violates one requirement of describing components independently of their utilization.

Olsen *et al.* [35] proposed building generic BOM by utilizing some constructs, namely procedure concept, variables, input concept, and selection (case) statement, of programming languages. This has been termed as a procedure-oriented approach. In this approach, the variant or customer-specific product specification is established through input from the user. A prototype was tested on a PC under MS Windows. The programs were implemented with SQL Windows case tools from the Gupta Corporation.

The description of the generic BOM (GBOM) as a program enables the system to support the user in the product variant specification task by defining each component with header and body, which is stored as a database record. Variants are expressed simply through attributes.

Suppose that a "stool" has a component "seat," which may have three options of colors, namely red, blue, and white. The followings are the codes

proposed by Olsen *et al.* [35].

```
component §400 is
  name("seat");
  seatColor(red|blue|white)
end component;
```

```
component §200 is
  name("stool");
end component;
body §200 is
  include §400;
  include §500;
end body;
```

In the above codes, a *procedure,* "component" has been illustrated. While the head identifies a component and presents its attributes, the body presents the goes-into relationships, i.e., BOM structure. Here, the head part of the component "seat" has a single attribute seatColor, which gives optional views of the component, i.e., may have different colors of red, blue, or white, based upon the user's selection. The other codes show it as a part of "stool" (ID: 200). The body part of the "seat" may be defined as shown in Fig. 27 [35].

In the Figure, the head part of codes for only cover (ID: 450) has been shown, although the seat is composed of cushion, chipboard, and cover. The heads of other components can be defined similarly. The codes show that the seat color may vary, depending upon the options of colors of cover, which may be selected by the user during variant creation. The attribute coverColor is, thus, used as a specification of the product variant. Further specifications with options in part features can be added, such as options of different seat sizes. The generic view of the complete procedure to create a specific BOM from the generic one is shown in Fig. 28 [35].

The system, in Fig. 28, has three tasks, Generic BOM Task (GBST), Product Variant Specification Task (PVST), and BOM Conversion Task (BCT). The GBST specifies the generic BOM for one or more products. At the stage of PVST, the attributes for a specific variant are given, in order to create a specific BOM for a particular product. At the step of BCT, each component variant referenced in the ABOM is replaced with a unique item number. The translation tables (TTAB) contain the necessary information for conversion from attribute values to component numbers [35].

The data structures, output from these three tasks, are: (i) The GBOM describes all components used in a BOM; (ii) ABOM describes a specific product variant; and (iii) NBOM describes the BOM in terms of indented BOM for use in a conventional information system, such as for use in commercial MRPII systems [35].

```
body §400 is
  include §410;  ---- cushion
  include §420;  ---- chipboard
  include §450 with;  ---- cover
    coverColor(seatColor);
  end include;
end body;
```

```
component §450 is
  name("cover")
coverColor(red|blue|white);
end component;
```

FIGURE 27 A GBOM view [35].

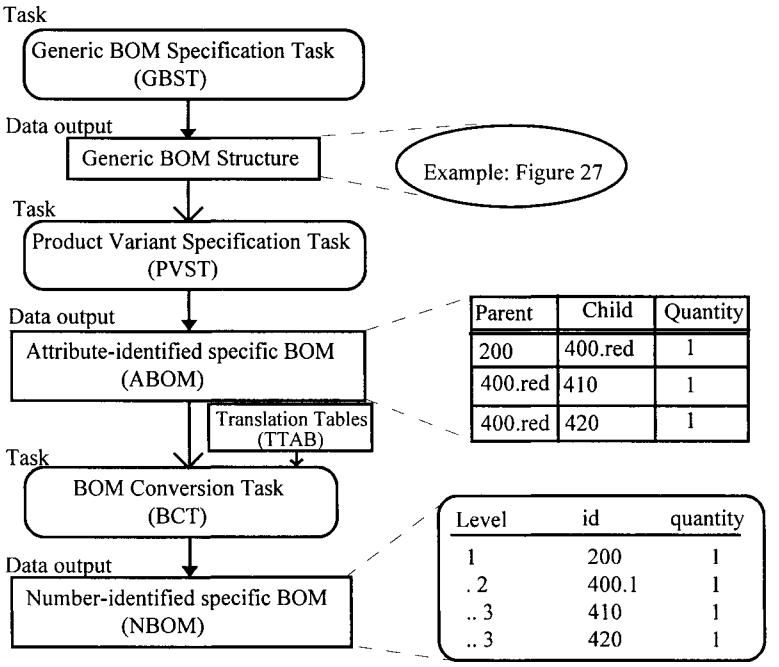


FIGURE 28 Procedure-oriented BOM generation [35].

V. APPLICATIONS OF OBJECT-ORIENTED TECHNIQUES IN MRPII

Many authors have discussed the weaknesses of traditional forms of database management techniques for MRPII and other similar systems. They are of the opinion that object technology may be helpful for solving many of the data management problems in complex manufacturing systems, composed of several planning functions, although it is fact that object technology also has some limitations [7,8,10–12,19,21,22,30,35,37,41,44,49,56–58].

As an MRPII system is composed of several planning and control functions, integration inside the MRPII system and its periphery for third party data connection is a vital requirement, and of interest too, as it contains a large volume of complex data. Each of these functions serves as an application module inside the system. It requires coordinated solutions to data management problems for not only individual application modules, but also for intermodule data exchange. The object paradigm, in many cases, has brought quite considerable success.

Out of several limitations of the traditional data management, a lack of data semantics is of particular importance. Thus, developing techniques for representing and using the semantics of data is key to realizing improvements in data management inside the MRPII system, and its integration with other manufacturing applications, such as quality management functions and maintenance scheduling.

Spooner *et al.* [47] discuss the research work at the Rensselaer Polytechnic Institute, which addressed the above problems with hybrid types of solutions. As the authors discuss, the desire to capture more of the semantics of data leads naturally to the use of data abstraction and object-oriented techniques to

model and integrate data in a CIM environment. The authors present CIM as a system composed of design and engineering data and some production-related data, which is in fact an integrated MRPII system. Interestingly, the authors prescribe the use of object technology in the case of the first type of data, and the relational data technique for the MRPII-related functions. The logic behind this is explained:

> Data abstraction provides a means for organizing data into objects that are semantically meaningful to application systems. Object Oriented programming techniques enhance the semantic content of data by defining relationships, possibly with inheritance of properties for CIM applications [the authors here mean the engineering and design applications] because the fundamental principles behind it are the creation and manipulation of data objects, and these are exactly the activities done during typical engineering, design and manufacturing processes. The object paradigm also provides a flexible data model in which the complex data structures found in many design and manufacturing applications can be created and manipulated easily. This flexibility also helps in integration of the many applications in a typical CIM environment. [47, p. 144]

At the same time, the authors were of the opinion that other aspects of CIM, arguably the MRPII functions, having quite similar data, fit into the realm of traditional relational database technology. As a result, the CIM database management system was shown as a hybrid system, comprising object technology and relational data technology. This hybrid system was proposed as the ROSE Data Manager. The framework of this system, as shown in Fig. 29, contains a distributed object management component, which can be used to transfer data between an object database (design, engineering, etc.) and a relational database (MRPII functions).

In majority of the cases, the MRPII systems were built using the relational database management systems, although it has shortcomings in processing long-duration transactions, in storing images or large texture items, and in having complex data structures. Currently, the research direction in database management systems of MRPII (and other manufacturing systems, such as CAD and CAM) has moved from the relational database toward the object-oriented

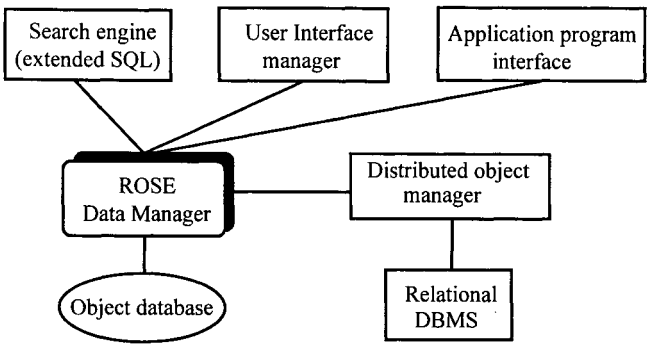


FIGURE 29 A hybrid DBMS for CIM, including MRPII [47].

database, which is capable of supporting complex data structures and network transactions [10].

Although *object-oriented database management systems* (OODMS) have proven to be very fruitful in many areas of MRPII, it has its own limitations as well. As discussed in Section II of this chapter, an ad hoc query is very essential in MRPII. As discussed by Du and Wolfe [10], it is difficult to declare a query in an OODMS. To query an object-oriented database (OOD), the user needs to navigate from one object to another through links. Also, so as not to violate the rule of encapsulation of object-oriented technology (OOT), the only way to access data is through designated methods in objects. Some commercial OODMSs, such as O2, ONTOS, ORION, Iris, and POSTGRES, support this ad hoc query facility, whereas others, such as Versant and Gemstone, do not possess that capability [10].

Zhou *et al.* [56] discuss the use of object technologies for information management in production planning and control (PPC). Although production planning and control do not exactly mean MRPII, they play a significant role in MRPII. Thus, on many occasions, the MRPII is termed as being similar to a production planning and control system. The authors present a distributed information management architecture, written in Visual C++ on a Windows NT platform, for PPC in a semiconductor manufacturing virtual enterprise, under the ESPRIT project (X-CITTIC). The Oracle 7 workgroup server has been used to build the databases, Iona's Orbix as ORB to link distributed information managers and applications, and ILOG for building the object servers, linking relational databases, and servers and building GUIs.

The authors describe some information/data, e.g., resource data, as local data, and other, such as customer orders, as global data. In this system every virtual enterprise unit connects to a local information manager (LIM), which holds all local data and serves local applications (e.g., local capacity models). Some units of the virtual enterprise may share one LIM. The authors cite an example: front end (FE) and back end (BE) wafer fabrications often share one production site and one shop floor control system and even some production resources. The global information manager (GIM), located at head quarters, maintains global data/information and is used by global applications (e.g., enterprise planner and controller). The communications between the information managers are performed using *object request broker* (ORB) with Internet inter-ORB protocol (IIOP) capability (Fig. 30).

Although of particular importance for further research on application of object-oriented technology in MRPII and PPC, the authors, however, do not elaborate how each of the elements of PPC were modeled using object technology, and how the functions of PPC (or modules of MRPII) interact with each other.

Park *et al.* [37] also report on a similar work on object-oriented production planning (aggregate) and control (PPC), as part of the *enterprise resource planning* (ERP) system. They present an object class extraction system in several steps through: (i) defining the object classes within the system using the Use Case approach, (ii) establishing relationships among the classes, (iii) establishing a hierarchical structure of classes, (iv) establishing event trace diagram, and (v) defining the detail attributes and operations about the object classes, although the last step is left for future work. The functions, which have been

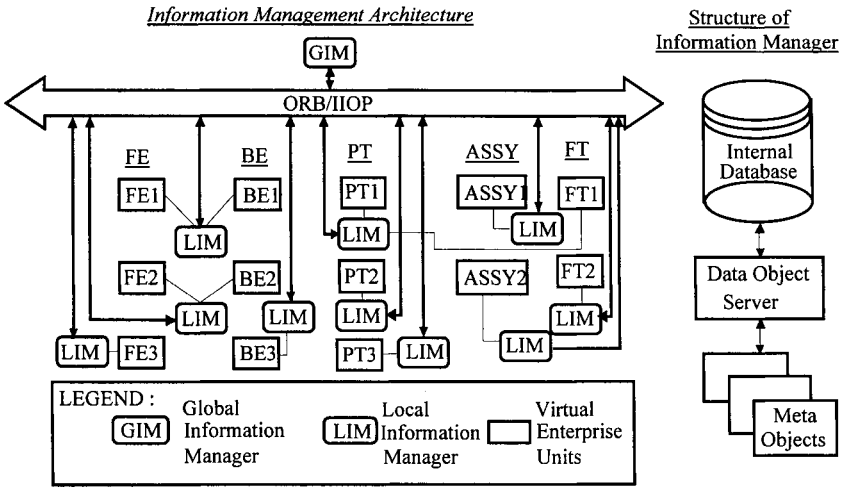


FIGURE 30 Virtual enterprise information management architecture [56].

covered in PPC systems, are scheduling and rescheduling, capacity requirements planning, inventory control at the factory level, dispatching, load monitoring, and process monitoring. The PPC system has been classified into the following object categories:

Production objects
- At factory level: factory level scheduling object, factory level rescheduling object, capacity requirements planning object, inventory control, factory level load monitoring object, dispatching object
- At shop level: release control object, shop level load monitoring object, shop level scheduling object, shop level rescheduling object

Database interface object
- Oracle DB interface object, Informix DB interface object, file interface object

User interface object as a database query translator
- process progress indicator object
- event generating object by user input data

In this study, the authors [37] further classify the PPC-related object classes into the following groups:

Object class group associated with production plan function
- manufacturing control object classes
- manufacturing control interface object classes

Object class group associated with production plan information
- manufacturing data object class
- manufacturing data interface object class
- database interface object classes

Object class group associated with user interface
- user interface object classes

Work flow object class group object class
manufacturing work flow object class
manufacturing work flow engine object class
event handler and equipment monitoring object classes

The above object classes and groups are shown in Fig. 31. The complete architecture was designed using the object patterns listed in Table 15.

Once the classes are defined and established, they can be extracted using the steps stated earlier. The authors used the Use Cases approach using the SELECT case tool for the process. The manufacturing control interface (MCI) object group has been classified into OrderManagement, LongTermPlanning, MidTermPlanning, ShortTermPlanning, RealTimeScheduling, PurchasingControl, InventoryControl, PcgInputOutputControl, and MfgInputOutputControl classes, the example of the extracted OrderManagement class with their interfaces being given in Table 16.

However, the details of how individual object classes, for instance, the components of the BOM, are defined and the relationships among the components are established; how one class is related to other classes for interdependent planning, for instance, how the MPS is linked to BOM, and how a process like MRP explosion takes place, are not explained [37].

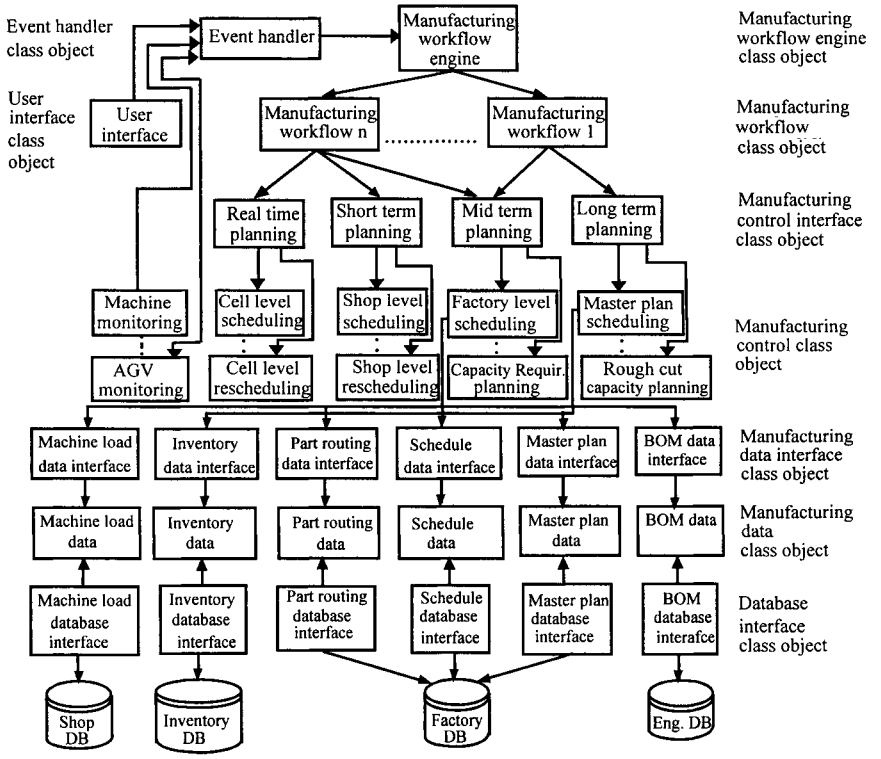


FIGURE 31 Object classes [37].

TABLE 15 Features of Patterns [37]

Pattern	Domain	Solutions
Functional object class pattern	Function-oriented system which is hard to extract objects	The individual function is regarded as a object class. The operations of the function are mapped into the member functions of object class.
Interface object class pattern	A various of objects needed for accessing other objects	Member functions generate interface object classes, calling for other objects consistently. As an example, work flow accesses other multiple objects using interface objects.
Multiple-strategy pattern	Algorithms utilized for object class member function	If one or more membership functions can be implemented using another type of algorithm, the abstracted object class is generated and constructs an object class including another type of algorithm under the abstract object class member functions. The work flows are accessing other multiple objects using interface objects.
Chained strategy pattern (structural pattern)	Algorithm utilized for object class member function generated through the multi strategy pattern	If the previously generated member functions of object class can be implemented using the multiple-strategy pattern, the objects are connected with the supplementary algorithm in the object class.

Ng and Ip [58] report on a case where the *enterprise resource planning* (ERP) system has been integrated with the real-time monitoring system (RTMS), in order to facilitate real-time data collection for planning and scheduling of manufacturing orders, etc., which, in turn, enhance abilities in the areas of production and material control, shop floor control, warehouse planning and control and other logistics functions. This is an enhanced MRPII system. The system was designed and implemented with a *distributed object-oriented technology* (DOOT), which increases system scalability, configurability, flexibility, reusability, and interoperability. For data collection, hard automation using sensors, bar code readers, *programmable logic controllers* (PLC), etc., were used. All relevant real-time shop floor data can be transferred to the ERP system in office bidirectionally. The schematic of the integrated system is shown in Fig. 32.

According to the authors, the ERP represents the application of newer information technology, which includes the move toward relational database

TABLE 16 OrderManagement MCI Class [37]

	Interface	MC class	Interface
OrderManagement	ExecOrderAccept( )	OrderAcception	OrderDataAccept(OrderData)
	ExecOrderCheck( )	OrderAvailabilityCheck	OrderDataCheck(OrderData)
	ExecDuedateEstimate( )	DuedataEstimation	OrderDataEstimate(OrderData)
	ExecOrderSequence( )	OrderSequencing	OrderDataSequence(OrderData)

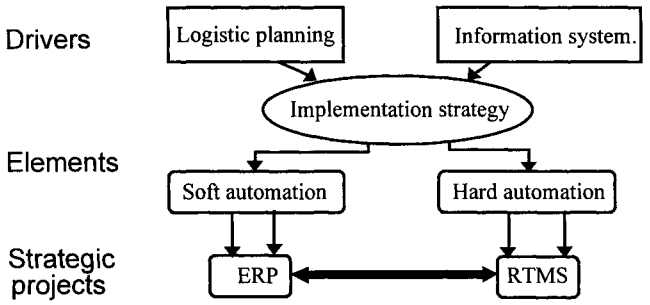


FIGURE 32 Integrated ERP system [58].

management systems (RDBMS), the use of a graphical user interface (GUI), open systems, and client-server architecture, to the MRPII model. The DOOT system architecture, presented by the authors, is divided into four domains: (i) presentation domain—it concentrates on the interaction with the user and managing the user interface; (ii) reporting domain—it is responsible for preparing reports, graphs, or queries; (iii) logic domain—it focuses on managing corporate resources (e.g., database tables) that users share, in particular the application logic; and (iv) database domain—it is a relational database server (e.g., Oracle, Informix, etc.) that manages the persistent storage of data; only the Logic domain can interact with the database domain. The presentation, reporting, logic, and database domains can be placed in different platforms and/or machines in a configurable manner, as shown in Figs. 32 and 33.

In Fig. 33, the *business object* is the conceptual image of the job-related object that the user deals with everyday. The model has two distinct layers: (i) the real object layer defines the user's conceptual view and belongs to the presentation domain, and (ii) the application layer defines the corporate resources view and belongs to the logic domain. The *business object model,* as defined by the object management group, represents the real-world business entities required in planning, for instance, a customer, a manufacturing or purchasing order, a finished good, or subassemblies. The application model is established as the *entity/relationship* (ER) model that defines business entities, their relation ships, and attributes that preserve the integrity of the corporate information resources. The Business Object Model represents the user's view of the data

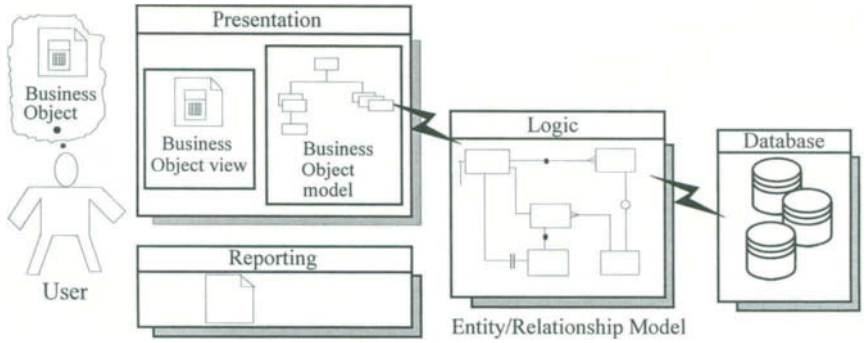


FIGURE 33 ERP system architecture [58].

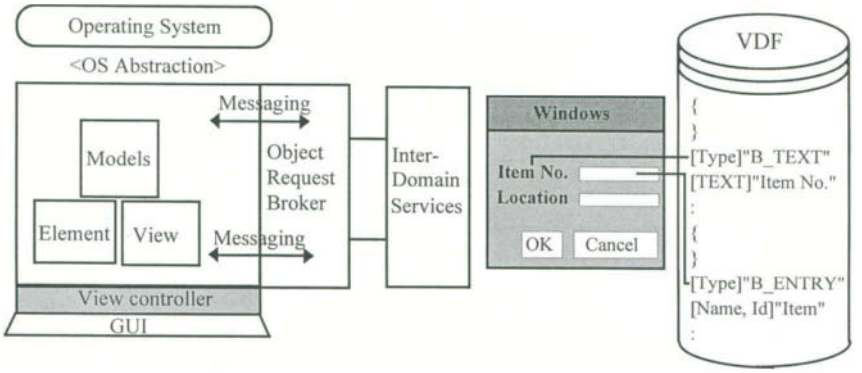


FIGURE 34 Presentation domain and VDF [58].

representation, and only a part of the overall E/R model, which a user deals with in the ERP system. Conceptually, the business object is a hierarchical composite object that has a root subject entity. For instance, a purchase order of an ERP/MRPII system is a form created and filled out from the purchase request, with associated information of a header, item names and IDs, vendor name, price, etc., whereas an E/R relationship model would show it as a combination of several entities interacting with it [58].

As Ng and Ip [58] explain, the presentation domain, responsible for managing the user interface and related matters, consists of three components, namely, *business object models, business object views,* and *business object elements*. The model, as the central control point of a business object, represents a hierarchical data collection, with a root object that identifies the business object. It is responsible for managing the data cache, requesting data from servers, updating the servers against transactions, and interacting with the user through the business object view. Each E/R represented in the business object model is implemented as a server object in the logic domain. The business object view is responsible for displaying the information, using GUI windows, as per user input/output through peripherals, without involving the server. The view layout is defined in ASCII format in a file, termed as the *view definition file* (VDF), as shown in Fig. 34.

The functions of the logic domain are to manage corporate data storage and retrieval, trigger event-driven processes, and provide program logic for the business operations. Its components are entities, relationships, transactions, and databases (Fig. 35). In this logic model, the main target is to establish the *entities object* and their *relationships object*. The hierarchical object model is shown in Fig. 36. The relationships manager in the logic domain is responsible for spreading the event-driven business processes, providing query methods for listing members of the relationship, assigning sequential keys to entity by querying the database, and resolving concurrency lock and registering interest. The transaction manager is responsible for managing the activities of the database, including data updating and consistency [58].

The reporting domain is responsible for presenting reports, graphs, and queries to the user on screen and printer through the Presenter, communicating

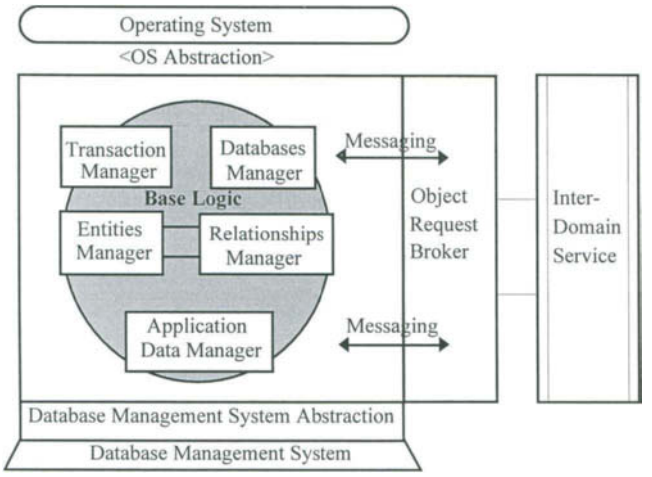


FIGURE 35 Logic domain [58].

with the logic domain through the database object to call the DBMS in order to perform SQL operations through the Extractor, and analyzing, calculating, and sorting operations for a report, and assimilating data.

A. Object-Oriented MRPII Modules

Development of object-oriented systems for individual modules/functions of MRPII system have been reported by several authors [7,8,11,12,44, 49].

B. Object-Oriented Inventory Control System

Sinha and Chen [44] have suggested the object-oriented inventory control system using the hierarchical inventory management model, introduced by Browne [4]. The multilevel (inventory control) system has been designed utilizing the

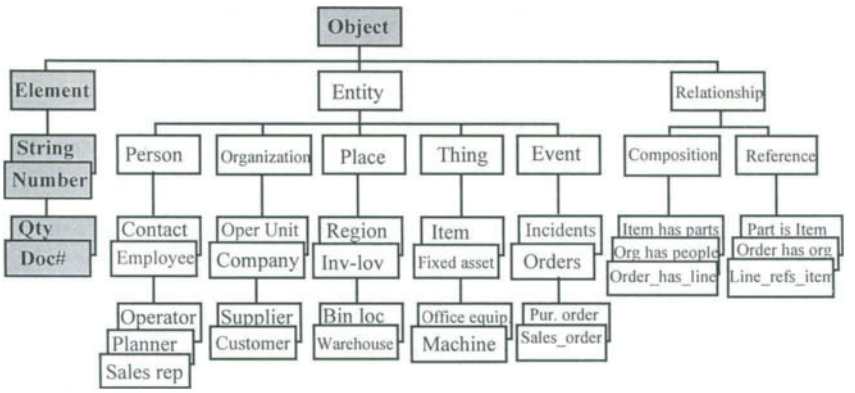


FIGURE 36 Hierarchical object model [58].

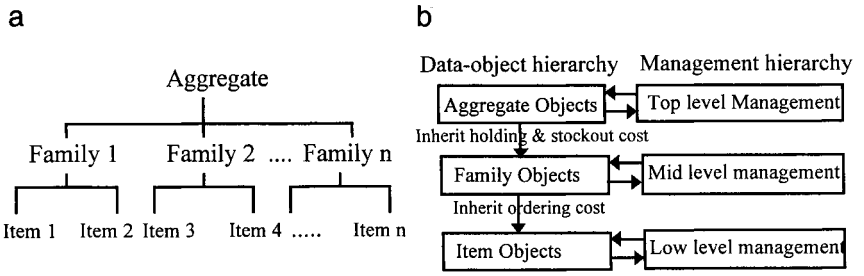


FIGURE 37 Object-oriented hierarchical inventory system: (a) inventory hierarchy and (b) inventory data-object hierarchy [44].

hierarchical parent–child inheritance relation of the object-oriented approach. In this hierarchical inventory system, it has been found that in a company, several similar individual inventoried Items may belong to a Family, and the Families, in turn, belong to the overall company's Inventory (termed here as Aggregate) (Fig. 37a).

The top level, which is decided by the top management to aim at the corporate financial and marketing goals, decides on inventory holding costs and stockouts, depending upon the level of customer service aimed at, which are the company's management policy variables. While decision analysis may be based upon sample inventory, the population consists of all inventoried items to form the object Aggregate. The second level of the hierarchy addresses the issues of inventory Family, which is a group of similar items based on ordering costs (and other characteristics when necessary). Decisions, such as order quantities, at this level may be based on an exchange curve, which is constructed using classical inventory parameters, the holding cost and ordering cost, in order to achieve an optimal point. The analysis may be based on a sample from the population, and then extended to the population to decide among others the ordering cost. The lowest level is the ultimate execution level to decide individual Item optimal quantities (may be EOQ), by inheriting the ordering cost, holding cost, and stockout [44].

The inheritance facility of object-oriented systems permits upper echelons to build on constraints to be observed by the lower levels. An example of Item and Family classes are shown below [44]:

```
Class Family
  Attributes: Order_Cost, Holding_Cost,& others
End Family
Class Item inherit family
  Attributes: Unit_Cost, Demand, and other
  Function Inventory_Cost (X)
    Inventory_Cost = (X/2)*Unit_Cost*Hold_Cost +
    (Demand*365/X)*Order_Cost
  End Inventory_Cost
End Item
```

In the above case, although the ordering and holding costs are not exported

by Class Family, they are available within Item. Here, X represents the order quantity of an item.

It is observed that this object orientation in inventory eliminates the duplication of data storage of ordering and holding costs for individual items, as they are stored under Family or Aggregate. The authors have also developed, as a user interface, different objects for this inventory control system. These are system level objects, such as Print, Save_List, and Graph (Cost curve), and user level objects, which are described above (e.g., Item, Family, Aggregate [44]. From the success of this system, it is suggested that benefits of inheritance of object orientation can be utilized to develop other subsystems of MRPII, where many things are hierarchical in nature, e.g., (i) BOM, (ii) Aggregate planning and MPS, (iii) shop level, family level, and work center level scheduling, and (iv) capacity planning.

C. Object-Oriented PAC System

Reports also reveal that object-oriented systems for production activity control (PAC), or alternatively known as shop floor control, subsystem of the overall resource planning system, have also been developed.

Gausemeier *et al.* [11] report a case of successful development and implementation of such an object-oriented PAC system, integrated with other modules of MRPII. In this respect, it must be noted that the MRPII system's PAC module is limited in its functionality, thereby, needing additional third party development of a detailed PAC system, and integration with it for full-fledged operation, or customized needs of control.

Gausemeier *et al.* [11] are of the opinion that although the material flow in the shop floor has reached a high level of automation, information processing related to that has not yet achieved that level. With differences between job processing situations and environments in different shops, disturbances (e.g., accidental events), and corresponding capacity changes in a single shop, and with ever-changing customer requirements from time to time, the PAC system must be flexible enough to accommodate those changes. Additionally, it must be noted that the PAC subsystem is one of the areas of a manufacturing information system (another, for instance, is the BOM-designed integrated system) that must deal with complex databases, because of the presence of heterogeneous types of hardware systems. Thus, these areas need special attention regarding system development and maintenance. Several authors [11,12] have advocated that object-oriented systems can be best utilized to deal with such complex requirements (Fig. 38).

Gausemeier *et al.* [11] also opine that the PAC system needs to fulfill three special requirements, such as, flexibility for requirement change from time to time, a coordinated communication structure for integrating different devices, and efficient scheduling and rescheduling capability in cases of disturbances and overloads. This can be best satisfied by object orientation. Without detailing the object structures and patterns, Gausemeier *et al.* [11] give a brief outline of the system as given below.

Gausemeier *et al.* [11] draw an analogy between the basic construction and operation principles of natural objects (or organisms) and manufacturing

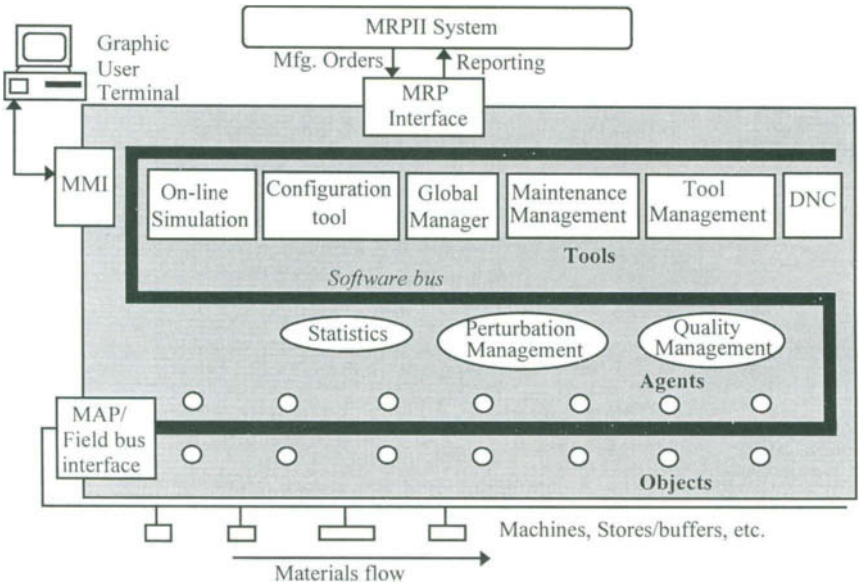


FIGURE 38 A PAC system integrated with MRP [11].

systems. As the authors say,

> Complex organisms are based on organs, controlled by the nervous system, which is in fact a network of communication between organs. Each organ has specialized functions and operates autonomously. The organs are again composed of several specialized cells, though the cells belonging to an organ perform the same basic functions. The entire organic system is managed and coordinated by the communication network, the nervous system. [11]

Similar concepts can be applied to the PAC system, which should be composed of functional elements, such as a machine or a conveyor belt, and a communication network. These basic functional elements are the manufacturing objects (MO), as accepted by Gausemeier *et al.* These MOs need to be scheduled, rescheduled, and monitored. For instance, to complete an order, a sequence of MOs need to developed, and then materials need to be directed along this sequence of MOs. When some additional information, or intelligence, is added to the real-world manufacturing objects by propagating the knowledge of the planner or scheduler, as algorithm or rule, it is termed as an *intelligent object* (IO). A *global manager* (GM), similar to the organic nervous system, is created to coordinate and manage the tasks of these IOs. This GM serves the communication between this enhanced PAC module and the MRPII system (as shown in Fig. 38). The IO is built using the object-oriented principle. As is necessary in an object, an IO encapsulates data and methods. Every IO has methods for scheduling, controlling, and monitoring [11].

The GM provides all information to the IOs for scheduling. The IOs contain different scheduling algorithms, as methods. When jobs are released from MRP to the PAC module, all suitable production devices compete for the job. Once

the IOs calculate grades of competing devices, the one with the highest score is selected for the operation. This is some kind of priority fixing mechanism. The results of the scheduling are set into the time schedule which exists for every IO. In cases of disturbances, which is quite common in any shop, the IO first tries to fix it using strategies and procedures encapsulated in the method. In case, it goes beyond the method, the operator is informed through dialogues [11].

Grabot and Huguet [12] also presented an object-oriented PAC system development method at Aerospatiale. The authors opine similarly to Gausemeier *et al.* that a PAC module changes from time to time, because of the changes of jobs of different customers, disturbances in the shop, expansions in shop size, inclusion and exclusion of resources, i.e., in general, because of factory dynamics. This makes the development process of a PAC module and subsequent integration with an MRPII system difficult, time consuming, and costly. The authors are of the opinion that object-oriented technology, with the ability to reuse past experience by utilizing reference models, can be of tremendous help in this regard.

The authors consider a PAC system composed of PAC modules, which communicate among themselves and with other external systems. There may be several types of modules in a PAC system, depending upon the workshop characteristics. The authors found several such modules, three of them being given in Fig. 39, as example. These modules can be organized through a hierarchic or distributed structure. The design of a PAC system depends on the association of these modules. Once they are defined, the activities involved in corresponding functions associated with each must be described. The functions,

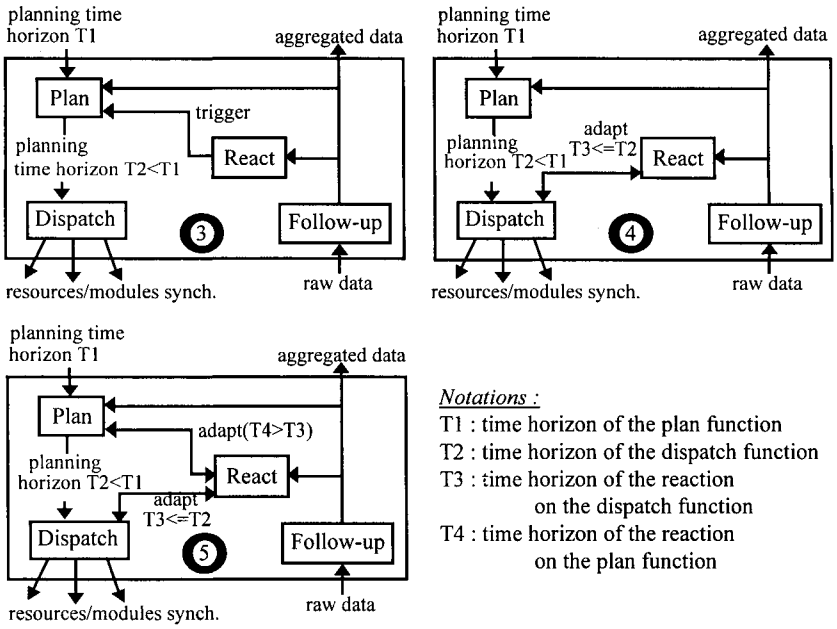


FIGURE 39 Selected typology of PAC modules [12].

associated with these modules are:

- *Plan.* It considers the general plan required in executing the activities of the workshop. The plan function can be ensured manually by the scheduler or a software. In case of the Kanban type of production control, no software is necessary.
- *Dispatch.* It coordinates the resources at the workshop. It releases either a list of jobs to perform or a list of activities/operations to be performed under a job.
- *Follow-up.* It collects all the results provided by sensors, as well as requests from outside of the PAC unit, and then transmits them to other functions of the PAC unit, other PAC modules.
- *React.* It allows an adaptation of the decision made by the Plan function. It generally does not act on the whole plan, but rather on a part of it that may vary according to the global time horizon of the reaction level [12].

Grabot and Huguet [12] use the HP FUSION method for the design of an object-oriented PAC system. It follows the steps given below [12]:

1. Analysis (description of what the system does)
   - *Object model (OM) development.* Object formalism (instantiation, decomposition, inheritance) is added to the entity relationship diagrams. An example of an OM is given in Fig. 40.
   - *Interface.* System interfaces are determined to know about the relationship with the environment.
   - *Interface model development.* The life-cycle model defines allowable sequences of interactions with the environment; and the operation model defines the semantics of each system operation in the interface. An example of an interface model to show links between the workshop and its environment, which is termed here as a scenario, is shown in Fig. 41.
   - Checking the completeness and consistency of the model.

1. Design
   - *Object interaction graphs (OIG).* These are developed to show the exchanges (and their sequences) of messages between objects (an example is shown for dispatch manager in Fig. 42).
   - *Visibility graph.* These are developed to show the structure that enables communication of the *object-oriented* (OO) system.
   - *Class description.* It is the initial codification phase required for describing the internal and external interfaces.
   - *Development of inheritance graphs.*
   - *Updating of class descriptions.*

1. Implementation. It involves codification, optimization, inspection, and testing.

In this method, Grabot and Huguet [12] use state-chart formalism to model the internal behavior of the classes. They focus on the reusability of past reference models to develop new system in a shorter time. They propose some building bricks, or components, for reuse. The components are gathered in functional groups, as shown in Fig. 43.

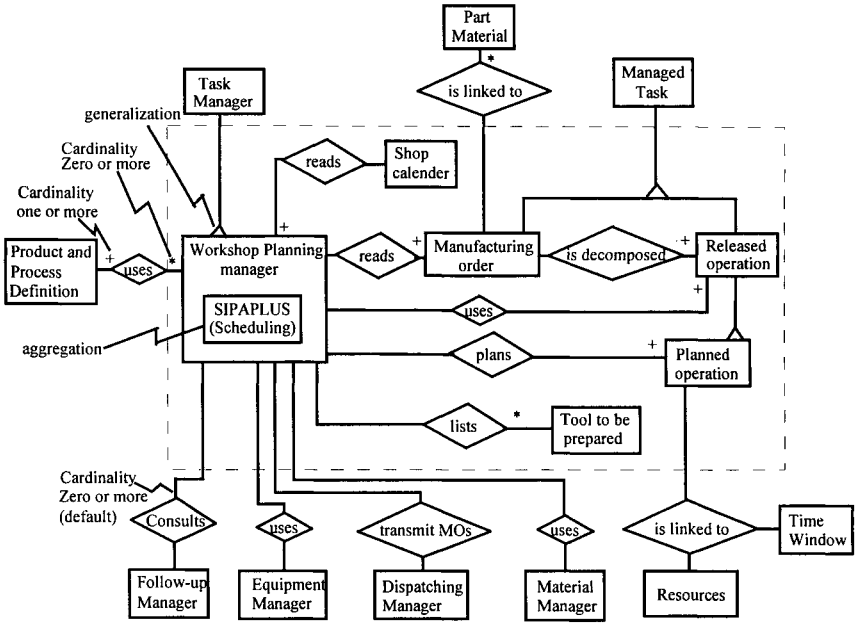


FIGURE 40 An object model of the PAC module components [12].

The OM and OIG of the FUSION method have been suggested for describing each component, whereas the state-chart formalism has been suggested for describing the internal behavior of the classes (i.e., dynamic model). Two case tools, FUSION softcase and Paradigm Plus, have been tested and proven to be useful. The authors are of the opinion that reusability is an essential requirements in the case of PAC system development, which can be ensured by developing first the generic components and then by efficiently managing object and object pattern (component) libraries [12].

Smith and Joshi [46] present an object-oriented architecture for the *shop floor control* (SFC) system, though under the CIM environment. It must be noted that SFC is a dynamic module of MRPII, which is a collection of physical and conceptual objects, when viewed from the OO point, such as orders (to be competed), parts (to be manufactured and/or assembled), machines, equipment, material handling systems (e.g. conveyor belt, forks, robots), and people. The target of OO SFC is to create suitable objects with necessary behavior with their attributes, and methods for interaction among objects, in order to complete the production as per plan generated by the MRPII system, be it as a stand alone island, or under a CIM environment.

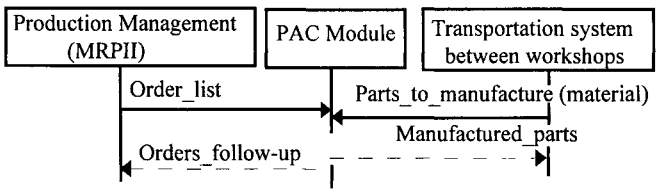


FIGURE 41 An affectation scenario of manufacturing orders [12].

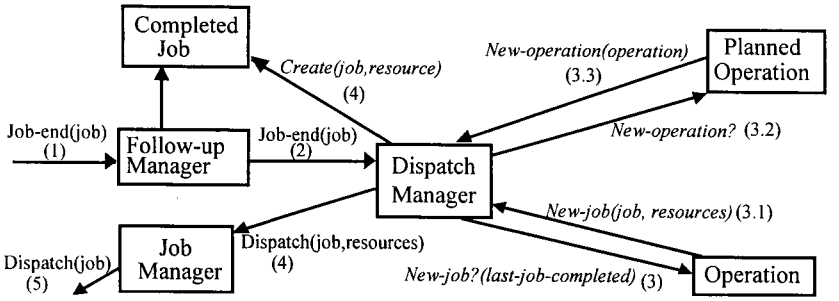


FIGURE 42 A partial view of an object interaction graph of "dispatch manager" [12].

Smith and Joshi [46] have defined the organization of an SFC system as hierarchical, with three levels—shop, workstation, and equipment—in order to improve controllability and make it generic. The equipment level provides a view of physical machines on the shop floor. The authors have created several objects at the equipment level. Out of these objects, which are of particular importance in the case of SFC for MRPII, are *material processing machines* (MP) consisting of the machines, inspection devices, etc. that autonomously processes a part, *material handling machines* (MH) for intraworkstation part movement functions (e.g., loading, unloading, processing machines, and *material transport machines* (MT) for interworkstation part movement functions (e.g., transporting parts from one station to another). At the workstation level, a workstation is considered as a set of equipment level objects and additionally some other objects, such as a part buffer. The shop floor level, composed of workstation objects, is responsible for managing part routing and reporting shop status to other modules.

This SFC system developed a shop floor equipment controller class, consisting of communications and storage classes. These aspects mostly relate to the information integration of CIM hardware systems, which is beyond the scope of this chapter. However, the system has also introduced scheduling as part of the OOSFC (object-oriented SFC) information system, which has direct relevance to MRPII's PAC aspect. The control cycle (function) has been presented as a sequence of activities having the pseudo-codes (thus, generic)

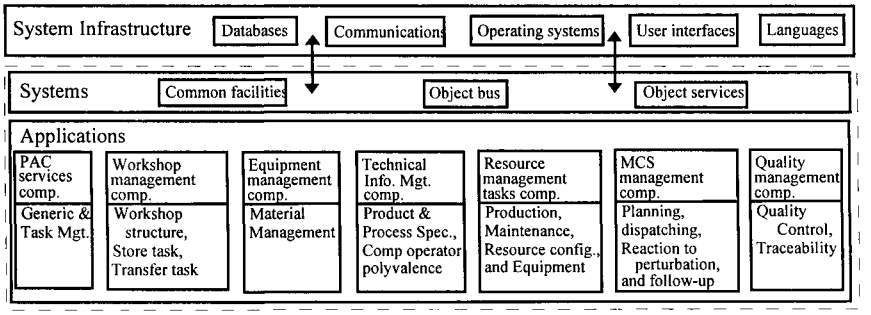


FIGURE 43 PAC development architecture, proposed by Grabot and Huguet [12].

as follows [46]:

```
wait for startup message
read configuration information
do until shutdown message received
  call scheduler
  call task executor
  read messages
  update system states
enddo
write configuration information
shutdown
```

The separate calls of execution and scheduling in the above codes of the SFC system have kept scheduling separate from execution. While "execution" is responsible for determining whether a particular action is valid, and performing it if valid, "scheduling" is responsible for determining an efficient sequence of actions. The scheduler is responsible for placing task request records in the task queue. The task executor, after examining the records, executes the tasks if the preconditions (such as, if machine is idle) are met. By separating the scheduling and execution functions, different scheduling algorithms can be "plugged-in" to the controller, depending on the production requirements. The system has been implemented in C++ [46].

Moriwaki *et al.* [26] have developed an object-oriented scheduling system. Although the system is real-time and has special reference to FMS and autonomous distributed manufacturing, the concept of the system is well applicable to the MRPII system in general. The study first discusses the architecture of autonomous distributed manufacturing systems (for FMS), and then its real-time scheduling system. Since the architecture of FMS is beyond the scope of this chapter, the following section discusses only the scheduling function as it has direct relevance to MRPII's PAC module.

A monitoring system has an important role in this scheduling system, as it needs to monitor and inform the SFC system of the needs for real-time scheduling. The status of the objects is changed according to the status data transmitted from the monitoring system. Thus, the objects are autonomous. The conflicts between objects are resolved through negotiation. The ultimate manufacturing processes are done according to the schedules determined in the system. It considers only machining processes, and not any assembly or transportation process, and is based on the assumptions that the process plans and machining times of individual work pieces are fixed [26].

Figure 44 shows the objects, their attributes, and methods. The objects are classified into four classes—work pieces, process plans of the work pieces, equipment, and operations. The objects representing the work pieces include three classes of objects—part type, lot, and the part. While the "part type" object represents the kinds of part, the "lot" object represents individual lots of that part type, and the "part" object corresponds to individual work pieces in the lot. The "equipment" class is represented by the objects of equipment type and equipment. The process plans of individual part types are established by the process objects and preference relations objects. The operation objects give the

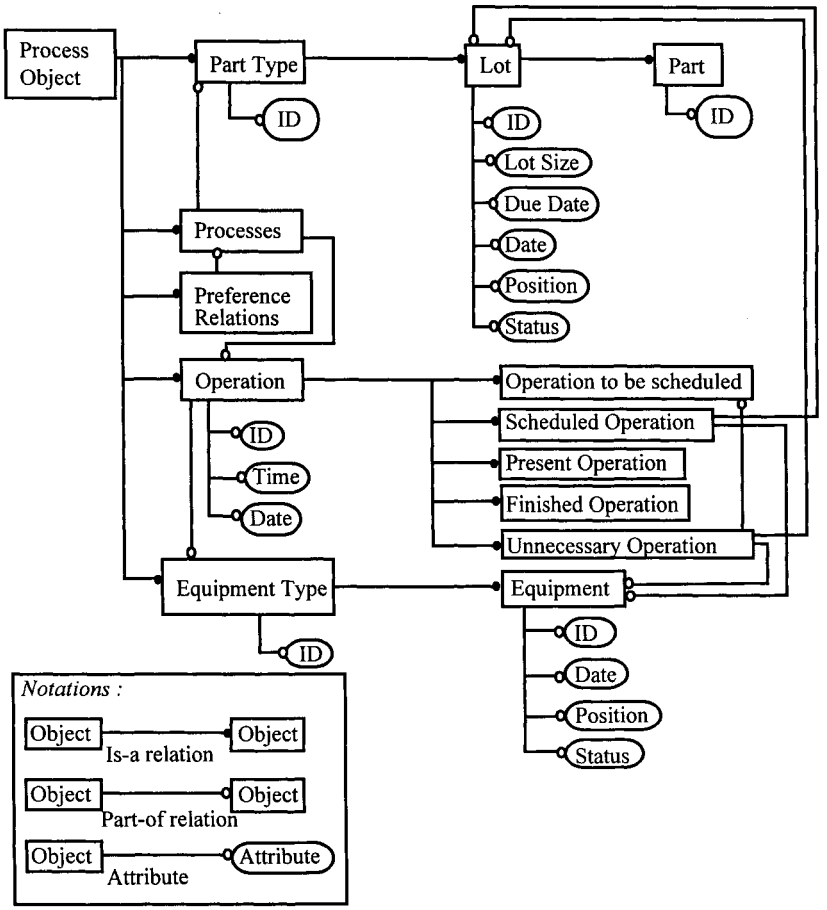


FIGURE 44 Scheduling system objects [26].

relations among the processes of the part types and feasible "equipment type" that can perform the operation. An example of such processes, operations, and their preference relations of part types A and B is shown in Fig. 45. In this figure, a_i and b_j are the processes, and the arcs are the preference relations among the processes. Each process has been defined as a set of alternative operations available (e.g., a_{ik}, b_{jm}), which specify a feasible type of machining cells ("equipment" and "equipment type" objects) [26].

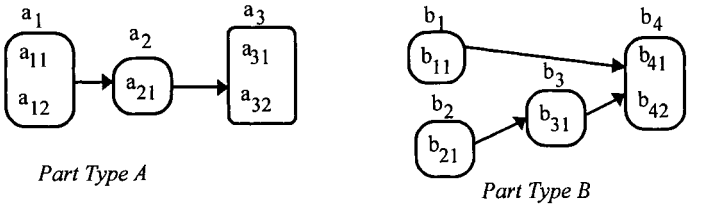


FIGURE 45 Preference relations between processes [26].

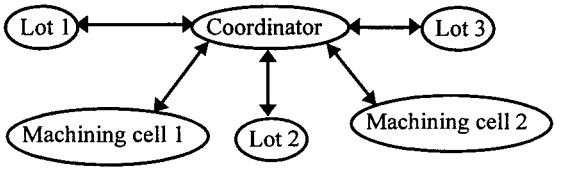


FIGURE 46 Negotiation through coordinator [26].

The schedules of individual machining cells ("equipment") and lots are determined by selecting a suitable operation of that lot, and by loading the selected operation to a suitable machining cell. The start and finish times are specified accordingly. Then the scheduled operations of that lot are performed by the cells ("equipment"). Therefore, the status of the operations are changed as the scheduling progresses. Five possible statuses are given in Fig. 44. In this system, each machining cell (an "equipment" object) can select a suitable operation to perform and determine start and finish times of the operation. On the other hand, each lot can also select and determine a schedule. In some cases, conflicts may arise when several cells select the same operation of a particular lot. A negotiation that is established by creating another object, "coordinator," as shown in Fig. 46, is necessary. A summarized version of the complete scheduling is shown in Fig. 47. A prototype system in Smalltalk was developed [26].

D. Object-Oriented Capacity Planning System

Capacity planning is an aspect of scheduling, which in turn is an aspect of PAC of MRPII. Since it presents some additional information, when described in detail, this is considered separately in many cases in commercial systems. Thus, it is presented separately in this chapter too.

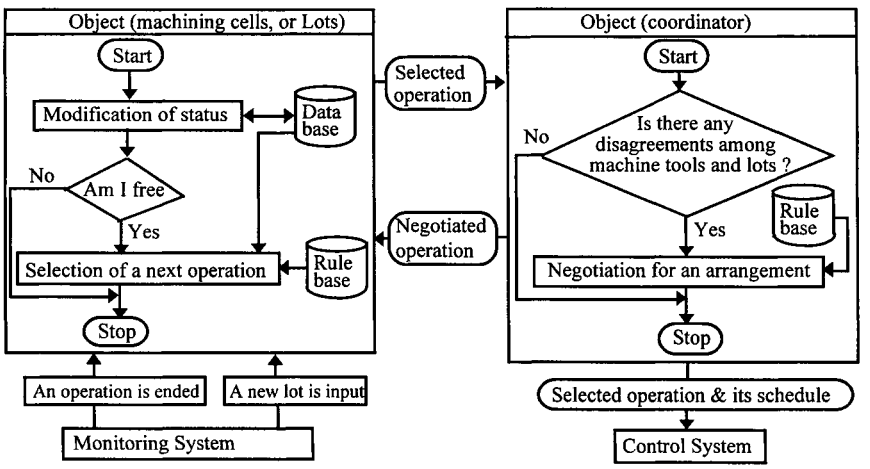


FIGURE 47 Scheduling process [26].

Ulrich and Durig [50] present a capacity planning and scheduling system, using object-oriented technology. It is known that these two functions together aim at achieving a production schedule on the time axis (time horizon) consuming (available) resources.

In order to ensure reusability of objects, a three-level OO system is suggested as follows [50]:

Universal platform. It provides a universal base (library) of available objects that are quite generic and, thus, applicable to several kinds of application areas/functions.

Specific platform. This becomes more specific to a functional application area (e.g., capacity planning). The objects are defined based on the field/function in addition to the universal ones.

Individual supplements. For a field (such as capacity planning), individual realizations need supplements to the specific platform. It may be an individual capacity planning solution for a particular case, whereas the specific platform defines the capacity planning method in general.

Three basic elements, as listed below, are identified as describing the problem at a very high level of abstraction for the specific platform [50].

Resources. It represents capacity of resources, such as machines and labor.

Planning items. These are the production orders, generated internally, or ordered externally.

Time axis. It shows a shop calendar on a continuous time axis.

The following two data administration techniques for administering a set of objects of the same kind that belong to a collection are developed: (i) Open administration, which enables the user to have access to these objects and can perform normal database operations, and (ii) hidden administration, which does not allow the user to have access to these objects, and the administration is performed automatically by the system [50].

The consumption resources comprise consumption elements, which perform the acts of consumption, for which a hidden data administration is necessary to manage those "acts." Each resource administers its adjoined consumption elements according to a well-defined pattern, e.g., intervals and areas. A link to the shop calendar helps to perform that along the time axis. A scheduling of the consumption elements along the time axis according to the defined pattern determines the availability of resources [50].

This kernel of resources and consumption elements, administered by the data administration technique, is linked to the production process environment for consumption (Fig. 48). Production factors, such as machines and labors, comprise resources, represented by workload or the available material for consumption to the production process. The planning items utilize the consumption elements, which are allocated to resources. The user manages these directly, using open data administration [50].

In order to demonstrate the use of individual supplement object classes, a particular task of capacity-oriented time scheduling of production orders for the period of 1 week to 3 months (i.e., short term planning) for final assembly

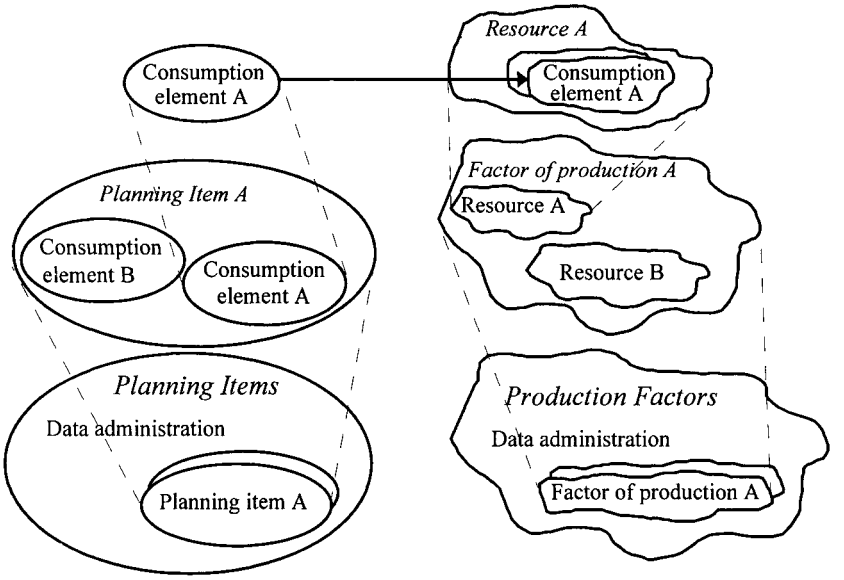


FIGURE 48 Object-oriented view of the relation between planning items and production factors [50].

has been illustrated. For this situation, the hierarchies of abstract objects for specific platforms and individual supplements are given in Table 17. The classes in italics are additional for individual supplements. For every customer order, a number of elementary orders may exist, which act as internal production orders, and are released to fulfill a specific customer request. A new class "production order" is introduced as an individual supplement class, comprising "customer order" and "planning item". The "planning item" contains all general methods for performing planning activities within the scheduling process. The new subclass "elementary order" inherits all these methods, in addition to some supplementary methods for this specific application [50].

E. Object-Oriented Bill of Materials

Several authors [7,8,49] have reported successful development of an OOBOM (object-oriented BOM) system, narrating its advantages. On the other hand Olsen *et al.*, [35] disagree. According to Olsen *et al.*, although the power of an OO approach is recognized in general, the composite object hierarchy and inheritance features are not advantageous for modeling product structures. The reason behind this is that in an OO system, the components inherit the features of their parents, which violate the requirement of describing components independently of their utilization.

Trappey *et al.* [49] state that a conventional BOM structure, which manages data in relational database management style, cannot satisfy the needs requested by all departments in a company. In this respect, they mention several limitations, such as rigidity of data structure, inability to describe behavioral relations, and difficulty in changing data description. They advocate that the object-oriented programming (OOP) concept can design a BOM (OOBOM)

TABLE 17 Specific Platform and Individual Supplement Classes [50]

Specific platform	Specific platform and supplement
Object	Object
Model	Model
Production planning model	Production planning model
Element of production	Element of production
Factor of production	Factor of production
Planning item	*Assembly line*
Process data	*Production order*
Calendar	*Customer order*
Data-administration	Planning item
Open administration	*Elementary order*
Calendar administration	Process data
Production order administration	*Bill of Materials*
Planning item administration	*Production parameters*
Process data administration	Calendar
Hidden administration	Data-administration
Resources	Open administration
Interval covering resource	Calendar administration
Area covering resource	Production order admin.
Element of consumption	*Customer order admin.*
Interval covering element	*Elementary order admin.*
Area covering element	*Production factor administration*
	Assembly line admin.
	Process data administration
	Bill of materials admin.
	Production parameter admin.
	Hidden administration
	Resources
	Interval covering resource
	Area covering resource
	Element of consumption
	Interval of consumption
	Interval covering element
	Production element
	Setup element
	Blocking element
	Area covering element
Parts view and controller	
View	View
Production planning view	Production planning view
One-dimensional view	One-dimensional view
Two-dimensional view	*Assembly line view*
Controller	Two-dimensional view
Production planning controller	Controller
One-dimensional controller	Production planning controller
Two-dimensional controller	One-dimensional controller
	Assembly line controller
	Two-dimensional controller

efficiently to complement the limitations of the conventional BOM system. The authors develop a prototype OOBOM in a Smalltalk environment.

Trappey *et al.* [49] suggest that a BOM may consist of the following classes and subclasses:

Part class
- *MfgPart.* Parts to be manufactured in-house
  - *Routing.* Describes the information in process sheets, such as operation number, operation description, and material required.
  - *Packaging.* Specifies the packaging information of a manufactured part, such as container or size.
- *ProcPart.* Parts to be procured
  - *ProcCo.* Records the information relevant to the supplier of a procured part, such as coName (company name), coAddr (address), FOB points, price, and telephone numbers.

The Part class is in fact a meta class, which is an abstraction of components, subassemblies, and products. The definition of an object is encapsulated in the "instant variables" of its class. For example, "Part" has instant variables, such as partName, partNo, parents, type, and attributes. While the "MfgPart" inherits the variables from "Part," it may have additional variables of its own, such as children (subassemblies), consumableTool, packaging, routing, and cadLink. The attributes of a part can be used to describe objects of different collections. The attributes instance variable is a dictionary data type and stores key-value data pairs. The authors cite an example of a turnbuckle where diameter is a parameter. If it has a value of 10, then the list "(diameter 10)" is described as a pair data in the attributes variable of "Part" [49].

The OOBOM [49] stores the data entities of objects, in both a logical (a data entity is a logical representation) and a physical (a table or a record) representation. A data dictionary is responsible for (i) managing and controlling the data, which stores the logical relationship among data entities and relationships between an object and its attributes, and (ii) managing relationships in a physical database among tables and between a table and its columns.

In the OOBOM system [49], the classes for transaction objects are all the subclasses of the "BomRecord" class, which serves as the data dictionary. It has been defined as the subclass of "Model." The "BomRecord" has the following a set of global variables:

LibPart Dic. Records all existing standard parts in the categorized sets.
LibraryCategory. Lists all part categories.
PartRecord. Lists all existing parts, and all part definitions are encapsulated in PartRecord by the part names.
ProductCategory. Stores the categories of final products.
ProductCategoryDic. Lists all final products organized by product categories.

The hierarchy is shown below [49]:

```
Object ()
  Model ('dependents')
    BomRecord ('model')
    BOMPrint ('selProduct' 'selType' 'maxLength')
    NodeTree ('buttonSel' 'onPart' 'myWindow' 'myComposite'
              'compositeSet' 'lineComposite' 'howManyNode')
    PartLibrary ('libSel' 'libSet' 'partSel' 'partSet')
    PartWindow ('dealPart' 'itemSel' 'itemSet' 'valueSet' 'valueSel'
                'parentSel' 'childSel')
    ProductBrowser ('categorySelectOn' 'finalProductSelectOn'
                    'finalProductSet' 'textMode')
```

The BOMPrint, NodeTree, PartLibrary, PartWindow, and ProductBrowser classes manipulate the transaction management via the MVC (model–view–controller) paradigm of Smalltalk. While doing this, each uses its own window interface, as shown in Fig. 49, and described below [49]:

PartWindow. It supports the basic operations on part data, such as part creation, modification, and query. It is invoked by all other windows to support part data entry and editing.

ProductBrowser. It provides users with functions for browsing the final products in the BOM database. It can invoke a BOM tree to define or edit a product structure. Indented BOM can also be shown.

PartLibrary. It manages all standard parts by grouping them in categories.

BOMTree. It shows the assembly structure of a product.

Authors [Trappey *et al.*] [49] also advocate that this OOBOM system can further be extended to integrate engineering data management system (maybe MRPII) and CAD/CAM as well.

Chung and Fischer [8] gave an outline of the structure and a data model of an OOBOM. They also demonstrated the use of the proposed architecture by developing a prototype system through coding in C++ language. The authors are of the opinion that a structure of BOM is in fact an abstraction hierarchy

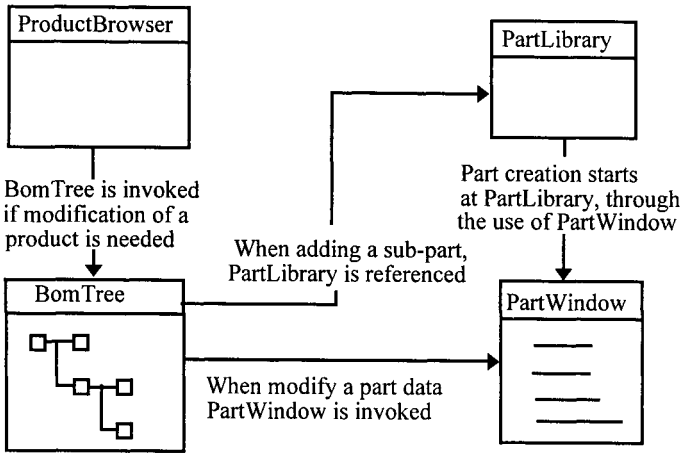


FIGURE 49 The transaction model [49].

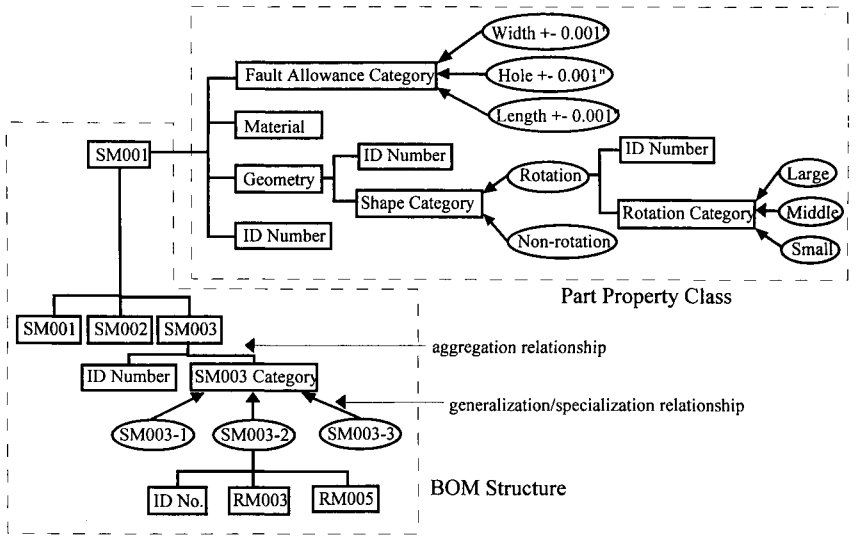


FIGURE 50 Example of aggregation/generalization in an OOBOM system [8].

of an OO data model, and thus, the conceptual data model of OOBOM can be mapped onto the abstraction and inheritance architectures of an OO data model. The conceptual data model integrates the semantic relationships, such as References, Referenced-by, Owns, Owned-by, Composed-of, and Part-of, with object orientation concepts.

In an OOBOM system, the parts, which are assembled together to form a subassembly, can have aggregation semantic relationships, i.e., is-part-of relationship, where the subassembly is termed an aggregation object and the parts that go into subassembly are called the component objects. For instance, in Fig. 50, SM003 is an aggregation object and ID number and SM003 Category are the component objects. A generalization relationship, i.e., is-kind-of relationship, is used to define generic part objects from their possible categories or options. For instance, in Fig. 50, Rotation and Nonrotation are the categories of the generic object Shape Category. SM003-1 is a possible option of SM003; thereby, it is having a specialization relationship, which is the converse of a generalization one [8].

The authors propose two classes of objects, namely the BOM class, which simply defines the physical and logical relationships between components in the hierarchy of BOM, and the Part Property class, which defines the properties, such as shape, color, and material, of the parts in the BOM class (Fig. 50 and 51).

They opine that this classification is similar to E-class (entity class) and D-class (domain class) of an OSAM database respectively. The BOM objects may reference (Reference semantic relationship) the properties in the Part Property class. The Own semantic relationship can be used to generate/delete/change/modify an individual object to maintain engineering changes that happen frequently in production organizations. While a BOM object may contain a set of instances created by the user of the OOBOM database, the property object does not contain the user-created instances. The property object is only a property that specifies the data type and/or a permissible range of values [8].

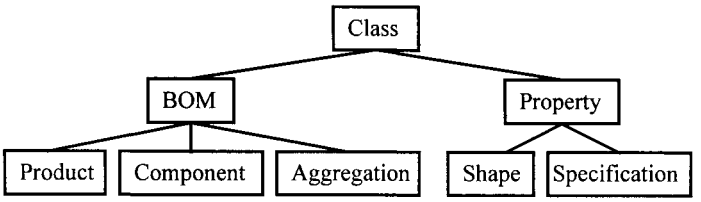


FIGURE 51 Class hierarchy, as used in Fig. 50 [8].

An object in the BOM class can get property information of the Property class via Referencing (R), Generalization (G), Own (O), and Aggregation (A) relationships. When referencing is needed, an attribute (which is a character string that is the unique identification of the referenced object, or an address pointer to the object) that establishes reference to another object is created, thereby creating a relationship between two objects. Communications between objects are established by sending messages through the execution of methods and utilization of defined semantic relationships. Since several objects may try to have a lot of communications, a control mechanism is necessary to simplify that. The OOBOM allows a relationship and communication between objects when, and only when, one object has an attribute that is a reference to another object. The relationships are shown in Fig. 52. In the case of referencing, the messages

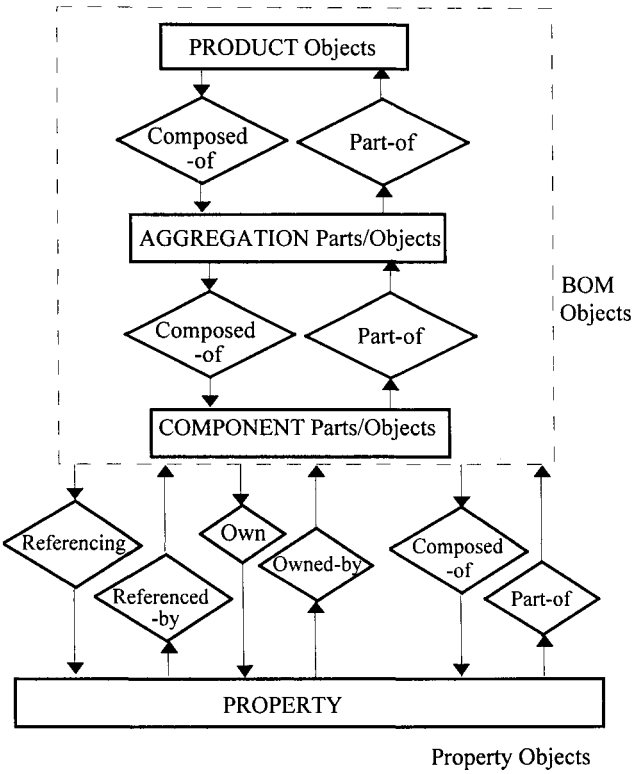


FIGURE 52 Relationships in a BOM hierarchy [8].

associated with a Referencing relationship are sent by the BOM object, and any message associated with the Referenced-by relationship are sent by the property object. The existence of a Referenced-by relationship means that a property object contains a list of all the BOM objects that reference the property object [8].

In an Aggregation relationship (Fig. 52), messages associated with composed-of relationships are sent from parent to child, whereas messages associated with part-of relationships are sent from child to parent. In an MRPII environment, there may be frequent engineering changes. In such a situation, decomposed components need to be updated if their parent is modified. In an Own relationship, BOM object may own another object, which may be a property object as well. The owner object can order the destruction of the owned object and can perform operations, like resetting the values of the owned objects' attributes. An example of an Own relationship is the creation and maintenance of BOM and property objects [8].

Chung and Fischer [8] have listed possible attributes of both BOM and property classes. The example of a property object is given in Table 18.

In another previous occasion, Chung and Fischer [7] illustrate the use of an object-oriented database system, using ORION, a commercial OO database system of Itaska, for developing BOM for use in the MRPII system. ORION is a distributed database system. As such, the object identity also needs the site identifier of the site in which the object is created. Being opposed to the traditional idea of using a relational database system for BOM, the authors mention

TABLE 18 Property Object Description [8]

CLASS NAME	P-RM005
DOCUMENTATION	Properties of BOM Object RM005
CLASS ATTRIBUTES	
Superclass	PROPERTY
Subclass	None
Class category	Property
OBJECT IDENTITY	Object Identifier
DATA ATTRIBUTES	Lead-Time, Make-or-Buy, Cost, Safety-Stock, Stock-on-Hand, etc.
RELATIONSHIP METHODS	
Referenced-by-Relationship	All BOM Objects
Owned-by-Relationship	RM005
Part-of-Relationship	P-SM001, P-SM003
OTHER METHODS	Query (Query methods are used to search and retrieve properties.), Check (Check and/or reset methods can be exploited to modify and initialize the values of the part attributes.), Reset, Display (It controls the display of the details and is necessary for linking with query, check, and reset methods.)
END CLASS	

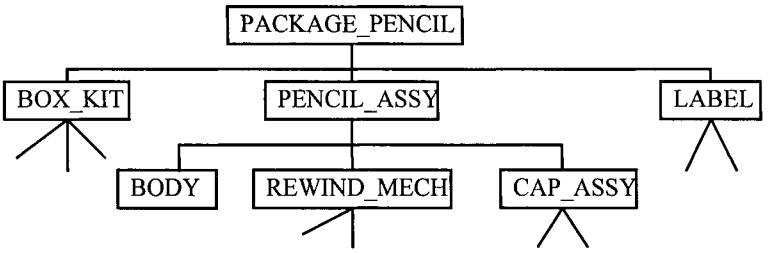


FIGURE 53 An example of a BOM [7].

several limitations, and advocate the use of OODB instead. The authors opine that unlike relational data models, an OO data model allows the design of an application in terms of complex objects, classes, and their associations, instead of tuples and relationships. This characteristic of the OO technique is highly desirable in a BOM system, as a BOM is a complex composite object.

Aggregation is useful and necessary for defining part–component hierarchies, and is an abstraction concept for building the composite objects from the component objects. It must be remembered that BOM is a hierarchy that relates parent–component relationships, and several components are logically (logically during the modeling stage and, later on, physically on the shop floor) assembled together to form a composite object. During operation of the BOM module in the MRPII system, it may be necessary to perform some operations (such as deletion, addition, storage management) in the entire BOM, considering it as a single logical unit. In the OOBOM proposed by Chung and Fischer [7], the BOM has a composite object hierarchy with an is-part-of relationship between parent and component classes, and this composite object may be treated as a single logical unit, and thus fulfills the requirement.

Fig. 53 is an example of a PACKAGE_PENCIL object class, having component object classes of BOX_KIT, PENCIL_ASSY, and LABEL. The PENCIL_ASSY is again composed of several component classes. Thus, the BOM shown has a hierarchy of object classes that can form a composite object hierarchy.

The detailed information on the PENCIL_ASSY object class is shown in Table 19.

The components in a BOM require a unique component identifier. This (in the form of an object identifier) is created automatically in ORION, without the designer's interference, while instantiating an object. The object identifier is used to facilitate the sharing and updating of objects. The uniqueness of an identifier allows a direct correspondence between an object in the data representation and a subassembly in BOM.

During creation of a BOM, each part is defined independently, and then dependency relationships are established between parts and components defining parent–component relationships. In the case of OOBOM, objects are created independently, but then dependency relationships between object instances are established. An example of such a BOM for PACKAGE_PENCIL is in Fig. 54. There exists the aggregation relationship between the component instances associated with PACKAGE_PENCIL. For example, each PACKAGE_PENCIL

TABLE 19 PACKAGE_PENCIL Object Class [7]

PACKAGE_PENCIL CLASS	
Class name	PENCIL_ASSY
Object identity	OID
Documentation	Parts for PACKAGE_PENCIL, part supplier, part description, part material, part specification,
Superclass	PACKAGE_PENCIL
Subclass	BODY, REWIND_MECH, CAP_ASSY
Attributes	Lead time, safety stock, quantity on hand, quantity per, assembly time, machines_and_tools.
Methods	display, change_part, add_part, delete_part, manufacturing_techniques, material_requirement_planning, machining_process, inventory_control.

instance owns specific instances of component classes PENCIL_ASSY, BOX_KIT, and LABEL [7].

In an industry, sometimes a planning BOM [39,40] is necessary, when there are several options against a particular part. For example, METAL_CASE type BOX_KIT may have two options, that made of iron and that made of copper (Figs. 54a and 54b). In the OOBOM [7], this type of situation has been dealt with efficiently using the concept of generic instance and version instance. In Fig. 54b, METAL_CASE may have a generic instance, which may have two options of iron and copper, known as version instance. In Fig. 54c, the M292 model is a generic instance of PACKAGE_PENCIL class, whereas M292-1 is a particular version instance for M292. In Fig. 54a, M292 is a model of PACKAGE_PENCIL, in which the BOX_KIT may be of metal type,

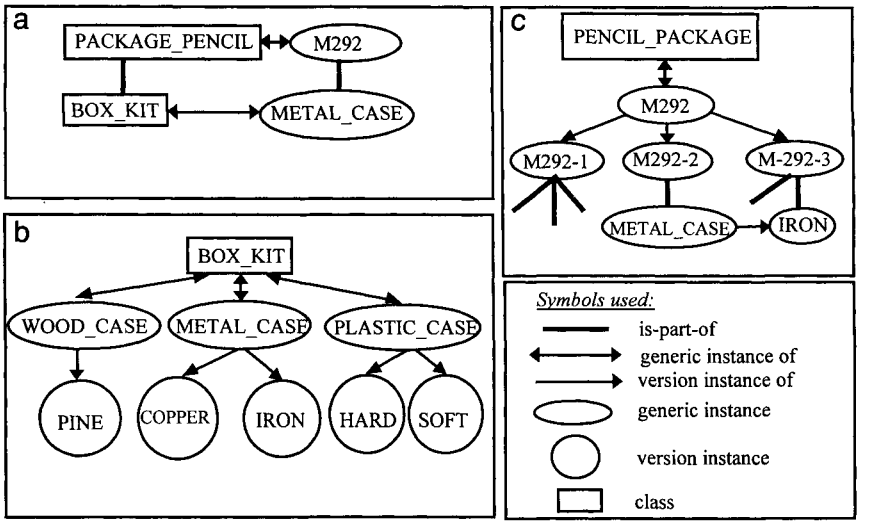


FIGURE 54 Hierarchy of object classes and instances [7]. (a) Composite link between generic instances. (b) Hierarchy of version and generic instances of BOX_KIT. (c) Object version hierarchy.

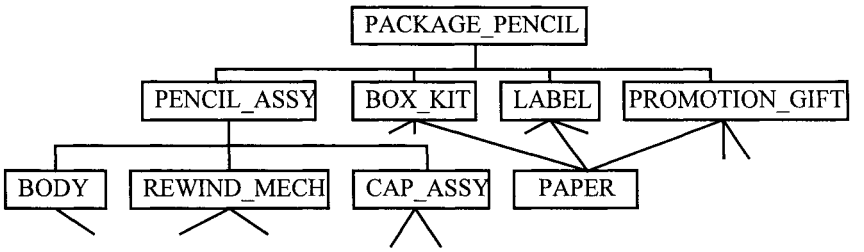


FIGURE 55 An example of multiple inheritance [7].

i.e., METAL_CASE. Again, M292 may have several options, like M292-1 and M-292-2. Thus, M-292 is a generic instance of PACKAGE_PENCIL, and METAL_CASE is a generic instance of BOX_KIT. The generic instance may be used to reference objects (e.g., iron) without specifying in advance the particular version needed. Changes in the definition of a class must be propagated to its generic instances unless a new generic or version instance is explicitly created.

When part/component is used to assemble several parents in a BOM, there arises the situation of multiple inheritance, as shown in Fig. 55. In this situation, there may be name conflicts, which have been resolved by arranging (user-defined or default ordering) the order of the parent class for each component class. In this example, the order of parent classes for the component class PAPER is BOX_KIT, LABEL, and PROMOTION_GIFT (this is an object class to promote sells). However, this order can be changed by the user, based on requirement. In the case of name conflicts arising from the inheritance of instance variables and methods, one must add new instance variables or modify the name in a class lattice composed of the conflicting component classes [7].

Chung and Fischer [7] suggest that certain numerical algorithms should be added to the OOBOM system, in the form of rule-based procedures, to facilitate integration of OOBOM with the MRP system, which needs to perform calculations for MRP. The authors also suggest that for graphical display, Pascal language may be used for pixel-based graphic operations. However, several other issues need to be addressed.

F. MRPII as an Enterprise Information System

On many occasions, OO information systems have been modeled, where MRPII played either the central role of a company-wide information system or a major role in it. [22,57].

An information system is considered a vital ingredient of a computer-integrated manufacturing (CIM) system. An efficient design of an information system, based on data of different islands of information, is thus a must.

Ngwenyama and Grant [57] present a case study for modeling an information system involving MRPII as the center part. The primary reason for using object-oriented paradigm is the recognition that the OO approach offers powerful, but simple, concepts for representing organizational reality and the system designer's ideas about the system. They identified three levels of abstraction in OO modeling: (i) organizational, (ii) conceptual, and (iii) technical, as shown in Table 20.

TABLE 20 Conceptual Framework for Information Modeling [57]

Levels of modeling	Object-oriented concepts		
	Objects	**Messages**	**Methods**
Organizational	Business process	Information flows	Business procedures
	Business activities	Physical flows	Decision making procedures
	Organizational roles		
	Information views		
Conceptual	Information handling	Data flows	Information processing procedures
	Processes		Communication procedures
	Communication processes		
	Data		
	Subject databases		
Technical	Information systems	Data flows	Application programs
	Database structures		Operating procedures
	Hardware		
	Systems software		
	Communication system		

Ngwenyama and Grant [57] develop the information system using seven conceptual models. The procedure for deriving a model is as follows: (i) identify candidate objects, (ii) classify them as active or passive, (iii) define their characteristics, (iv) define the methods they enact with or which operate on them, (v) define the messages that they send or receive, and (vi) construct the diagrams. The models are shown in Fig. 56.

The object of the *global model* is to capture, at the highest level of abstraction, the interactions among the organization system and/or subsystems. This shows the primary information links (messages). Developing the global model requires the definition of major operating systems, also considered objects, embedded in the enterprise without regard to artificial boundaries, such as departments. The MRPII system and its modules have the central role of linking several business activities at this level. Next, the *business model* decomposes each subsystem (i.e., objects) into its set of related activities/processes. These can be further divided into individual tasks. Later, the information flow

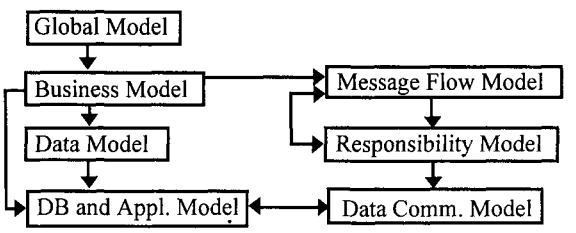


FIGURE 56 Models of an information system [57].

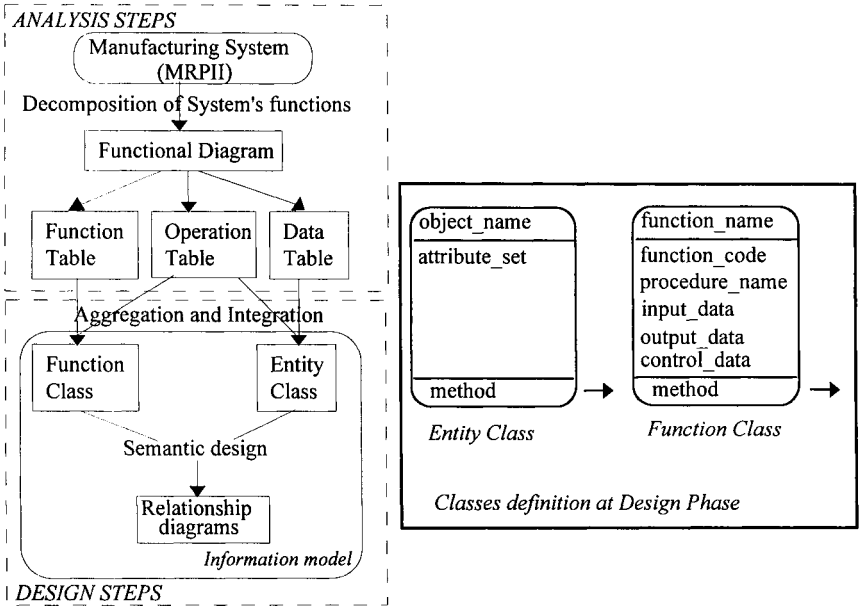


FIGURE 57 OOMIS Methodology [22].

(messages) received or generated by such activities/functions/tasks at different levels of abstraction are described in the *message flow model.* The *responsibility model* documents the role of responsibility for information processing at various levels of organization. It documents which objects (such as machines and individuals) have access or control over what messages. Then the conceptual design of constructing the detailed architecture starts, which is done through the *data model, database and application model,* and *data communication model.* The names of these three models describe their roles and functionalities. However, the authors did not give any detail of the data construction of each of the entities or objects of any system (e.g., MRPII).

Kim *et al.* [22] present an object-oriented information modeling technique for MIS, which is in fact the MRPII system. In place of existing methodologies, like IDEF and GRAI, the authors develop a new technique, known as *object-oriented modeling for manufacturing information system* (OOMIS), which uses a representational scheme, class, for integrating manufacturing (MRPII) functions and data. The OOMIS consists of two steps, analysis and design, as in Fig. 57 [22].

In the analysis phase, the complete manufacturing (MRPII) system is decomposed into its functions. The functional diagram in Fig. 58 shows three such functions of the complete MRPII system. These are again divided into subfunctions, up to specific small interacting unit functions, which can perform fundamental and simple operations. The functional diagram in Fig. 59 shows the next level decomposition for inventory control function (module of MRPII) as seen in Fig. 58 [22].

From the proven psychological fact that it is difficult to grasp more than five to seven distinct concepts at a time, this OOMIS methodology suggests decomposing a function into not more than six subfunctions at a level. Then,

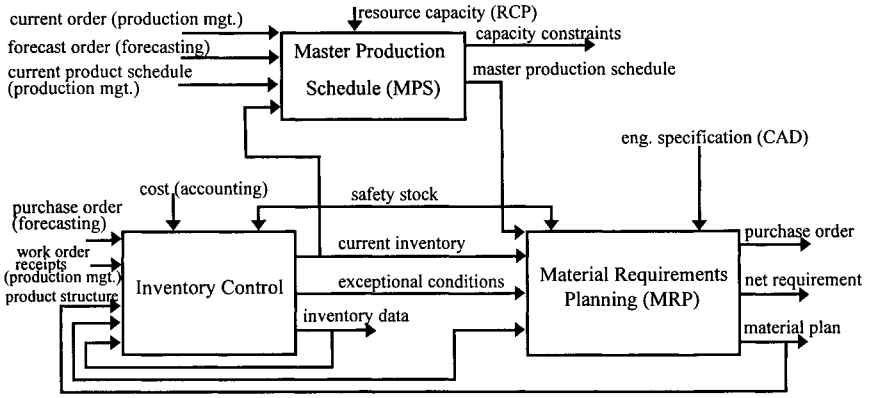


FIGURE 58 Partial functional diagram of MRPII system [22].

similar to the IDEF technique, the functional diagrams are prepared indicating inputs, outputs, and control mechanism, as shown in Figs. 58 and 59. In the diagrams, information and materials must be distinguished by their different logistic properties. The functional diagrams are converted into a function table giving some additional information, like function code and procedure name. The function table for inventory control function is shown in Table 21a. The function code includes some codes for indicating decomposition level and order. The data table represents output data of the functional diagrams. This data may be modified and referenced by other functions, since the complete system's functions are highly interrelated and interacting, as is the case of MRPII. A data table contains function name and code received respectively from the functional diagram and table; attribute_name, which is a part of the manufacturing data and represents the output data name literally; attribute_of, which is the entire manufacturing data; and a description of attribute. Next in the hierarchy is the operation table, which describes how a manufacturing function handles its operations. The operation table for inventory control function is shown in Table 21b. The operations in the operation table are the aggregated set of processes or procedures of the unit functions, which become the class methods later in the design phase [22].

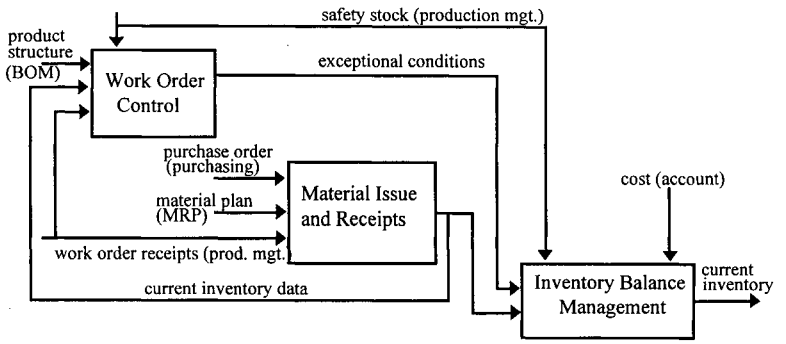


FIGURE 59 Decomposed inventory control function [22].

TABLE 21 Function and Operation Tables for Inventory Control [22]

(a) Function name: inventory_control

function code: F7

procedure name: i_c_proc

input data: material plan (MRP)
purchase order (purchasing)
work order receipts (production mgt.)
product structure (BOM)

output data: current inventory status

control data: cost (account)
safety stock (production mgt.)

(b) Function name: inventory_control	**Function code: F7**
Operation	**Description**
inventory_trans	Process all inventory translation on line
report_inventory	Report current inventory status
material_receipt	Update received material
work_order_proc	Process work orders and report material issue
proc_bom	Make part list from work order

In the design phase, the tables obtained in the analysis phase are translated into an object-oriented information system (such as for MRPII) using OOMIS. The aggregation and integration process converts the function, operation, and data tables into a class dictionary having two types of classes, namely function class and entity class (Fig. 57). As example, a part of the class dictionary is shown in Table 22. Here, the operations of inventory control function (Table 21b) have been used to define the methods of the classes of Table 22. While the function class describes the properties of the MRPII functions, the entity class describes the data. The class dictionary can be translated into a specific data dictionary of OODBMS. The semantic relationships (aggregation, generalization, unidirectional and bidirectional synchronous and asynchronous interaction) are used to describe class relationships. Figure 60 illustrates the aggregation hierarchy of the classes of Table 22 [22].

The aggregation process is carried out by identifying common concepts in the function, operation, and data tables. The attribute_name tuples of the data tables, which have the same attribute_of field, are aggregated to represent a manufacturing datum whose name is attribute_of. If the function tables have the same input and control data, these are merged together, where the merged function table contains the output data that are related to each function table. This combined table can refer to the operation tables, which are then related to each function table, to choose a suitable method for organizing the function table. Then the integration process is carried out to resolve

TABLE 22 A Part of the Class Dictionary [22]

fn_name: work_order_control	**obj_name: work_order**	**obj_name: inventory_date**
fn_code: F7.1	product_id:	product_id:
proc_name: w_o_c_proc	batch_id:	part_id:
in_data: work_order	priority:	on-hand:
BOM_info	order_date:	lead_time:
inventory_data	due_date:	lot_size:
out_data: exceptional_conditions	completion_date	on_order:
con_data: safety_stock	quantity:	location:
	cost:	raw_material_type:
	BOL_no:	alternative:
	Routing:	
work_order_control	update	update_inventory_status
obj_name: production_info	**obj_name: part_info**	**obj_name: BOM_info**
product_id:	part_id:	BOM_no:
production_name:	material_code:	product_id:
description:	process_plan_no:	part_id:
version_no:	NC_prog_no:	quantity:
uom:	GT_code:	uom:
part_info:	drawing_no:	assembly_instruction:
BOM_no:		
BOL_no:		
update	update	add_bom
		modify_bom

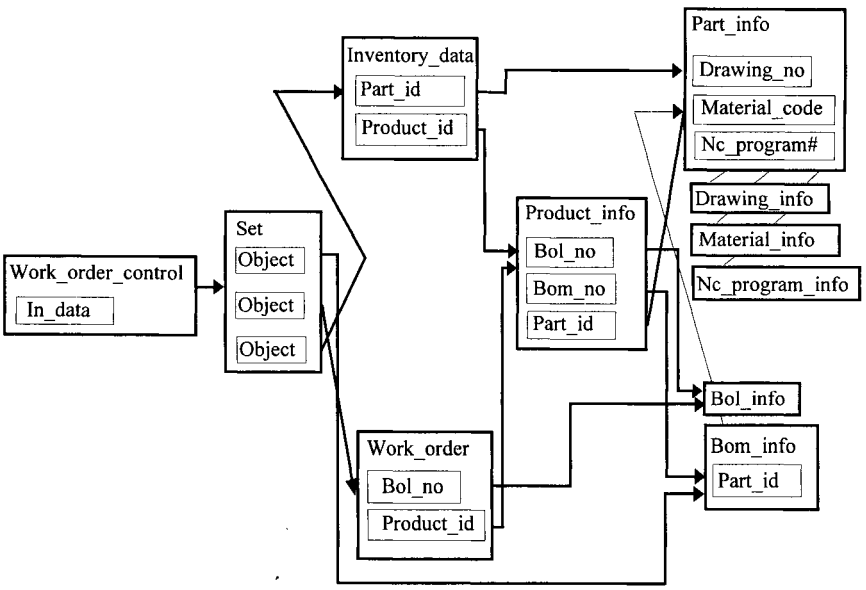


FIGURE 60 Aggregation hierarchy of classes [22].

conflicts in the aggregated tables. During this, each table is compared with other tables to identify and remove conflicts between corresponding components [22].

The function class represents the characteristics of the manufacturing (MRPII here) function. Methods of this class describe the operations of the function. The attributes of the class are function code, procedure name, source function, and manufacturing entities, which are the function's input, output, and control data. Entity class describes the physical properties of the manufacturing (here, MRPII) data (or entities). The methods of this class come from the operation tables, and are responsible for calculating the values of the manufacturing data or evaluating the status of the function [22].

If a new function is necessary in the system (here MRPII), it can easily be designed by selecting the suitable classes from the existing class dictionary. In case there is no such class, new classes may be designed by describing the properties of the function [22].

Huarng and Ravi [19] are of the opinion that object orientation provides the modeler the liberty of representing interactions between object behaviors. This makes it possible to understand the dynamics of information flow. It must be noted that several modules of MRPII, such as BOM and PAC, are highly dynamic, particularly in a make-to-order environment, which in turn, makes the information flow dynamic. Each function in an organization is composed of a sequence of activities, or in other words, each function is a cycle of activities. Such is the case for MRPII as well. For example, order generation can be represented as a sequence of behaviors of the following objects: gross requirement, inventory on-hand, open order.

G. Scopes of Object Orientation

Although object orientation has got momentum in research in the areas of manufacturing, including MRPII, its application has also been defined to be confined within the boundary of some specific areas.

Nof [32,33] assesses what can and cannot be expected to be accomplished from the promising object orientation in manufacturing systems engineering. It simplifies the burdensome complexity of manufacturing, including planning and control and software development. It provides intuitive clarity, discipline, and flexibility essential in the functions of manufacturing. *Object-oriented programming* (OOP) can increase a programmer's productivity. It is also flexible to subsequent modifications. Although most of the MRPII functions do not contain complex data structures, some modules, such as PAC, BOM generation, and its (optionally) graphical display, indeed need a complex data structure. In such a case, object orientation is favorable. *Object-oriented control* (OOC) and object-oriented modeling also provide distinct advantages, such as modularity of modeling and logic requirements, direct and intuitive mapping of real objects to software objects, and minimization of efforts of software development through reusability. As discussed earlier, these are specially useful in the case of PAC system design. Table 23 lists some manufacturing objectives, which includes functions of MRPII, and the possibility of object orientation.

With the increase in the abilities of computations and extended functions in the MRPII system, customer expectations are also increasing. Software

TABLE 23 Affinity Mapping between Functions and OO Premises [32,33*]

Objectives	Object-orientation promises
Produce well-defined, repeatable, interchangeable parts and products	Classifies objects in hierarchical structures, having data and attributes encapsulated
Manufacture by process and assembly plans	Applies assembly structures and a high level of abstraction
Based on methods for defining process operations and manufacturing services	Methods define operations and services inside an object and pass it through messages
Design complex products and process their elements	Specifies complex systems by their elementary object components
Combine information and materials processing to create end products	Combines data and process model together in one object
Depend on consistent and systematic plans	Represents consistently and systematically natural reality and human thinking
Communicate and integrate among multiple functions	Communicates by messages, polymorphism, and corresponding operations
Have a culture of clear, ordered, stable processes and procedures	Model stability, clarity, and flexibility due to minimal dependency between objects
Rapidly adapt to change	Reusability minimizes effort to rewrite codes

*Copyright Taylor & Francis.

developers are providing these additional functions as much as possible with minimal effort. Sometimes data textual elements, graphics, and even knowledge are also introduced together in a single system. In many cases, object orientation has been successful in minimizing cost of development by minimizing effort. Table 24 lists some complexities in the MRPII systems [32,33].

Nof [32,33] finds object orientation of particular advantage in the case of several manufacturing functions, which dominantly includes MRPII, although it has some limitations too, which include a high run-time cost among others. However, the author in conclusion hopes that the limitations will be relaxed by emerging and future capabilities.

Wang and Fulton [53] report on a case of manufacturing information system development. Although it is an MIS in general, it contains many MRPII functions, though not in a disciplined way as in MRPII. Earlier the system was developed using a relational DBMS. Later the system was developed using

TABLE 24 Increasing Complexity in Manufacturing Software [32,33*]

Information level	Examples of manufacturing applications (MRPII)
Data	All modules of MRPII
Graphics	Bill of Materials (Some MRPII systems)
Information	All modules of MRPII
Knowledge	Sometimes, scheduling, forecasting, etc.
What-if simulation	Production flow analysis, lead time determination

*Copyright Taylor & Francis.

OODMBS. The reasons behind this are the limitations of RDBMS and the corresponding advantages of OODBMS. As stated, a relational data model has several drawbacks including a lack of semantic expression, a limitation of supporting heterogeneous data types, a difficulty in representing complex objects, and an inability to define a dynamic schema, whereas these are overcome in object-oriented design and OODBMS. Thus, object orientation has been highly recommended for such manufacturing information systems (e.g., MRPII).

REFERENCES

1. Arnold, J. R. T. *Introduction to Materials Management.* Prentice-Hall, Englewood Cliffs, NJ, 1991.
2. Berio, G. *et al.* The M*-object methodology for information system design in CIM environments. *IEEE Trans. Systems Man Cybernet.* 25(1):68–85, 1995.
3. Bertrand, J. W. M., and Wortmann, J. C. Information systems for production planning and control: Developments in perspective. *Production Planning and Control* 3(3):280–289, 1992.
4. Browne, J. Production activity control—A key aspect of production control. *Int. J. Prod. Res.* 26(3):415–427, 1988.
5. Browne, J., Harhen, J., and Shivnan, J. *Production Management Systems: A CIM Perspective.* Addison–Wesley, Reading, MA, 1989.
6. Chang, S.-H. *et al.* Manufacturing bill of material planning. *Production Planning and Control* 8(5):437–450, 1997.
7. Chung, Y., and Fischer, G. W. Illustration of object-oriented databases for the structure of a bill of materials (BOM). *Comput. Indust. Eng.* 19(3):257–270, 1992.
8. Chung, Y., and Fischer, G. W. A conceptual structure and issues for an object-oriented bill of materials (BOM) data model. *Comput. Indust. Engrg.* 26(2):321–339, 1994.
9. Dangerfield, B. J., and Morris, J. S. Relational database management systems: A new tool for coding and classification. *Int. J. Oper. Production Manage.* 11(5):47–56, 1991.
10. Du, T. C. T., and Wolfe, P. M. An implementation perspective of applying object-oriented database technologies. *IIE Transactions,* 29(9):733–742, 1997.
11. Gausemeier, J. *et al.* Intelligent object networks—The solution for tomorrow's manufacturing control systems. In *Proceedings of the 3rd International Conference on Computer Integrated Manufacturing,* Singapore, 11–14 July, 1995, pp. 729–736.
12. Grabot, B., and Huguet, P. Reference models and object-oriented method for reuse in production activity control system design. *Comput. Indus.* 32(1):17–31, 1996.
13. Harhalakis. G. *et al.* Development of a factory level CIM model. *J. Manufact. Systems* 9(2):116–128, 1990.
14. Hasin, M. A. A., and Pandey, P. C. MRPII: Should its simplicity remain unchanged? *Indust. Manage.* 38(3):19–21, 1996.
15. Higgins, P. *et al. Manufacturing Planning and Control: Beyond MRPII.* Chapman & Hall, New York, 1996.
16. Higgins, P. *et al.* From MRPII to mrp. *Production Planning and Control* 3(3):227–238, 1992.
17. Hitomi, K. *Manufacturing Systems Engineering.* Taylor & Francis, London, 1979.
18. Hsu, C., and Skevington, C. Integration of data and knowledge in manufacturing enterprises:A conceptual framework. *J. Manufact. Systems* 6(4):277–285, 1987.
19. Huarng, A. S., and Krovi, R. An object-based infrastructure for IRM. *Inform. Systems Manage.* 15(2):46–52, 1998.
20. Jackson, D. F., and Okike, K. Relational database management systems and industrial engineering. *Comput. Indust. Engrg.* 23(1–4):479–482, 1992.
21. Kemper, A., and Moerkotte, G. Object-oriented database management: Application in engineering and computer science. Prentice–Hall, Englewood Cliffs, NJ, 1994.
22. Kim, C. *et al.* An object-oriented information modeling methodology for manufacturing information systems. *Comput. Indust. Engrg.* 24(3):337–353, 1993.

23. Korhonen, P. *et al.* Demand chain management in a global enterprise—Information management view. *Production Planning and Control* 9(6):526–531, 1998.
24. Kroenke, D. *Management Information Systems.* Mitchell McGraw–Hill, New York, 1989.
25. Meyer, A. De. The integration of manufacturing information systems. In *Proceedings of the International Conference on Computer Integrated Manufacturing,* Rensselaer Polytechnic Institute, Troy, New York, May 23–25, 1988, pp. 217–225.
26. Moriwaki, T. *et al.* Object-oriented modeling of autonomous distributed manufacturing system and its application to real time scheduling. In *Proceedings of the International Conference on Object-Oriented Manufacturing Systems,* Calgary, Alberta, Canada, 3–6 May, 1992, pp. 207–212.
27. *Micro-Max User Manual.* Micro-MRP, Foster City, CA, 1991–1992.
28. *MRPII Elements: Datapro Manufacturing Automation Series.* McGraw-Hill, Delran, NJ, 1989.
29. Nandakumar, G. The design of a bill of materials using relational database. *Comput. Indust. Engrg.* 6(1):15–21, 1985.
30. Naquin, B., and Ali, D. Active database and its utilization in the object-oriented environment. *Comput. Indust. Engrg.* 25(1–4):313–316, 1993.
31. Nicholson, T. A. J. Beyond MRP—The management question. *Production Planning and Control* 3(3):247–257, 1992.
32. Nof, S. Is all manufacturing object-oriented? In *Proceedings of the International Conference on Object-Oriented Manufacturing Systems,* Calgary, Alberta, Canada, 3–6 May, 1992, pp. 37–54.
33. Nof, S. Y. Critiquing the potential of object orientation in manufacturing. *Int. J. Comput. Integr. Manufact.* 7(1):3–16, 1994.
34. Ojelanki, K. N., and Delvin, A. G. Enterprise modeling for CIM information systems architectures: An object-oriented approach. *Comput. Indust. Engrg.* 26(2):279–293, 1994.
35. Olsen, K. A. *et al.* A procedure-oriented generic bill of materials. *Comput. Indust. Engrg.* 32(1):29–45, 1997.
36. Pandey, P. C., and Hasin, M. A. A. A scheme for an integrated production planning and control system. *Int. J. Comput. Appl. Technol.* 8(5/6):301–306, 1995.
37. Park, H. G. *et al.* An object oriented production planning system development in ERP environment. *Comput. Indust. Engrg.* 35(1/2):157–160, 1998.
38. Pels, H. J., and Wortmann, J. C. Modular decomposition of integrated databases. *Production Planning and Control* 1(3):132–146, 1990.
39. Ramalingam, P. Bill of material: A valuable management tool: Part I. *Indust. Manage.* 24(2):28–31, 1982.
40. Ramalingam, P. Bill of material: A valuable management tool: Part II. *Indust. Manage.* 25(1):22–25, 1983.
41. Ranganathan, V. S., and Ali, D. L. Distributed object management: Integrating distributed information in heterogeneous environment. *Comput. Indust. Engrg.* 25(1–4):317–320, 1993.
42. Ranky, P. G. *Manufacturing Database Management and Knowledge-Based Expert Systems.* CIMware, Addingham, England, 1990.
43. Seymour, L. *Data Structures,* Schaum's Outline Series in Computers. McGraw–Hill, New York, 1997.
44. Sinha, D., and Chen, H. G. Object-oriented DSS construction for hierarchical inventory control. *Comput. Indust. Engrg.* 21(1–4):441–445, 1991.
45. Smith, S. B. *Computer-Based Production and Inventory Control.* Prentice–Hall, Englewood Cliffs, NJ, 1989.
46. Smith, J., and Joshi, S. Object-oriented development of shop floor control systems for CIM. In *Proceedings of the International Conference on Object-Oriented Manufacturing Systems,* Calgary, Alberta, Canada, 3–6 May, 1992, pp. 152–157.
47. Spooner, D., Hardwick, M., and Kelvin, W. L. Integrating the CIM environment using object-oriented data management technology. In *Proceedings of the International Conference on Computer Integrated Manufacturing,* Rensselaer Polytechnic Institute, Troy, New York, May 23–25, 1988, pp. 144–152.
48. Starmer, C., and Kochhar, A. K. Fourth generation languages based manufacturing control systems—Lessons from an application case study. *Production Planning and Control* 3(3):271–279, 1992.

49. Trappey, A. J. C. *et al.* An object-oriented bill of materials system for dynamic product management. In *Proceedings of the 3rd International Conference on Computer Integrated Manufacturing,* Singapore, 11–14 July, 1995, pp. 629–636.
50. Ulrich, H., and Durig, W. Object-oriented realization of planning tools for production planning and control. In *Proceedings of the International Conference on Object-Oriented Manufacturing Systems,* Calgary, Alberta, Canada, 3–6 May, 1992, pp. 158–167.
51. Van Veen, E. A., and Wortmann, J. C. Generative bill of materials processing systems, *Production Planning and Control* 3(3):314–326, 1992.
52. Van Veen, E. A., and Wortmann, J. C. New developments in generative BOM processing systems. *Production Planning and Control* 3(3):327–335, 1992.
53. Wang, C.-Y., and Fulton, R. E. Information system design for optical fiber manufacturing using an object-oriented approach. *Int. J. Comput. Integr. Manufact.* 7(1):61–73, 1994.
54. Winter, R. E. A database approach to hierarchical materials planning. *Int. J. Oper. Production Manage.* 10(2):62–83, 1990.
55. Wortmann, J. C. Flexibility of standard software packages for production planning and control. *Production Planning and Control* 3(3):290–299, 1992.
56. Zhou, Q. *et al.* An information management system for production planning in virtual enterprises. *Comput. Indust. Engrg.* 35(1/2):153–156, 1998.
57. Ngwenyama, O. K., and Grant, D. A. Enterprise modeling for CIM Information systems architectures. An object-oriented approach. *Comput. Indust. Engrg.* 26(2):279–293, 1994.
58. Ng, J. K. C., and Ip, W. H. The strategic design and development of ERP and RTMS. *Comput. Indust. Engrg.* 34(4):777–791, 1998.

13

DEVELOPING APPLICATIONS IN CORPORATE FINANCE: AN OBJECT-ORIENTED DATABASE MANAGEMENT APPROACH

IRENE M. Y. WOON
MONG LI LEE

School of Computing, National University of Singapore, Singapore 117543

Financial information systems has evolved from spreadsheets to simple database systems where data are stored in a central repository, to today's sophisticated systems that integrate databases with complex applications. The object-oriented paradigm promises productivity and reliability of software systems through natural modeling concepts and cleaner and extensible designs while database management systems offer necessary capabilities such as persistence, integrity, and security. In this paper, we propose a unified framework for modeling both financial information and policies. A financial information system can be modeled and implemented using the object-oriented model. This forms the basis for exploratory and complex business data analysis and strategic decision-making support. In addition, we will illustrate that the object-oriented approach also provides a uniform representation for expressing financial policy formulation.

Database and Data Communication Network Systems, Vol. 2

I. INTRODUCTION

In the early 1980s, a number of integrated financial spreadsheets–graphics–word processing, e.g., Symphony and Excel, were used to capture, analyze, and present the financial information of a company. The main problem with such packages was their inability to automatically update links; when financial data in one spreadsheet were changed, then many related spreadsheets would have to be updated manually by the user to ensure the correctness of the financial reports. As a consequence, interest in databases as a medium for storing financial data grew.

As the amount of data multiplies, the many features offered by a database management system (DBMS) for data management, such as reduced application development time, concurrency control and recovery, indexing support, and query capabilities, become increasingly attractive and ultimately necessary. In addition, consolidating data from several databases, together with historical and summary information can create a comprehensive view of all aspects of an enterprise. This is also known as data warehousing. The data warehouse facilitates complex and statistical analysis such as:

- On-line analytic processing (OLAP), which supports a multidimensional model of data and *ad hoc* complex queries involving group-by and aggregation, and
- Exploratory data analysis or data mining where a user looks for interesting patterns in the data.

Systems that were specifically developed for use by the financial community included SAP, Peoplesoft, and Oracle Financials, which had Oracle or Sybase as its data repository.

Research into artificial intelligence provided the impetus for the development of products that embed human expertise into financial software programs. The expert's knowledge was typically represented in the form of a production rule in the form: observation → hypothesis. FINSIM Expert [1] is one such system used to analyze financial information for various purposes, e.g., credit analysis and investment analysis. Real-world "expert" systems would typically marry a traditional database with a knowledge base in which was encoded the expert's cognitive processes, to allow access to financial data. This was the early precursor to deductive databases. Research into deductive databases is aimed at discovering efficient schemes for uniformly representing assertions and deductive rules and for responding to highly expressive queries about the knowledge base of assertions and rules.

In many application domains, complex kinds of data must be supported. Object-oriented (OO) concepts have strongly influenced efforts to enhance database support for complex types. This has led to the development of object database systems. An important advantage of object database systems is that they can store code as well as data. Abstract data type methods are collected in the database, and the set of operations for a given type can be found by querying the database catalogs. Operations can be composed in *ad hoc* ways in query expressions. As a result, the object-oriented DBMS is like a software repository, with built-in query support for identifying software modules and

combining operations to generate new applications. Data consistency is preserved as the DBMS provides automatic management of constraints, which include user-defined operations.

The features of the financial domain that make this technique appealing are:

- Financial data have a relationship to one another, which can be naturally expressed in the object-oriented modeling notation.
- Within the framework of generally accepted financial accounting principles, accountants are free to choose from a number of alternative procedures to record the effects of an event. The encapsulation feature in the object-oriented approach allows these different procedures to be modeled such that the applications used are not affected.
- Financial data requirements vary for different industries and for different companies within the same industry, which is also supported by the encapsulation feature in the object-oriented approach.

The remainder of this paper is organized as follows. Section II gives a quick tutorial in finance covering various relevant aspects such as financial statements and their definition and use such as in implementing financial policies. This will give the reader an appreciation of the domain area with a view to understanding fragments of the model shown in later sections. Section III provides an introduction to database management systems covering its components, capabilities, and data models. Readers well versed in finance and/or databases may choose to skip the appropriate section(s). In Section IV, we show how the data in the financial statements may be modeled using the OO paradigm. In addition, we demonstrate how financial policies, which can be expressed in terms of financial statement information, may be modeled.

II. FINANCIAL INFORMATION AND ITS USES

Peter and Jane thought they could make a lot of money selling musical shoes. They soon found out, however, that to make money, they first had to have money. For a start, they needed money to buy machinery that would make these shoes from raw materials of leather, plastic, rubber, bells, sound cards, etc., money to buy the raw materials, and money to rent premises to house the shoe-making machinery. They decided to cut manpower costs by doing everything themselves. Money from their piggy banks was not enough, so they decided to approach some banks for a loan. Luckily for them, Peter's uncle was the Chairman of the Board of a small bank—so they received a small loan at very favorable rates. Peter and Jane were ecstatic—with this loan, they could start making their shoes and selling it. To cut down on marketing costs, they settled on buying a second-hand van and selling their shoes door-to-door.

In the language of finance, Peter and Jane have made investments in assets such as inventory of raw materials, shoes, and machinery. The loan they have taken out is referred to as a liability; they represent obligations that would have to be met, i.e., the loan must be repaid. They expect to do this by selling the shoes at a profit. Business grew beyond their expectations. To meet demands for their product, they had to expand their operations, buying more equipment, renting

TABLE 1 Peter and Jane's Balance Sheet as at 27.03.1999

Assets	(£000s)	Liabilities	(£000s)
Current		*Current*	
Cash	100	Accounts Payable	300
Accounts Receivables	400	**Total current liabilities**	300
Inventories	500		
Total current assets	1000	*Long-term*	
		Bank Loans	1700
Fixed		**Total long-term liabilities**	1700
Net Value of Equipment	1000		
Net Value of Vans	1000	**Shareholder's fund**	1000
Total fixed assets	2000		
Total assets	3000	**Total liabilities**	3000

bigger premises, hiring more workers, etc. To finance all these investments, they decided to incorporate their business and float some shares to the public, i.e., in return for funds, members of the public became joint owners of the firm.

The value of Peter and Jane's business can be recorded and reflected in simple financial models. We will look at three of these: the balance sheet, the income statement, and the cash flow statement.

A. Financial Statements

Table 1 shows Peter and Jane's balance sheet. The balance sheet gives a financial picture of the business on a particular day.

The assets that are shown on the left-hand side of the balance sheet can be broadly classified as current and fixed assets. Fixed assets are those that will last a relatively long time, e.g., shoe-making machinery. The other category of assets, current assets, comprises those that have a shorter life-span; they can be converted into cash in one business operating cycle, e.g., raw materials, semi-finished shoes/parts of shoes, and unsold finished shoes. An operating cycle is defined as a typical length of time between when an order is placed and when cash is received for that sale. Whether an asset is classified as current or fixed depends on the nature of the business. A firm selling vans will list it (a van) as a current asset, whereas Peter and Jane using vans to deliver goods will list it as a fixed asset. The liabilities represents what the firm owes, e.g., bank loans and creditors, and this is shown on the right-hand side of the balance sheet. Just as assets were classified based on their life span, so are liabilities. Current liabilities are obligations that must be met in within an operating cycle while long-term liabilities are obligations that do not have to be met within an operating cycle. Also shown on the right-hand side of the balance sheet is the shareholder's fund, e.g., money received from the issuance of shares.

The balance sheet shows the values of fixed assets at its net value, i.e., the purchase price less accumulated depreciation. Depreciation reflects the accountant's estimate of the cost of the equipment used up in the production process. For example, suppose one of the second-hand vans bought at £500 is estimated to have 5 more useful years of life left and no resale value beyond that.

TABLE 2 Peter and Jane's Income Statement for 1.7.1998–30.3.1999

	(£000s)
Revenues	
Sales	3000
Expenses	
Cost of goods sold	(1800)
Admin and selling expenses	(500)
Depreciation	(200)
Profit/loss	
Earnings before interest and taxes	500
Interest expense	(100)
Pretax income	400
Taxes	(150)
Net income	250

According to the accountant, the £500 cost must be apportioned out as an expense over the useful life of the asset. The straight-line depreciation method will give £100 of depreciation expense per year. At the end of its first year of use, the van's net value will be £500 – £100 = £400 and at the end of its second year of use, its net value will be £500 – £200 = £300, and so on.

Table 2 gives the income statement of Peter and Jane's business for the past year. Rather than being a snapshot of the firm at a point in time, the income statement describes the performance of the firm over a particular period of time, i.e., what happened in between two periods of time. It shows the revenues, the expenses incurred, and the resulting profit or loss resulting from its operations. Details of income and expenses shown in the statement differ according to the firm, industry, and country.

The last financial model is called the cash flow statement. This shows the position of cash generated from operations in relation to its operating costs. Consider some typical cash flows: cash outflows in the purchase of raw materials, payment of utilities and rents, cash inflows when debtors pay or bank loans are activated. This statement shows the operating cash flow, which is the cash flow that comes from selling goods and services. This should usually be positive; a firm is in trouble if this is negative for a long time because the firm is not generating enough cash to pay its operating costs. Total operating cash flow for the firm includes increases to working capital, which is the difference between the current assets and current liabilities. The total operating cash flow in this case is negative. In its early years of operations, it is normal for firms to have negative total cash outflows as spending on inventory, etc., will be higher than its cash flow from sales.

Profit as shown in the income statement is not the same as cash flow. For instance, sales on the income statement will tend to overstate actual cash inflows because most customers pay their bills on credit. The cash flow statement shown in Table 3 is derived from the balance sheet and the income statement. In determining the economic and financial condition of a firm, an analysis of the firm's cash flow can be more revealing.

TABLE 3 Peter and Jane's Cash Flow Statement for 1.7.1998–30.3.1999

Cash flows from operations	(£000s)
Earnings before interest and taxes	500
Depreciation	200
Taxes	(150)
	450
Increases to working capital	(700)
Total operating cash flows	(250)

B. Using the Financial Statements

Managers and shareholders of a firm rely on the financial information given in the financial statements to perform many tasks such as reporting profit, computing tax, evaluating investment options, analyzing performance, appraising the values of assets, and designing financial strategies. However, financial information is only useful when they are interpreted correctly. A variety of standard techniques can be employed to do this. Typically, the trend of absolute historical figures, the trend of certain ratios (the proportion of one financial variable to another), and the sources and uses of funds are analyzed.

An analysis of the periodical series of absolute financial figures is undertaken in order to draw attention to changes that have taken place, identifying trends and regular changes due to trade cycles, and irregular fluctuations which denote instability and greater risks. This initial analysis provides a rough indicator of the condition of the firm. The analysis of a firm's ratios involves a more detailed scrutiny of the financial statements. There are several reasons why ratio analysis is used. Isolated figures mean very little on their own. For example, does an increase in sales indicate that the firm is better off? Surely not, if its costs are increasing on a larger scale. Thus, significant pieces of data need to be logically related, if their meaning is to be interpreted correctly. Ratio analysis also allows for meaningful comparisons within a firm over time, against other firms and against the industry as a whole. Individual ratios are grouped into different categories, each representing a financial concept, such as liquidity, profit, risk, and growth, that is deemed relevant in characterizing a firm's financial condition and performance. Comparing a firm's ratios against those for the industry as a whole (industry average) allows one to judge whether the firm is adequate in some area. There are various categorization schemes and some variation in the definition of the ratios. For instance, some authors, e.g., Van Horne [2] recommend that the inventory turnover ratio be computed from the cost of goods sold and average inventory while others, e.g., Miller [3] prefer to use sales and end-of-year inventory values.

Typical ratio categories are:

1. *Liquidity ratios* indicate a firm's ability to meet its daily obligations. It is possible for a firm to be profitable and yet fail if it is unable to settle its short-term debts. Examples are the ratio of current assets to current liabilities, the

ratio of all current assets except inventory to current liabilities, and the ratio of cost of goods sold to average inventory.

2. *Activity ratios* are constructed to measure how effectively a firm's assets are being managed. The idea is to find out how quickly assets are used to generate sales. Examples are the ratio of total operating revenues to average total assets, and the ratio of total operating revenues to average receivables.

3. *Debt ratios* show the relative claim of creditors and shareholders on the firm and thus indicate the ability of a firm to deal with financial problems and opportunities as they arise. A firm highly dependent on creditors might suffer from creditor pressure, be less of a risk-taker, and have difficulty raising funds; i.e., it will have less operating freedom. Examples are the ratio of total debt to total shareholders' fund, the ratio of total debt to total assets, and the ratio of total debt to total shareholders' fund.

4. *Coverage ratios* are designed to relate the financial charges of a firm to its ability to service them, e.g., the ratio of interest to pretax profit.

5. *Profitability ratios* indicate the firm's efficiency of operation. Examples of ratios measuring profitability are the ratio of profit to sales, the ratio of profit to total assets, and the ratio of profit to shareholders' funds.

C. Financial Policies

The total value of a firm can be thought of as a pie (see Fig. 1). Initially, the size of the pie will depend on how well the firm has made its investment decision. The composition of the pie (usually known as capital structure) depends on the financing arrangements. Thus, the two major classes of decisions that financial managers have to make are investment and financing.

A major part of the investment decision is concerned with evaluating long-lived capital projects and deciding whether the firm should invest in them. This process is usually referred to as capital budgeting. Although the types and proportions of assets the firm needs tends to be determined by the nature of the business; within these bounds, there will many acceptable proposals that the firm may need to consider. Many rules can be employed to decide which ones to select, e.g., NPV rule, payback rule, the accounting rate of return rule, internal rate of return rule, profitability index, and capital asset pricing model.

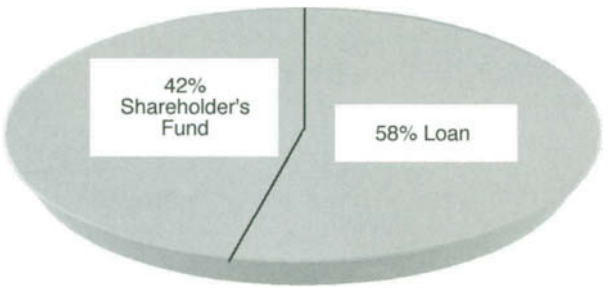


FIGURE 1 Pie model of a firm.

After a firm has made its investment decisions, it can then determine its financing decisions. This is what Peter and Jane have done—they identified their business, worked out what they needed, and then looked for ways to raise money for what they needed to start up the business. Thus, financing decisions are concerned with determining how a firm's investments should be financed. In general, a firm may choose any capital structure it desires: equity, bonds, loans, etc. However, the capital structure of the firm has a great impact on the way in which the firm pays out its cash flows. There are many theories put forward to suggest the optimal capital structure of a firm [4,5].

From the examination of the domain area of investment and financing decisions, it can be seen that there are many theories and methods supporting various arguments as to what policies financial managers should pursue to ensure that the firm achieves growth and profitability. Each theory/method has its own assumptions, shortcomings, and advantages. This paper will not attempt to model any of these theories or methods. Instead it will look at the implementation of these policies and demonstrate how they can be modeled so that the object-oriented implementation can be automatically generated.

Some examples of policy implementations are:

- Capital budgeting (investment).
    1. Fund capital expenditures from profit only.
    2. Expected return on investment >= X%.
    3. Total operating revenues to average inventory ratio = $X \ldots Y$.
    4. The market value of any individual stock cannot exceed X% of total account.

- Capital structure (financing).
    1. Debt <= X times share capital.
    2. Interest expense > X times of operating cash flow.
    3. Debt–equity ratio = $X \ldots Y$.
    4. Liquidity ratio >= X.

Policies are thus implemented as rules of thumbs, giving targets to be achieved or indicators to monitor. These targets could be expressed as values, ranges of values, and upper and lower bounds of values that ratios or variables must have. These targets could be absolutes or benchmarked to some other ratio/variable in a firm and could be derived from comparison with some industry average or derived from comparison with the past year's data, etc. They have an impact of increasing/decreasing funds flow within the firm by controlling investments made or changing the financial structure of their firm.

In this section, we have shown that financial policies are expressed as target values for ratios or financial variables to satisfy. These values are derived from the financial statements, which in turn record and assign values to events and operations that occur within a firm (such as accepting orders, producing goods, and delivering goods). This section sets the scene for the rest of the paper and allows the reader to appreciate examples given in subsequent sections.

III. DATABASE MANAGEMENT SYSTEMS

Given the rapid growth of data and the wide recognition of data as a valuable organizational asset, it is increasingly important for an organization to manage the amount of data and extract useful information in a timely manner. Databases and database technology have become indispensable in many areas where computers are used such as business, engineering, medicine, law, education, and library science. A database is basically a collection of data that describes the activities of one or more related organizations. For example, a sales database may contain information about customers (such as their customer number, name, address, person to contact, credit limit), products (product code, description, price, amount sold), and retail outlets (name, monthly turnover). Relationships, such as which products are popular with particular customers and retail outlets, may also be captured in the database.

A DBMS is a software designed to facilitate the processes of defining, constructing, and manipulating databases for various applications. Defining a database involves specifying the types of data to be stored in the database, together with detailed descriptions of each type of data (also known as the meta-data). Constructing the database involves storing the data itself on some storage medium that is controlled by the DBMS. Manipulating a database includes querying the database to retrieve specific data (such as "retrieve the products sold by outlet A"), updating the database to reflect changes in the miniworld (such as "increase the price of new product B"), and generating reports from the data.

The data in a DBMS is traditionally viewed at three levels:

- *Internal level.* Data are organized physically on a storage medium, which involves the ordering and encoding of field values, the ordering and size of records, and access methods or indexes of the data.
- *Conceptual level.* This is the data modeling level, which captures the abstract representation of the data in a schema.
- *External level.* Applications at this level may impose additional constraints on the data and their semantics.

The process of creating a database for an application typically goes top-down through these three levels, from the external to the conceptual, and then to internal designs.

A. Components of a DBMS

Figure 2 shows the architecture of a typical DBMS. Records of data and meta-data are usually stored on storage devices such as disks and tapes. The disk space manager is responsible for managing the available disk space in units of a page (4 Kbytes or 8 Kbytes). Requests to allocate, deallocate, read, and write pages are carried out by this layer. When a data record is needed for processing, it is fetched to main memory from the disk. The file manager determines the page on which the record resides; access methods or indexes may be used to facilitate fast identification of the required page. The buffer manager, which partitions the available memory into a collection of pages called a buffer pool,

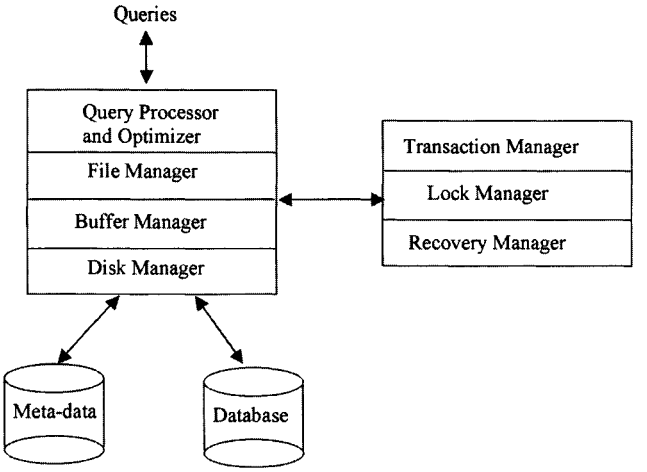


FIGURE 2 Components of a DBMS.

fetches the requested page to the buffer from the disk. When a user issues a query, the query optimizer uses information about how the data are stored to generate an efficient execution plan.

In order to provide concurrency control and crash recovery in the DBMS, the disk space manager, buffer manager, and file manager must interact with the transaction manager, lock manager, and recovery manager. The transaction manager ensures that transactions (or user programs) request and release locks on data records according to some protocol and schedules the transaction executions. The lock manager keeps track of requests for locks while the recovery manager maintains a log of changes and restores the system to a consistent state after a crash.

B. Capabilities of a DBMS

Using a DBMS to manage data offers many advantages. These include:

- *Persistence.* Large volumes of data can be managed systematically and efficiently as a DBMS utilizes a variety of sophisticated techniques to store and retrieve data. Persistence due to permanent storage of data is important to many applications.
- *Data independence.* The DBMS provide an abstract view of the data, which insulate application programs from the details of data, representation, and storage.
- *Control of data redundancy.* When several users share the data, centralizing the administration of data can minimize redundancy, without which an undesirable inconsistency in data and wastage of storage space can occur.
- *Data integrity and security.* The DBMS can enforce compliance to known constraints imposed by application semantics. Furthermore, it can restrict access to different classes of users.
- *Data availability and reliability.* In the event of system failure, the DBMS provides users access to as much of the uncorrupted data as possible. It also has the ability to recover from system failures without losing data (crash recovery).

The DBMS provides correct, concurrent access to the database by multiple users simultaneously.

- *High-level access.* This is provided by the data model and language which defines the database structure (also known as schema) and the retrieval and manipulation of data.
- *Distribution.* Multiple databases can be distributed on one or more machines, and viewed as a single database is useful in a variety of applications. This feature is crucial especially when databases in the different departments of an organization may have been developed independently over time and the management now needs an integrated view of the data for making high-level decisions.

This list of capabilities is not meant to be exhaustive but it serves to highlight the many important functions that are common to many applications accessing data stored in the DBMS. This facilitates quick development of applications that are also likely to be more robust than applications developed from scratch since many important tasks are handled by the DBMS instead of being implemented by the application.

C. Data Models

A central feature of any DBMS is the data model upon which the system is built. At the conceptual level of a DBMS architecture, the data model plays two important roles. First, the data model provides a methodology for representing the objects of a particular application environment, and the relationships among these objects (the conceptual or semantic role). Second, the data model is structured to allow a straightforward translation from the conceptual schema into the physical data structures of the internal level of the DBMS (the representational role).

Many commercial DBMS today such as DB2, Informix, Oracle, and Sybase are based on the relational data model proposed by [6]. At that time, most database systems were based on either the hierarchical model [7] (IBM's IMS and SYSTEM-2000) or the network model [8] (IDS and IDMS). The hierarchical and network models are strongly oriented toward facilitating the subsequent implementation of the conceptual schema. This is because, historically, the physical structure of a DBMS was designed first, and then a data model was developed to allow conceptual modeling on the particular physical design. Thus, the hierarchical data model is based on underlying tree-oriented data structures, while the network model is based on ring-oriented data structures. The use of the hierarchical and network data models in the semantic role is thus burdened with numerous construction roles and artificial constraints.

On the other hand, the relational data model is simple and elegant: a database is a collection of one or more relations, and each relation is a table with rows and columns. The tabular representation of data allows the use of simple, high-level languages to query the data. Despite its attractive simplicity, however, the relational model must be enhanced to fulfil the two roles of a conceptual level data model. In pursuit of discovering semantic enhancements to the relational model, a rich theoretical foundation about data dependencies and normalization was produced [9].

Database design is a complex process, and research into database design methods has developed the idea of using two distinct data models at the conceptual level. An enhanced conceptual data model would provide an effective means of describing the database application environments. A representational data model would be employed for efficient translation of a schema into physical data structures. Thus, the hierarchical, network, or relational data model could be employed as a representational data model. The DBMS would provide a user-invisible translation between the conceptual schema and the representational schema. Therefore, database applications are able to declare and reference data as viewed in the conceptual schema. Conceptual data modeling of an application is often carried out using semantic data models such as the object-oriented model and the entity–relationship (ER) model. Note that the object-oriented model is used in Objectstore and Versant, but there are no database systems that support the ER model directly, although an ER model description of data can be translated into a collection of relations if we want to use a relational DBMS.

The ER approach [10] incorporates the concepts of entity types and relationship sets, which correspond to structures naturally occurring in information systems. An entity is an object that exists in our minds and can be distinctly identified. Entities can be classified into different entity types; each entity type contains a set of entities, each satisfying a set of predefined common properties or attributes. A relationship is an association among two or more entities. Relationships can be classified into different relationship sets; each relationship set contains a set of relationships, each satisfying a set of attributes. The structure of a database organized according to the ER approach can be represented by a diagrammatic technique called an entity–relationship diagram (ERD).

The objected-oriented (OO) paradigm gained prominence in the 1990s. OO concepts such as classes and inheritance are a powerful tool for modeling the real world. The advantages of relationship modeling, extensibility and easy maintenance provided by this technology, were quickly recognized by the database community. This led to the development of object-oriented databases. An object-oriented database combines object-orientation with database capabilities. Opinion is divided, however, as to what particular features and characteristics it should possess. Atkinson *et al.* [11] give the mandatory features that define an object-oriented database system. From the DBMS perspective, it must have persistence, secondary storage management, concurrency, recovery, and *ad hoc* query facility. From the OO perspective, it must support complex objects, identity, encapsulation, types or classes, inheritance, overriding with late binding, extensibility, and computational completeness. Note that some of these features deal with implementation and performance issues rather than facet modeling.

IV. FINANCIAL OBJECT-ORIENTED DATABASES

As financial information is so crucial in running a business, there has always been an interest in developing data models [12–14] that will capture the key characteristics of the domain. In this section, we will introduce some

fundamental OO modeling concepts. Subsequently, we will apply these concepts to the financial domain. A comprehensive exposition of the OO method and its modeling and application can be found in [15,16].

A. General Concepts

The object-oriented approach views software as a collection of discrete objects that incorporates both structure (attributes) and behavior (operations). This is in contrast to conventional software development where data and behavior are loosely connected.

An object is defined as a concept, abstraction, or tangible object with well-defined boundaries and meaning for the problem at hand. Examples of objects are customer John Doe, Order No 1234, or a pair of shoes stock no 12345 B-PRI. All objects have an identity and are distinguishable. This means that two objects are distinct even if they have the same attributes, e.g., two pairs of shoes of the same color, style, and weight with the same stock number. The term "identity" means that objects are distinguished by their inherent existence and not by the descriptive properties they may have.

An object is an encapsulation of attributes and operations. Encapsulation separates the external interface of an object from the internal implementation details of the object. The external interface is accessible to other objects and consists of the specifications of the operations that can be performed on the object. The operations define the behavior of the object and manipulate the attributes of that object. The internal implementation details are visible only to the designer. It consists of a data section that describes the internal structure of the object and a procedural section that describes the procedures that implement the operations part of the interface. This means that the implementation of an object may be changed without affecting the applications that use it. For example, the cost of a pair of shoes can be measured by different methods, such as average cost, LIFO, and FIFO. If the firm decides to change its costing method for shoes, the encapsulation feature ensures that no changes will have to be made to any application that requires the unit cost for a pair of shoes.

A class is an abstraction that describes properties that are important to the application. Each class describes a set of individual objects, with each object having its own value for each attribute but sharing the attribute names and operations with other objects in the class. The objects in a class not only share common attributes and common operations, they also share a common semantic purpose. Even if a van and a shoe both have cost and size, they could belong to different classes. If they were regarded as purely financial assets, they could belong to one class. If the developer took into consideration that a person drives a van and wears shoes, they would be modeled as different classes. The interpretation of semantics depends on the purpose of each application and is a matter of judgement.

Figure 3 shows that objects such as Leather and Shoe can be abstracted into an Inventory Item class with attributes, Description, Quantity on Hand, and Unit Cost. Operations that manipulate the attributes include Issue Stock (which reduces the Quantity on Hand when the raw material is required for the

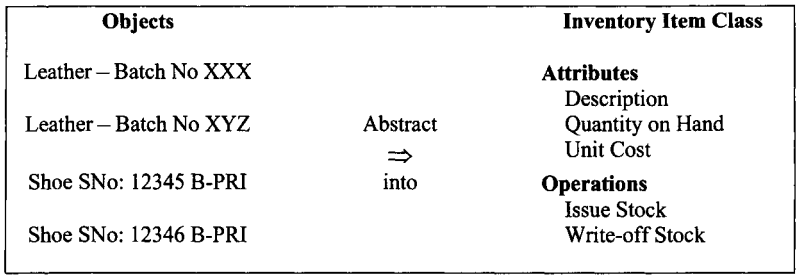


FIGURE 3 Objects and classes.

assembly process), Write-off Stock (which sets the value of Quantity on Hand to 0 when the goods are considered damaged and unusable).

Note that the Inventory Item Class includes raw materials (as in leather and rubber) and finished products (shoes). Hence we can design classes such as Raw Materials and Products, which are refinements of Inventory Item.

B. Object Modeling

To define an object model for any domain, the following logical activities will have to be carried out:

1. Identify the objects and classes and prepare a data dictionary showing the precise definition of each.
2. Identify the association and aggregation relationships between objects.
3. Identify the attributes of the objects.
4. Organize and simplify the object classes through generalization.

These activities are logical as in practice; it may be possible to combine several steps. In addition, the process of deriving an object model is rarely straightforward and usually involves several iterations.

The object modeling technique consists of three models, each representing a related but different viewpoint of the system:

1. the object model, which describes the static structural aspect,
2. the dynamic model, which describes the temporal behavioral aspect, and
3. the functional model, which describes the transformational aspect.

Each model contains references to other models; e.g., operations that are attached to objects in the object model are more fully expanded in the functional model. Although each model is not completely independent, each model can be examined and understood on its own.

1. Object Model

The object model describes the structure of the objects in the system—their identity, their relationships to other objects, their attributes, and their operations. The object model is represented graphically with object diagrams

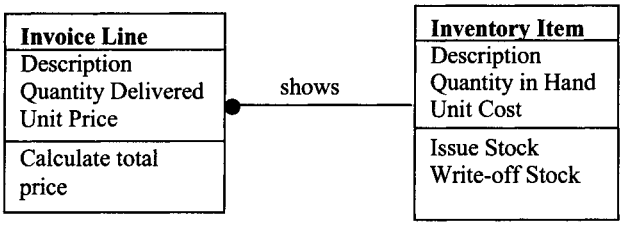


FIGURE 4 Association relationship.

containing object classes. Classes are arranged into hierarchies sharing common structure and behavior and are associated with other classes. Classes define attribute values carried by each object and the operations that each object performs or undergoes. It is thus the framework into which the dynamic and functional model may be placed.

Objects and object classes may be related to other objects in several ways. They could be dependent on one another in which case the relationship is known as an *association.* Associations may be binary or tenary or of higher order and are modeled as bidirectional. These links may also express the multiplicity of the relationship, either 1-1, 1-to-many, or many-to-many. For example the association between an Inventory Item and an Invoice Line is 1-to-many because an inventory item may appear on several invoice lines. The solid ball in Fig. 4 denotes the multiplicity of the Inventory Item objects in the Invoice Line class.

The second type of relationship is *aggregation,* which expresses a "part–whole," or a "part-of" relationship in which objects representing the components of something are associated with an object representing the entire assembly. An obvious example from Peter and Jane's firm is the components that go into making a shoe, e.g., leather, rubber, bells, and thread. Another example is shown in Fig. 5, where an invoice is composed of its individual invoice lines. A small diamond drawn at the assembly end of the relationship denotes aggregation. The Invoice class has a Status attribute which gives the state of the object. We will elaborate on this in the next section on dynamic models.

Generalization is the relationship between a class and its more refined classes. It is sometimes referred to as a "is-a" relationship. For example, Current Asset is the generalization for Cash, Inventory Item, and Accounts Receivables. Current Asset is called the superclass while Cash, Inventory Item, and

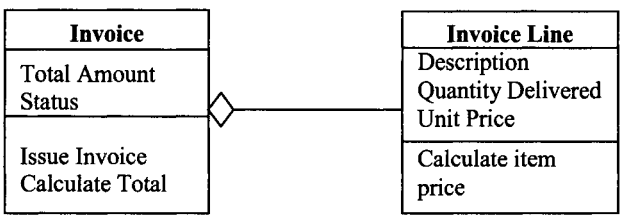


FIGURE 5 Aggregation relationship.

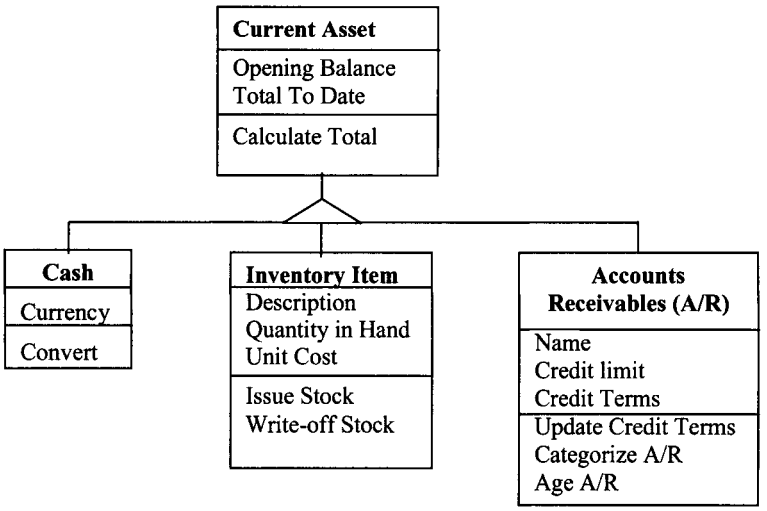


FIGURE 6 Generalization relationship.

Accounts Receivables are called subclasses. Each subclass inherits all the attributes and operations of its superclass in addition to its own unique attributes and operations. Figure 6 shows how this relationship is depicted. The subclass Accounts Receivables has attributes Name, Credit Limit, and Credit Terms for its customers and operations Update Credit Terms and Categorize A/R, which determines good and bad payers. In addition to its own class attributes and operations, objects of Account Receivables will also inherit the attributes and operations of Current Asset.

This ability to factor out common attributes of several classes into a common class and to inherit the properties from the superclass can greatly reduce repetition within design and programs and is one of the advantages of object-oriented systems.

2. Dynamic Model

The dynamic model describes those aspects of a system concerned with time and order of operations. It captures the control aspect of the system that describes the sequences of operations as they occur regardless of what the operation does, what they operate on, and how they are implemented. The dynamic model is represented graphically with state diagrams. Each state diagram shows the state and the event sequences permitted in a system for one class of objects. Figure 7 shows the state diagram for an invoice. When a customer takes goods on credit, he will be issued an invoice. At this point in time, the invoice is deemed to be outstanding. At some point in time, the customer may pay. If payment is not received within the credit period granted to him, the invoice is said to be overdue. When the customer makes full payments, the accounts will be adjusted to reflect this (i.e., the customer's debts are reduced and the amount of cash increases), and the customer will be issued a receipt. The firm might decide to write-off the invoice if the customer subsequently goes bankrupt or it becomes uneconomical to pursue the customer for paying.

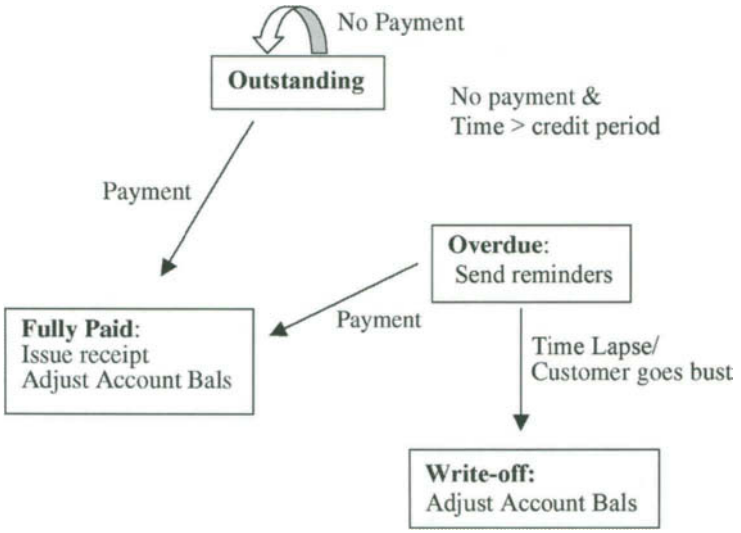


FIGURE 7 State diagram for an invoice.

3. Functional Model

The functional model describes those aspects of a system concerned with transformation of values—functions, mapping, constraints, and functional dependencies. The functional model captures what the system does regardless of how and when it is done. The functional model may be represented with data flow diagrams that show the dependencies between the inputs and the outputs to a process. Functions are invoked as actions in the dynamic model and are shown as operations on objects in the objection model. An example of an operation on Invoice (see Fig. 7) is "Issue receipt." The inputs to this process are the invoice details, such as its number and customer name, and its corresponding payment details, and its output is the receipt that is sent to the customer.

C. Modeling Financial Policies

In the previous section, we have shown how the OO paradigm can be used to model financial information. Entities such as customers, customer orders, suppliers, requisition orders, and employees can be intuitively modeled as objects and classes. Based on these classes, events and operations that occur in the daily routine of running a firm such as Peter and Jane's can also be modeled. In this section, we will examine how financial policies may also be modeled using the OO approach.

We view financial data as a class that is a generalization of the Balance Sheet, Income statement, and Cash Flow Statement classes. The accounts' hierarchy of each of the financial statements can be directly mapped as superclasses and subclasses. For example, the Balance Sheet class is a superclass of the Asset, Liability, and Shareholders' Fund classes, and the Asset class is in turn a superclass of the Current Asset and the Fixed Assets classes. Financial policies can therefore be modeled as a class that interacts with the Financial Data class.

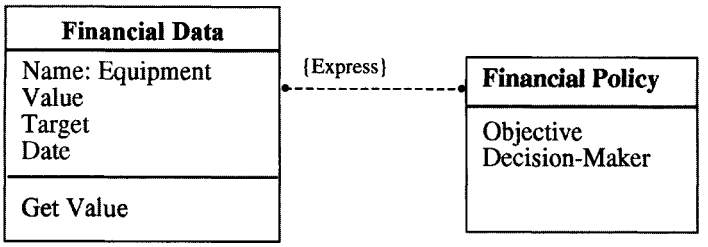


FIGURE 8 Constrained links and attributes.

However, to model financial policies, two other concepts are needed:

- Constraints on objects and attributes, and
- Derived objects and attributes.

Constraints are functional relationships between two or more components (objects, classes, attributes, links, and associations) of an object model. A constraint limits the value that a component can assume. Constraints on attributes are shown in braces, positioned near the constrained class. Constraints on classes are shown as a dotted line connecting the constrained class to the class it depends on. For example, to model a policy where the firm is only able to purchase new equipment if it has made profit in the previous years, we constrain the attribute Target of the financial class Equipment to the value "Financial Data. Value > 0" where "Financial Data. Value" denotes the attribute Value of the financial class Profit. This constraint may be explicit as shown in Fig. 8, or it may also be embedded in the operations (see next example). Since not all financial data are used to make policy decisions, but are only certain key indicators, we also have a constraint on the Financial Data class.

A derived object is defined as a function of one or more objects, which in turn may be derived. The derived object is completely determined by other objects and ultimately the derivation tree terminates with base objects. The notation for a derived object is a slash or diagonal line (on the corner of the box). For example, Financial Ratio is a derived class since its value is obtained from dividing one financial component by another. Note that these financial components may themselves be derived classes or financial data class. Figure 9

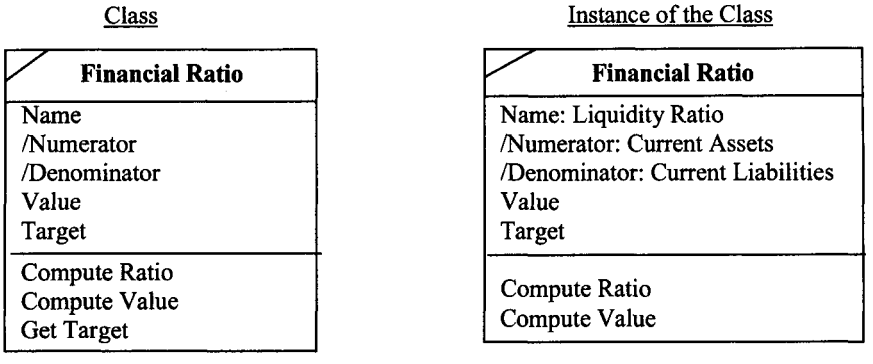


FIGURE 9 Derived classes and attributes: A class and an instance of the class.

shows the attributes and operations for a Financial Ratio class and an instance of this class.

Constraints on derived objects can be embedded in the operations. For example, most firms require their liquidity ratio to be within a certain range to ensure that they are well positioned to handle business emergencies. However, the value for the liquidity ratio is usually obtained by a trend analysis of the past 10 years or by comparing with the industry norm. This requires complex statistical computations and the constraint is better expressed procedurally in the operations such as Get Target in Fig. 9.

To conclude, we see that the OO approach provides a uniform and systematic way in which financial information and policies are modeled. This facilitates the automatic implementation of a financial database management system through the use of commercially available tools and products such as ObjectTeam and Intelligent OOA.

V. DISCUSSION

A database management system offers many capabilities and handles important tasks that are crucial to applications accessing the stored data. The object-oriented technology provides a framework for representing and managing both data and application programs. It is a promising paradigm to solve the so-called impedance mismatch: the awkward communication between a query language and a programming language that results when developing applications with a database system. We have seen how the financial domain can be sensibly modeled using object-oriented concepts. The object-oriented approach also removes the gap between an application and its representation. It allows the designer to model the real world as closely as possible, which can be mapped directly to design and implementation constructs.

Financial information can be viewed at two levels. At one level, we look at the raw data that are associated with routine events and operations such as sales, invoices, and payments. At another level, we abstract and generalize the raw financial data to analyze and interpret them. It is quite apparent from the examples given in Section IV that the basic financial data associated with the routine events and operations in a firm can be adequately represented by the OO modeling notation. Basic financial data associated with the routine events and operations in a firm can be abstracted to a higher level through the class hierarchy, which allows strategic decisions and policy formulations. For example, policy statements could be expressed in terms of financial ratios that are derived from financial statements, which are composed of basic financial data. The encapsulation feature proves useful as the composition of financial classes such as Asset and Liability varies for different industries. For example, the Current Asset class will have objects such as inventory items for the trading and manufacturing industry that will not exist in the service industry such as banks. Instead, the Current Asset class in the banking industry will have objects such as investment items, which include marketable securities, government bonds, and fixed deposits. In the same way, the key indicators used to implement financial policies may be modified without affecting the application

program. Object-oriented database systems are also extensible as we can extend its capabilities by adding new classes to the system when the need arises.

We can also model financial policies using rules and formal methods. Rules are predominantly used in expert systems and deductive databases. It is a problem-solving approach that uses rules to encode the heuristic and search strategies to infer new knowledge from the database of facts. The expression of the policy statement in rules may be quite natural but presents problems in updating especially when the number of rules grows. If we use formal methods [17], then each policy is precisely defined using some notations such as the Z notation and the Vienna Development Method. However, the tools support for the implementation of these specifications is inadequate. Furthermore, both these techniques are not suitable for modeling financial information. On the other hand, the OO approach focuses on objects and organizes domain knowledge around these objects and their relationships. It presents a seamless way for modeling financial information and policies.

REFERENCES

1. Klein, M., FINSIM Expert: A KB/DSS for financial analysis and planning. In *EUROINFO '88: Concepts for Increased Competitiveness* (H. J. Bullinger, E. N. Protonotaris, D. Bouwhuis, and F. Reim, Eds), 908–916. North-Holland, Amsterdam, 1988.
2. Van Horne, J. C. *Financial Management and Policy*. 11th ed. Prentice–Hall, Englewood Cliffs, NJ, 1998.
3. Miller, D. E. *The Meaningful Interpretation of Financial Statements: The Cause and Effect Ratio Approach*. Am. Management Assoc., New York, 1996.
4. Myers, S. C. The capital structure puzzle. *J. Finance*, 39:575–592, 1984.
5. Smith, C. Raising capital: Theory and evidence. In *The Revolution on Corporate Finance* (J. Stern and D. Chew, Jr, Eds.), 2nd ed. Blackwell, Oxford, 1992.
6. Codd, E. F. A relational model of data for large shared data banks. *Commun. ACM* 13(6): 377–387, 1970.
7. Tshichritzis, D. C., and Lohovsky, F. H. Hierarchical database management. *ACM Comput. Surveys* 8(1):105–123, 1976.
8. Taylor, R., and Frank, R. CODASYL database management systems. *ACM Comput. Surveys* 8(1):67–104, 1976.
9. Maier, D. *Theory of Relational Databases*. Computer Science Press, Rockville, MD, 1983.
10. Chen, P. P. The entity-relationship model: Toward a unified view of data. *ACM Trans. Database Systems* 1(1):166–192, 1976.
11. Atkinson, M., Bancilhon, F., Dewitt, D., Dittrich, K., Maier, D., and Zdonik, S. The object-oriented database system manifesto, deductive and object-oriented databases, 223–240. Elsevier Science, Amsterdam, 1990.
12. Lieberman, A. Z., and Whinston, A. B. A structuring of an event accounting information system. Accounting Rev. (April):246–258, 1975.
13. Haseman, W. D., and Whinston, A. B. Design of a multi-dimensional accounting system. *Accounting Rev.* (January):65–79, 1976.
14. Everest, G. C., and Weber, R. A relational approach to accounting models. *Accounting Rev.* (April): 340–359, 1977.
15. Booch, G. *Object-Oriented Analysis and Design with Applications*, 2nd ed. Addison-Wesley, Reading, MA, 1994.
16. Rumbaugh, J., Jacobson, I. and Booch, G. *The Unified Modeling Language Reference Manual*. Addison–Wesley, Reading, MA, 1999.
17. Woon, I. M. Y., and Loh, W. L. Formal derivation to object-oriented implementation of financial policies. *Int. J. Comput. Appl. Technol.* 10(5/6):316–326, 1997.

14

SCIENTIFIC DATA VISUALIZATION: A HYPERVOLUME APPROACH FOR MODELING AND RENDERING OF VOLUMETRIC DATA SETS

SANGKUN PARK

Institute of Advanced Machinery and Design, Seoul National University, Seoul 151-742, Korea

KUNWOO LEE

School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea

Database and Data Communication Network Systems, Vol. 2

I. INTRODUCTION

Scientific data visualization aims to devise algorithms and methods that transform massive scientific data sets into valuable pictures and other graphic representations that facilitate comprehension and interpretation. In many scientific domains, analysis of these pictures has motivated further scientific investigation.

A remarkable work on scientific visualization as a discipline started in 1987, which was reported by the National Science Foundation's Advisory Panel on Graphics, Image Processing, and Workstations [1]. The report justified the need for scientific visualization and introduced the short-term potential and long-term goals of visualization environments, and emphasized the role of scientific visualization in industrial fields.

The IEEE Visualization conference series has been a leading conference in scientific visualization since 1990, and its importance has grown within ACM's SIGGRAPH conference. Many researchers and practitioners get together at the annual Eurographics Workshop on Visualization in Scientific Computing. Similarly, hundreds of conferences and workshops in the world have developed the theme. Also numerous journals and books worldwide involved in computer science, physical science, and engineering now devote themselves, in part or in full, to the topic.

Recent work in scientific visualization has been stimulated by various conferences or workshops as described above. It mainly includes topics in volume visualization, volume graphics, and volume modeling. Also it also covers the following topics: multiresolution modeling, scattered data modeling, and feature segmentation. Now we will introduce key aspects of each topic below, commenting on the current status of work or requirements.

A. Volume Visualization

Volume visualization found its initial applications in medical imaging. The overall goal of volume visualization is to extract meaningful information from volumetric data. Volume visualization addresses the representation, manipulation, and rendering of volumetric data sets without mathematically describing surfaces used in the CAD community. It provides procedures or mechanisms for looking into internal structures and analyzing their complexity and dynamics. Recent progresses are impressive, yet further research still remains.

B. Volume Graphics

As an emerging subfield of computer graphics, volume graphics is concerned with the synthesis, manipulation, and rendering of 3D modeled objects, stored as a volume buffer of voxels [2]. It primarily addresses modeling and rendering geometric scenes, particularly those represented in a regular volume buffer, by employing a discrete volumetric representation. The primary procedure for its representation needs voxelization algorithms that synthesize voxel-based models by converting continuous geometric objects into their discrete voxel-based representation.

Volume graphics is insensitive to scene and object complexities, and supports the visualization of internal structures. However, there are several

problems associated with the volume buffer representation, such as memory size, discreteness, processing time, and loss of geometric representation. That is, volume graphics must consider such issues as geometric accuracy, minimality, effectiveness/efficiency, and representation and rendering quality (for example, 3D antialiasing). Nevertheless, by offering a comprehensive alternative to traditional surface graphics, volume graphics has the potential of developing into a major trend in computer graphics.

C. Volume Modeling

Volume modeling is often referred to by some authors as the process of identifying and synthesizing objects contained within a 3D data set, very regular and dense 3D image data obtained by a scanning instrument. The term "volume modeling" in a more general sense may be used to mean the methods for representing and modeling the attributes of 3D objects and their internal structures. It is worth noting the internal structures. Most geometric modeling methods have often assumed object interiors to be homogeneous. A solid modeling system is a typical example. Thus, the internal structure of an inhomogeneous object should be represented or modeled by a mathematical formulation or other techniques, which is a part of the major goals aimed for by volume modeling.

In the past several years, a tremendous amount of research and development has been directed toward these volume models, but there has been very little work on developing the volume models that feed a rendering pipeline for scientific visualization.

D. Multiresolution Modeling

Multiresolution models can capture a wide range of levels of detail of an object and can be used to reconstruct any one of those levels on demand. These levels of detail of multiresolution models allow us temporarily to filter out detailed information for faster or interactive visualization or other analysis purposes. This is the case, for example, that a faster zooming process allows us to enhance the opportunity of interactivity. In addition, multiresolution models can assist in geometric processing algorithms. For example, collision detection or volume intersection computations are often iterative and require reasonably good initial approximations. An outstanding approach to multiresolution modeling is the use of wavelet methods. Muraki [3] discussed some aspects of tensor product methods in the 3D case.

E. Scattered Data Modeling

The term "scattered data" means nonuniform distributions of the data and irregularities in the data, contrary to data that lies at regular Cartesian (or rectilinear) grids. For most applications, the data of interest is not available on a nice regular grid. Rather, it is arbitrarily located in a spherical or volumetric domain. Many applications have troubles with these types of data, but have solved these problems as follows: a modeling function that fits the scattered data is first found, and then the function is sampled on the type of grid. These sampling outputs are often passed onto a rendering procedure in which a 3D array would be used as a data structure.

Further research is needed to develop accurate, efficient, and easily implemented modeling methods for very large data sets. In particular, we are interested in estimating and controlling errors. Well-known approaches for scattered data modeling are the modified quadratic Shepard (MQS) [4], volume splines [5,6], multiquadric [5,6], and volume minimum norm network (MNN) [7]. See the references for more details on these methods.

F. Feature Segmentation

The term, "segmentation" has been defined as the process of identifying which pixels or voxels of an image belong to a particular object, such as bone, fat, or tissue in medical applications. Now it includes the extraction of intrinsic data characteristics or meaningful objects from given original data, and so is often referred to as feature extraction or segmentation.

Generally, the features are application-dependent, and so the mechanisms for characterizing them must be flexible and general. Some researchers expect that a certain mathematical model will make it possible to detect the feature objects and/or represent them. Also others expect that expert system technology will do.

Scientific data visualization has acquired increased interest and popularity in many scientific domains. It mainly takes aims at devising mathematical principles, computational algorithms, and well-organized data structures, which do transform massive scientific data sets into meaningful pictures and other graphic representations that improve apprehension or inspiration. To achieve these goals, many visualization tools and techniques have appeared and been integrated to some degree within a system. Few systems, however, fully meet their users' various needs for visualization, and system developers are not allowed to construct general visualization tools without considering a data dimensionality and distribution or other intrinsic data characteristics, i.e., domain-dependent features.

The crucial factor of these difficulties arises basically from a lack of mathematical models that represent a wide range of scientific data sets and realize all standard visualization procedures such as volumetric modeling, graphical representation, and feature segmentation within a unified system framework.

In this chapter, we describe a mathematical model necessary for establishing a foundation for the evolving needs of visualization systems as mentioned above, and demonstrate its usefulness by applying it to data sets from computational fluid dynamics.

II. REPRESENTATION OF VOLUMETRIC DATA

A. Introduction to Hypervolume

To understand the internal structures or the relationships implied by scientific data sets, we first should describe their shapes, which represent a three-dimensional geometric object or a region of interest in which physical phenomena occur. Also to visualize these phenomena at a given time, we must explain the scene's appearance at that time.

For all of them, we suggest a *hypervolume,* as a mathematical formulation, which is based on a higher dimensional trivariate and dynamic NURBS (nonuniform rational B-splines) representation [8–10]. The term "higher dimensional" presents no dependence of data dimensionality, and "trivariate and dynamic" implies the capabilities of describing an evolving physical object in spatial and temporal coordinates, respectively. NURBS, as a well-known interpolating function, plays a key role in transforming a discrete data set into a continuous world.

The hypervolume consists of two different models that are independent of each other. One is the *geometry volume,* which defines 3D geometric objects, such as inhomogeneous materials or a region of interest in which physical phenomena occur, covered by the scientific data sets. The other is the *attribute volume,* which describes scalar- or vector-valued physical field variables, such as pressure, temperature, velocity, and density, as functions of four variables, i.e., the three positional coordinates and time. It also can include the graphical variables, such as color and opacity. The combination of these two volumes can provide all the geometric, physical, and graphical information for representing, manipulating, analyzing, and rendering scientific data sets. The relevant procedure will be explained later in detail.

B. Mathematical Representation of a Hypervolume

The mathematical formulation of the geometry volume of a hypervolume, which is a tensor product of NURBS of order ku in u direction, kv in v direction, kw in w direction, and kt in t direction, is a three-dimensional, quadvariate vector-valued piecewise rational function of the form

$$\mathrm{G}(u,v,w,t)=\frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw}\sum_{l=0}^{nt}h_{ijkl}\mathrm{G}_{ijkl}N_i^{ku}(u)N_j^{kv}(v)N_k^{kw}(w)N_l^{kt}(t)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw}\sum_{l=0}^{nt}h_{ijkl}N_i^{ku}(u)N_j^{kv}(v)N_k^{kw}(w)N_l^{kt}(t)}. \tag{1}$$

$\{\mathrm{G}_{ijkl}=(x_{ijkl},y_{ijkl},z_{ijkl})\}\subset\mathbf{R}^3$ forms a tridirectional control grid in the three-dimensional rectangular space, $\{h_{ijkl}\}$ are the weights, and $\{N_i^{ku}(u)\}$, $\{N_j^{kv}(v)\}$, $\{N_k^{kw}(w)\}$, and $\{N_l^{kt}(t)\}$ are the normalized B-spline basis functions defined on the knot vectors

$$\mathbf{u}=\{u_i\}_{i=0}^{nu+ku}=\{\bar{u}_0,\ldots,\bar{u}_{ku-1},\bar{u}_{ku},\ldots\ldots,\bar{u}_{nu},\bar{u}_{nu+1},\ldots,\bar{u}_{nu+ku}\}$$

where $\bar{u}_0=\cdots=\bar{u}_{ku-1}$ and $\bar{u}_{nu+1}=\cdots=\bar{u}_{nu+ku}$,

$$\mathbf{v}=\{v_j\}_{j=0}^{nv+kv}=\{\bar{v}_0,\ldots,\bar{v}_{kv-1},\bar{v}_{kv},\ldots\ldots,\bar{v}_{nv},\bar{v}_{nv+1},\ldots,\bar{v}_{nv+kv}\}$$

where $\bar{v}_0=\cdots=\bar{v}_{kv-1}$ and $\bar{v}_{nv+1}=\cdots=\bar{v}_{nv+kv}$,

$$\mathbf{w}=\{w_k\}_{k=0}^{nw+kw}=\{\bar{w}_0,\ldots,\bar{w}_{kw-1},\bar{w}_{kw},\ldots\ldots,\bar{w}_{nw},\bar{w}_{nw+1},\ldots,\bar{w}_{nw+kw}\}$$

where $\bar{w}_0=\cdots=\bar{w}_{kw-1}$ and $\bar{w}_{nw+1}=\cdots=\bar{w}_{nw+kw}$,

$$\mathbf{t}=\{t_l\}_{l=0}^{nt+kt}=\{\bar{t}_0,\ldots,\bar{t}_{kt-1},\bar{t}_{kt},\ldots\ldots,\bar{t}_{nt},\bar{t}_{nt+1},\ldots,\bar{t}_{nt+kt}\}$$

where $\bar{t}_0=\cdots=\bar{t}_{kt-1}$ and $\bar{t}_{nt+1}=\cdots=\bar{t}_{nt+kt}$.

Also note that the parameters, u, v, and w, in Eq. (1) represent three positional coordinates in the four-dimensional parameter space of the geometry volume, and the parameter t is used for the time coordinate.

To get more understandable interpretation for Eq. (1), we can rewrite it as

$$\mathbf{G}(u,v,w,t) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\left(\sum_{l=0}^{nt} h_l \mathbf{G}_{ijkl} N_l^{kt}(t)\right) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\left(\sum_{l=0}^{nt} h_l N_l^{kt}(t)\right) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}, \quad (2)$$

where $h_{ijkl} = h_{ijk}h_l$, and it follows that

$$\begin{aligned}\mathbf{G}(u,v,w,t) &= \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\left(\sum_{l=0}^{nt} h_l \mathbf{G}_{ijkl} N_l^{kt}(t)\right) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\left(\sum_{l=0}^{nt} h_l N_l^{kt}(t)\right)\left(\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)\right)} \\ &= \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\left(\frac{\sum_{l=0}^{nt} h_l \mathbf{G}_{ijkl} N_l^{kt}(t)}{\sum_{l=0}^{nt} h_l N_l^{kt}(t)}\right) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}. \quad (3)\end{aligned}$$

Therefore, we obtain

$$\mathbf{G}(u,v,w,t) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\mathbf{G}_{ijk}(t) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}, \quad (4)$$

where

$$\mathbf{G}_{ijk}(t) = \frac{\sum_{l=0}^{nt} h_l \mathbf{G}_{ijkl} N_l^{kt}(t)}{\sum_{l=0}^{nt} h_l N_l^{kt}(t)}. \quad (5)$$

From Eq. (4), we know that it has dynamic behavior since the control grid term in Eq. (4), which is Eq. (5) or simply a NURBS curve, describes a spatial movement or deformation as the parameter t (= time) elapses. Also if we set the control grid term be constant at all elapsed time, then Eq. (4) describes only the static geometric object; that is, it does not contain dynamic behavior information. The static geometry volume is often referred to as a NURBS volume, which has the form

$$\mathbf{V}(u,v,w) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\mathbf{V}_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}. \quad (6)$$

Similar to Eq. (1), the attribute volume of a hypervolume can be expressed by

$$\mathbf{A}(u,v,w,t) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw}\sum_{l=0}^{nt} h_{ijkl}\mathbf{A}_{ijkl} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w) N_l^{kt}(t)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw}\sum_{l=0}^{nt} h_{ijkl} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w) N_l^{kt}(t)} \quad (7)$$

from which we obtain

$$\mathbf{A}(u,v,w,t) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\mathbf{A}_{ijk}(t) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}, \quad (8)$$

where

$$\mathbf{A}_{ijk}(t) = \frac{\sum_{l=0}^{nt} h_l \mathbf{A}_{ijkl} N_l^{kt}(t)}{\sum_{l=0}^{nt} h_l N_l^{kt}(t)}. \tag{9}$$

The attribute volume in Eq. (8) is also a quadvariate, i.e., u, v, w spatially and t temporally, vector-valued piecewise rational function, of which the order is the same as that of the geometry volume in Eq. (1). Note that Eq. (7) is also a tensor product of the nonuniform rational B-spline of order ku, kv, kw, and kt in the u, v, w, and t direction respectively.

$\{\mathbf{A}_{ijkl}\} \subset \mathbf{R}^a$ forms a tridirectional control grid in the a-dimensional application space. For a fluid flow application, $\mathbf{A}_{ijkl} = (\rho_{ijkl}, \mathbf{V}_{ijkl}, e_{ijkl})$ is defined in this chapter where $\rho \in \mathbf{R}^1$ is a flow density, $\mathbf{V} \in \mathbf{R}^3$ is a flow velocity, and $e \in \mathbf{R}^1$ is an internal energy per unit mass.

Finally, a hypervolume can be derived simply from Eqs. (1) and (7) by the following procedures. We assume that the orders, ku, kv, kw, kt, the weight h_{ijkl}, the number of control vertices along each direction, nu, nv, nw, nt, and the knot vectors, $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$, $\mathbf{t}$ of two volumes, i.e., Eqs. (1) and (7), are identical. Also we substitute $\binom{\mathbf{G}_{ijk}(t)}{0}$ for $\mathbf{G}_{ijk}(t)$ in Eq. (4) and also $\binom{0}{\mathbf{A}_{ijk}(t)}$ for $\mathbf{A}_{ijk}(t)$ in Eq. (8), and then perform the vector sum of the two results. From this summation, we obtain our goal of a hypervolume. That is,

$$\frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\binom{\mathbf{G}_{ijk}(t)}{0} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)} \tag{10}$$

$$+ \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\binom{0}{\mathbf{A}_{ijk}(t)} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)} \tag{11}$$

$$= \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\binom{\mathbf{G}_{ijk}(t)}{\mathbf{A}_{ijk}(t)} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}. \tag{12}$$

Also it follows that

$$\mathbf{H}(u, v, w, t) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\mathbf{H}_{ijk}(t) N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w)}, \tag{13}$$

where

$$\mathbf{H}_{ijk}(t) = \begin{pmatrix} \mathbf{G}_{ijk}(t) \\ \mathbf{A}_{ijk}(t) \end{pmatrix}. \tag{14}$$

C. Main Features and Application Areas of a Hypervolume

The main features of the hypervolume proposed in this chapter are summarized as follows:

- This volume makes it possible to analyze the physical phenomena structure at any arbitrary position in a continuous physical domain when only discrete scientific data are given.

- This volume has the properties of convex hull, local control, and affine invariance under geometric transformations such as translation, rotation, parallel, and perspective projections and can utilize other useful techniques developed in the CAGD (computer-aided geometric design) literature, because this model is based on a NURBS representation.
- The geometric volume, which describes a physical domain in the parametric form, provides many differential elements such as arc length, surface area, and volume element, which are often required to calculate numerically any physical or interesting feature for a feature-based visualization.
- The attribute volume, which describes field variables in the parametric form, provides various expressions for the derivative operators such as gradient, divergence, curl, and Laplacian, in connection with the geometry volume.
- This volume allows existing visualization techniques to visualize a physical phenomena structure without changing their internal system structure, and makes it possible to implement and enhance a feature-based visualization.
- This volume permits multiple physical domains. That is, a set of hypervolumes can be dealt with from the decomposed physical domains in a systematic way.
- This volume enables us to represent/manage both the spatial and the temporal domains. That is, the independent parameters, u, v, and w, of a hypervolume govern a physical domain at an instant time t, and a parameter t is used to record the time history of this physical domain. For example, in the case of flow visualization, this volume can describe a complex physical motion in a turbulent flow domain with a time history.

The hypervolume presented in this chapter can be applied to many physical situations where interesting objects are evolving as time elapses. Note that these situations typically have been described by numerical computations or experimental measurements, and recorded into scientific data sets. Some of the applications include:

- the description of 3D geometric objects when they do not have movement or deformation and their internal physical properties or field variables do not vary as time elapses (e.g., the description of homogeneous solids standing fixed),
- the description of the spatial movement or deformation of 3D geometric objects of which internal physical properties are constant along time (e.g., the description of a rigid body motion of inhomogeneous materials),
- the description of historic records of physical field variables when 3D geometric objects in which physical phenomena occur do not vary (e.g., the description of water flowing into a fixed space), and
- the description of dynamics of 3D geometric objects with varying physical field variables as time elapses (e.g., the description of gas movement as its internal properties are varying).

Expressing these situations in terms of two constituent volumes of a hypervolume, we simply state that:

- both the geometric volume and the attribute volume are static,
- the geometric volume is static but the attribute volume is dynamic,

- the geometric volume is dynamic but the attribute volume is static, and
- both the geometric volume and the attribute volume are dynamic.

Note that "static" means no change along time while "dynamic" denotes any alteration as time elapses.

III. MANIPULATION OF VOLUMETRIC DATA

We introduce the derivatives of a hypervolume that are required as an elementary function in most numerical computations, especially in the computation of differential elements or derivative operators for advanced functionality. We also discuss two different generation algorithms of a hypervolume. Using these generation methods, we can obtain an actual instance of a hypervolume model from a given original data set without depending on the data dimensionality and distribution. Then we introduce several gridding methods that enable a hypervolume to handle scattered data. For a simple description, we use a NURBS volume instead of a hypervolume. A simple extension allows us to construct complete algorithms for a hypervolume, since a hypervolume comes from a NURBS volume.

A. Derivatives of a NURBS Volume

We begin by recalling the *Leibniz* rule, a more general version of the product rule for differentiation,

$$\frac{\partial^p}{\partial x^p}\{f_1(x)\cdot f_2(x)\} = \sum_{r=0}^{p}\binom{p}{r}\left\{\frac{\partial^r}{\partial x^r}f_1(x)\right\}\cdot\left\{\frac{\partial^{p-r}}{\partial x^{p-r}}f_2(x)\right\}, \tag{15}$$

where

$$\binom{p}{r} = \frac{p!}{r!(p-r)!}.$$

For clarity, let us introduce the differentiation notation

$$\frac{\partial^{a+b+c}}{\partial u^a \partial v^b \partial w^c}\mathbf{V}(u,v,w) = D_u^a D_v^b D_w^c \mathbf{V}(u,v,w) \tag{16}$$

and denote a NURBS volume by

$$\mathbf{V}(u,v,w) = \frac{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}\mathbf{V}_{ijk}N_i^{ku}(u)N_j^{kv}(v)N_k^{kw}(w)}{\sum_{i=0}^{nu}\sum_{j=0}^{nv}\sum_{k=0}^{nw} h_{ijk}N_i^{ku}(u)N_j^{kv}(v)N_k^{kw}(w)} = \frac{\Omega(u,v,w)}{h(u,v,w)}. \tag{17}$$

From Eqs. (16) and (17), we can obtain

$$\frac{\partial^r \Omega}{\partial w^r} = D_w^r\Omega(u,v,w) = D_w^r\{\mathbf{V}(u,v,w)\cdot h(u,v,w)\}. \tag{18}$$

Substituting the *Leibniz* rule into Eq. (18) yields

$$D_w^r\{\mathbf{V}(u,v,w)\cdot h(u,v,w)\} = \sum_{k=0}^{r}\binom{r}{k} D_w^{r-k}\mathbf{V} D_w^k h$$

$$= D_w^r\mathbf{V}\cdot h + \sum_{k=1}^{r}\binom{r}{k} D_w^{r-k}\mathbf{V} D_w^k h \qquad (19)$$

and thus

$$D_w^r\mathbf{V} = \frac{1}{h}\left[D_w^r\Omega - \sum_{k=1}^{r}\binom{r}{k} D_w^{r-k}\mathbf{V} D_w^k h\right], \qquad (20)$$

which is the rth partial derivative of a NURBS volume with respect to w. Next, the qth partial derivative of Eq. (20) with respect to v is given by

$$D_v^q D_w^r\mathbf{V} = D_v^q\left[\frac{1}{h}\left\{D_w^r\Omega - \sum_{k=1}^{r}\binom{r}{k} D_w^{r-k}\mathbf{V} D_w^k h\right\}\right]$$

$$= \frac{1}{h}\left\{D_v^q\left(D_w^r\Omega - \sum_{k=1}^{r}\binom{r}{k} D_w^{r-k}\mathbf{V} D_w^k h\right) - \sum_{j=1}^{q}\binom{q}{j} D_v^{q-j} D_w^r\mathbf{V} D_v^j h\right\}$$

$$= \frac{1}{h}\left\{\begin{array}{c} D_v^q D_w^r\Omega - \sum_{k=1}^{r}\binom{r}{k}\sum_{j=0}^{q}\binom{q}{j} D_v^{q-j} D_w^{r-k}\mathbf{V} D_v^j D_w^k h \\ - \sum_{j=1}^{q}\binom{q}{j} D_v^{q-j} D_w^r\mathbf{V} D_v^j h \end{array}\right\}. \qquad (21)$$

Finally, the pth partial derivative of Eq. (21) with respect to u is given by

$$D_u^p D_v^q D_w^r\mathbf{V} = D_u^p\left[\frac{1}{h}\left\{\begin{array}{c} D_v^q D_w^r\Omega - \sum_{k=1}^{r}\binom{r}{k}\sum_{j=0}^{q}\binom{q}{j} D_v^{q-j} D_w^{r-k}\mathbf{V} D_v^j D_w^k h \\ - \sum_{j=1}^{q}\binom{q}{j} D_v^{q-j} D_w^r\mathbf{V} D_v^j h \end{array}\right\}\right]$$

$$= \frac{1}{h}\left[D_u^p\left\{\begin{array}{c} D_v^q D_w^r\Omega - \sum_{k=1}^{r}\binom{r}{k}\sum_{j=0}^{q}\binom{q}{j} D_v^{q-j} D_w^{r-k}\mathbf{V} D_v^j D_w^k h \\ - \sum_{j=1}^{q}\binom{q}{j} D_v^{q-j} D_w^r\mathbf{V} D_v^j h \\ - \sum_{i=1}^{p}\binom{p}{i} D_u^{p-i} D_v^q D_w^r\mathbf{V} D_u^i h \end{array}\right\}\right].$$

Thus we obtain

$$D_u^p D_v^q D_w^r\mathbf{V} = \frac{1}{h}\left[\begin{array}{c} D_u^p D_v^q D_w^r\Omega \\ - \sum_{k=1}^{r}\binom{r}{k}\sum_{j=0}^{q}\binom{q}{j}\sum_{i=0}^{p}\binom{p}{i} D_u^{p-i} D_v^{q-j} D_w^{r-k}\mathbf{V} D_u^i D_v^j D_w^k h \\ - \sum_{j=1}^{q}\binom{q}{j}\sum_{i=0}^{p}\binom{p}{i} D_u^{p-i} D_v^{q-j} D_w^r\mathbf{V} D_u^i D_v^j h \\ - \sum_{i=1}^{p}\binom{p}{i} D_u^{p-i} D_v^q D_w^r\mathbf{V} D_u^i h \end{array}\right], \qquad (22)$$

which is the pth, qth, and rth partial derivatives of a NURBS volume with respect to u, v, and w, respectively. Note that we can compute $D_u^p D_v^q D_w^r \Omega(u, v, w)$ and $D_u^p D_v^q D_w^r h(u, v, w)$ in Eq. (22) by using the *deBoor* algorithm [11].

B. Generation of a NURBS Volume

In this section, we consider two simple algorithms for a NURBS volume generation. The first algorithm belongs to the generation of an interpolated volume and the second is for swept volumes, all of which are based on NURBS parametric form.

1. Interpolated Volume

We describe how to construct a NURBS volume by an interpolation method, which will be presented in detail below, with a volumetric data distributed over a regular Cartesian or curvilinear grid. If the volumetric data are not available on a grid configuration, we need to go through the gridding process and produce an array of values from original data.

The geometry volume of a hypervolume can be generated by the simple extension of the same interpolation algorithm applied to the NURBS volume, if we have time series information of the volumetric data. Similarly the attribute volume can be constructed.

For a better understanding of the interpolation algorithm, we assume all homogeneous coordinates (i.e., weight values) have unit values, and thus the rational form of a NURBS volume is reduced to the simpler nonrational form. That is, with an assumption, $h_{ijk} = 1$, and a relation, $\sum_{i=0}^{nu} N_i^{ku}(u) = \sum_{j=0}^{nv} N_j^{kv}(v) = \sum_{k=0}^{nw} N_k^{kw}(w) = 1$, the NURBS volume in Eq. (6) can be written as

$$\mathbf{V}(u, v, w) = \sum_{i=0}^{nu} \sum_{j=0}^{nv} \sum_{k=0}^{nw} \mathbf{V}_{ijk} N_i^{ku}(u) N_j^{kv}(v) N_k^{kw}(w). \tag{23}$$

Note that we need to use the rational form of Eq. (23), i.e., Eq. (6), if we have an analytical shape [12]. For example, in analyzing or visualizing a pipe flow, the pipe's geometry, which has the shape of a cylinder, cannot be represented by a nonrational form.

Now, we are given a $(nu + 1) \times (nv + 1) \times (nw + 1)$ number of grid points and our goal is to build a trivariate piecewise nonrational function given in Eq. (23) that interpolates the grid data. The nonrational B-spline volume in Eq. (23) can be written as

$$\begin{aligned} \mathbf{V}(u, v, w) &= \sum_{i=0}^{nu} \left(\sum_{j=0}^{nv} \left(\sum_{k=0}^{nw} \mathbf{V}_{ijk} N_k^{kw}(w) \right) N_j^{kv}(v) \right) N_i^{ku}(u) \\ &= \sum_{i=0}^{nu} \left(\sum_{j=0}^{nv} \mathbf{C}_{ij}(w) N_j^{kv}(v) \right) N_i^{ku}(u) \\ &= \sum_{i=0}^{nu} \mathbf{S}_i(v, w) N_i^{ku}(u). \end{aligned} \tag{24}$$

In Eq. (24), $\mathbf{C}_{ij}(w)$ and $\mathbf{S}_i(v, w)$ are referred to as a control curve and a control

surface, respectively, and are defined as

$$\mathbf{C}_{ij}(w) = \sum_{k=0}^{nw} \mathbf{V}_{ijk} N_k^{kw}(w) \tag{25}$$

$$\mathbf{S}_i(v, w) = \sum_{j=0}^{nv} \mathbf{C}_{ij}(w) N_j^{kv}(v). \tag{26}$$

Now we will determine the knot vectors of $\mathbf{V}(u, v, w)$, $\mathbf{u} = \{u_i\}_{i=0}^{nu+ku}$, $\mathbf{v} = \{v_j\}_{j=0}^{nv+kv}$, and $\mathbf{w} = \{w_k\}_{k=0}^{nw+kw}$. Consider a k-directional data array $\{\mathbf{P}_{ijk}\}_{k=0}^{k=nw}$ for each $i = 0, \ldots, nu$ and $j = 0, \ldots, nv$. First, we compute $\{w_{ijk}\}_{k=0}^{k=nw+kw}$ with each k-directional data array by using the parameterization technique introduced by Hartley and Judd [13]. Then we compute w_k by

$$w_k = \frac{\sum_{i=0}^{nu} \sum_{j=0}^{nv} w_{ijk}}{(nu+1)(nv+1)}. \tag{27}$$

Similarly, u_i and v_j can be computed as

$$u_i = \frac{\sum_{j=0}^{nv} \sum_{k=0}^{nw} u_{ijk}}{(nv+1)(nw+1)}, \qquad v_j = \frac{\sum_{k=0}^{nw} \sum_{i=0}^{nu} v_{ijk}}{(nw+1)(nu+1)}. \tag{28}$$

From the knot vectors computed, we compute the Greville abscissa [14] (ξ_i, η_j, ζ_k), corresponding to the grid data $\mathbf{P}_{ijk}$; they are given by

$$\begin{aligned} \xi_i &= \frac{u_{i+1} + u_{i+2} + \cdots + u_{i+ku}}{ku}, \quad i = 0, \ldots, nu \\ \eta_j &= \frac{v_{j+1} + v_{j+2} + \cdots + v_{j+kv}}{kv}, \quad j = 0, \ldots, nv \\ \zeta_k &= \frac{w_{k+1} + w_{k+2} + \cdots + w_{k+kw}}{kw}, \quad k = 0, \ldots, nw. \end{aligned} \tag{29}$$

Now, with the Greville abscissa $\{(\xi_i, \eta_j, \zeta_k)\}$ computed in Eq. (29) and the grid data set $\{\mathbf{P}_{ijk}\}$, we compute $\{\mathbf{V}_{ijk}\}$ such that

$$\begin{aligned} \mathbf{V}(\xi_i, \eta_j, \zeta_k) &= \sum_{i=0}^{nu} \left(\sum_{j=0}^{nv} \left(\sum_{k=0}^{nw} \mathbf{V}_{ijk} N_k^{kw}(\zeta_k) \right) N_j^{kv}(\eta_j) \right) N_i^{ku}(\xi_i) \\ &= \sum_{i=0}^{nu} \left(\sum_{j=0}^{nv} \mathbf{C}_{ij}(\zeta_k) N_j^{kv}(\eta_j) \right) N_i^{ku}(\xi_i) \end{aligned} \tag{30}$$

$$= \sum_{i=0}^{nu} \mathbf{S}_i(\eta_j, \zeta_k) N_i^{ku}(\xi_i) = \mathbf{P}_{ijk}, \tag{31}$$

where

$$\mathbf{C}_{ij}(\zeta_k) = \sum_{k=0}^{nw} \mathbf{V}_{ijk} N_k^{kw}(\zeta_k) \tag{32}$$

$$\mathbf{S}_i(\eta_j, \zeta_k) = \sum_{j=0}^{nv} \mathbf{C}_{ij}(\zeta_k) N_j^{kv}(\eta_j). \tag{33}$$

First, we calculate unknown $\mathbf{S}_i(\eta_j, \zeta_k)$ from known $\mathbf{P}_{ijk}$ in Eq. (31). Then in Eq. (33), unknown $\mathbf{C}_{ij}(\zeta_k)$ is computed from $\mathbf{S}_i(\eta_j, \zeta_k)$. Finally, we find a goal $\mathbf{V}_{ijk}$ from the computed $\mathbf{C}_{ij}(\zeta_k)$ in Eq. (32).

For $j = 0, \ldots, nv$ and $k = 0, \ldots, nw$, the system of $(nv + 1) \times (nw + 1)$ equations in Eq. (31) is given by

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_0, \zeta_0) N_i^{ku}(\xi_i) = \mathbf{P}_{i00}$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_0, \zeta_1) N_i^{ku}(\xi_i) = \mathbf{P}_{i01}$$

$$\bullet \bullet \bullet$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_0, \zeta_{nw}) N_i^{ku}(\xi_i) = \mathbf{P}_{i0nw}$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_1, \zeta_0) N_i^{ku}(\xi_i) = \mathbf{P}_{i10}$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_1, \zeta_1) N_i^{ku}(\xi_i) = \mathbf{P}_{i11}$$

$$\bullet \bullet \bullet$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_1, \zeta_{nw}) N_i^{ku}(\xi_i) = \mathbf{P}_{i1nw}$$

$$\bullet \bullet \bullet \bullet$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_{nv}, \zeta_0) N_i^{ku}(\xi_i) = \mathbf{P}_{inv0}$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_{nv}, \zeta_1) N_i^{ku}(\xi_i) = \mathbf{P}_{inv1}$$

$$\bullet \bullet \bullet$$

$$\sum_{i=0}^{nu} \mathbf{S}_i(\eta_{nv}, \zeta_{nw}) N_i^{ku}(\xi_i) = \mathbf{P}_{invnw}.$$

From the system of equations above, we can see that $\mathbf{S}_i(\eta_j, \zeta_k)$ is a control net of a control curve that interpolates the $\{\mathbf{P}_{ijk}\}$ along the u direction. Similarly, for $k = 0, \ldots, nw$ and $i = 0, \ldots, nu$, the system of $(nw + 1) \times (nu + 1)$ equations in Eq. (33) is given by

$$\sum_{j=0}^{nv} \mathbf{C}_{0j}(\zeta_0) N_j^{kv}(\eta_j) = \mathbf{S}_0(\eta_j, \zeta_0)$$

$$\sum_{j=0}^{nv} \mathbf{C}_{1j}(\zeta_0) N_j^{kv}(\eta_j) = \mathbf{S}_1(\eta_j, \zeta_0)$$

$$\bullet \bullet \bullet$$

$$\sum_{j=0}^{nv} C_{nuj}(\zeta_0) N_j^{kv}(\eta_j) = S_{nu}(\eta_j, \zeta_0)$$

$$\sum_{j=0}^{nv} C_{0j}(\zeta_1) N_j^{kv}(\eta_j) = S_0(\eta_j, \zeta_1)$$

$$\sum_{j=0}^{nv} C_{1j}(\zeta_1) N_j^{kv}(\eta_j) = S_1(\eta_j, \zeta_1)$$

$$\bullet\bullet\bullet$$

$$\sum_{j=0}^{nv} C_{nuj}(\zeta_1) N_j^{kv}(\eta_j) = S_{nu}(\eta_j, \zeta_1)$$

$$\bullet\bullet\bullet\bullet$$

$$\sum_{j=0}^{nv} C_{0j}(\zeta_{nw}) N_j^{kv}(\eta_j) = S_0(\eta_j, \zeta_{nw})$$

$$\sum_{j=0}^{nv} C_{1j}(\zeta_{nw}) N_j^{kv}(\eta_j) = S_1(\eta_j, \zeta_{nw})$$

$$\bullet\bullet\bullet$$

$$\sum_{j=0}^{nv} C_{nuj}(\zeta_{nw}) N_j^{kv}(\eta_j) = S_{nu}(\eta_j, \zeta_{nw}).$$

From the system of equations above, we can see that $C_{ij}(\zeta_k)$ is a control net of a control curve that interpolates the $S_i(\eta_j, \zeta_k)$ along the v direction. Again, the system of $(nu+1) \times (nv+1)$ equations in Eq. (32) for $i = 0, \ldots, nu$ and $j = 0, \ldots, nv$ is given by

$$\sum_{k=0}^{nw} V_{00k} N_k^{kw}(\zeta_k) = C_{00}(\zeta_k)$$

$$\sum_{k=0}^{nw} V_{01k} N_k^{kw}(\zeta_k) = C_{01}(\zeta_k)$$

$$\bullet\bullet\bullet$$

$$\sum_{k=0}^{nw} V_{0nvk} N_k^{kw}(\zeta_k) = C_{0nv}(\zeta_k)$$

$$\sum_{k=0}^{nw} V_{10k} N_k^{kw}(\zeta_k) = C_{10}(\zeta_k)$$

$$\sum_{k=0}^{nw} V_{11k} N_k^{kw}(\zeta_k) = C_{11}(\zeta_k)$$

$$\bullet\bullet\bullet$$

$$\sum_{k=0}^{nw} \mathbf{V}_{1nvk} N_k^{kw}(\zeta_k) = \mathbf{C}_{1nv}(\zeta_k)$$

$$\bullet\bullet\bullet\bullet$$

$$\sum_{k=0}^{nw} \mathbf{V}_{nu0k} N_k^{kw}(\zeta_k) = \mathbf{C}_{nu0}(\zeta_k)$$

$$\sum_{k=0}^{nw} \mathbf{V}_{nu1k} N_k^{kw}(\zeta_k) = \mathbf{C}_{nu1}(\zeta_k)$$

$$\bullet\bullet\bullet$$

$$\sum_{k=0}^{nw} \mathbf{V}_{nunvk} N_k^{kw}(\zeta_k) = \mathbf{C}_{nunv}(\zeta_k).$$

From the system of equations above, we can see that $\mathbf{V}_{ijk}$ is a control net of a control curve that interpolates the $\mathbf{C}_{ij}(\zeta_k)$ along the w direction. Therefore, the $\mathbf{V}_{ijk}$ is a control net or grid of a control volume that interpolates the $\{\mathbf{P}_{ijk}\}$ along each parametric direction.

2. Swept Volume

Now we address the topic of sweeping a section surface along an arbitrary guide (= trajectory) curve. Denote the guide curve by $\mathbf{G}(w)$ and the section surface by $\mathbf{S}(u, v)$ as shown in Fig. 1. The procedure for generating a swept volume from $\mathbf{G}(w)$ and $\mathbf{S}(u, v)$ shown in Fig. 1 is as follows:

(a) Input the guide curve $\mathbf{G}(w)$, one point $\mathbf{P}$ located on both $\mathbf{G}(w)$ and $\mathbf{S}(u, v)$, and the section surface $\mathbf{S}(u, v)$.

(b) Arrange $\mathbf{G}(w)$ and $\mathbf{S}(u, v)$ to have the same configuration as shown in Fig. 1. That is, the normal vector of $\mathbf{S}(u, v)$ is directed along the tangent vector of $\mathbf{G}(w)$. The normal vector of $\mathbf{S}(u, v)$ is calculated by the cross product of the u and v directional tangent vectors of $\mathbf{S}(u, v)$.

(c) Compute the Greville abscissa, which is often referred to as node values, of $\mathbf{G}(w)$ and also those of $\mathbf{S}(u, v)$ in both directions, u and v.

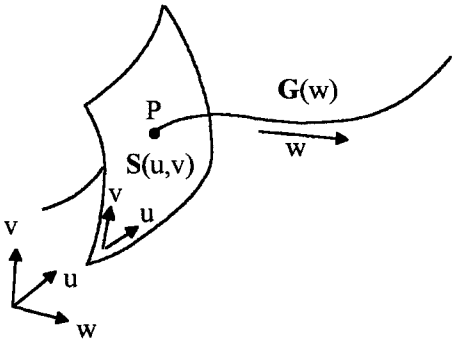


FIGURE 1 A guide curve and a section surface for sweeping.

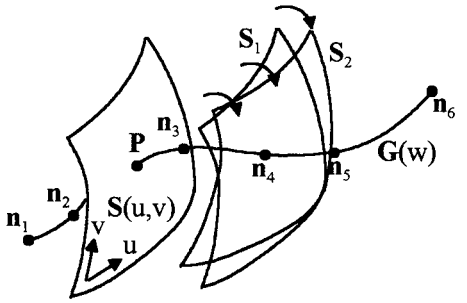


FIGURE 2 Transformations of a section surface in sweeping.

(d) Compute the Greville points (= node points) of $G(w)$ evaluated at the Greville abscissa and also the tangent vectors at the same positions. For a better understanding, see Fig. 2, which illustrates six node points, $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \mathbf{n}_4, \mathbf{n}_5, \mathbf{n}_6$ of $G(w)$ when $G(w)$ has six control points. In addition, compute the tangent vector of $G(w)$ at the point, $\mathbf{P}$.

(e) Transform the section surface, $S(u, v)$, by translation and rotation through two steps as follows: The first step is to translate $S(u, v)$ by $(\mathbf{n}_1 - \mathbf{P})$. The second step is to rotate $S(u, v)$ translated in the first step. Here, the rotational axis is determined from the cross product of two tangent vectors, one being the tangent vector of $G(w)$ at position $\mathbf{P}$ and the other being that at position $\mathbf{n}_1$. The rotational angle is measured from these two tangent vectors, and the center of rotation is $\mathbf{n}_1$. In addition, we evaluate the node points of the transformed $S(u, v)$ at the Greville abscissa of $S(u, v)$ already calculated in step (c), and save the results together with position $\mathbf{n}_1$.

(f) Transform $S(u, v)$ resulting from step (e) by a process similar to step (e). That is, move the $S(u, v)$ by $(\mathbf{n}_2 - \mathbf{n}_1)$ and then rotate $S(u, v)$ by the angle between two tangent vectors, one being the tangent vector of $G(w)$ at position $\mathbf{n}_1$ and the other being that at position $\mathbf{n}_2$. Here the rotational axis is calculated from the cross product of these tangent vectors, and the rotation center is $\mathbf{n}_2$. In addition, evaluate and save the node points of the transformed $S(u, v)$ at the Greville abscissa already calculated in step (c). The same procedure is repeated until the node points of the transformed $S(u, v)$ are computed at the last node point of $G(w)$. Figure 2 illustrates S_1, the section surface after translating from $\mathbf{n}_3$ to $\mathbf{n}_4$, and S_2, the section surface after rotation by the angle between two tangent vectors of $G(w)$ at positions $\mathbf{n}_3$ and $\mathbf{n}_4$. Through the processes described above, we can get the node points distribution in the u and v directions along the w directional node position.

(g) Now we can get the control grid (or regular control point set) for generating a swept volume by using the interpolated method described in the previous section, where the Greville abscissa (or node values) from step (c) and the node points from steps (e) and (f) are used as input data. The knot vectors along the u and v directions of a swept volume are the same as those of the section surface $S(u, v)$ and the w-directional knot vector of a swept volume is the same as that of the guide curve $G(w)$.

Note that the procedure for constructing extruded volumes and revolved ones can be easily derived from the swept volume algorithm described above.

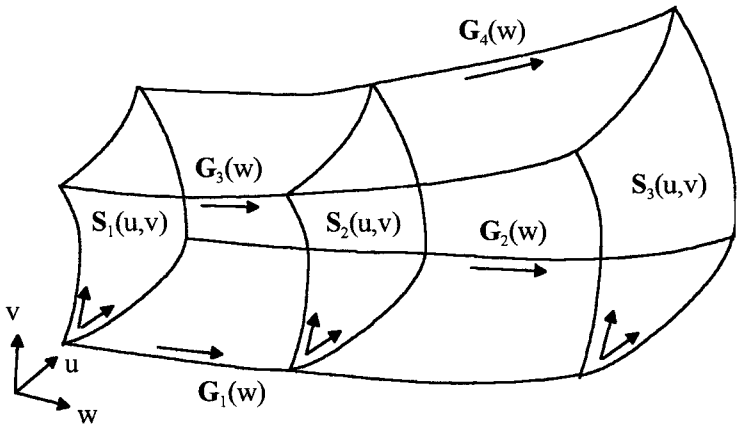


FIGURE 3 Four guide curves and three section surfaces for a swept volume.

The extruded volume can be viewed as a special case of swept volume when the guide curve $G(w)$ is a straight line. Similarly the revolved volume can be generated if $G(w)$ is a circular arc or a circle.

Furthermore, we will present another algorithm for constructing a swept volume in a situation different from that described above. Its configuration is shown in Fig. 3. That is, four guide curves, $G_1(w)$, $G_2(w)$, $G_3(w)$, and $G_4(w)$, and n section surfaces, $S_1(u, v)$, $S_2(u, v)$,..., $S_n(u, v)$, are used to define the swept volume. Note that only three section surfaces are shown in Fig. 3. The detail procedure is described as follows:

(a) Input four guide curves $G_1(w)$, $G_2(w)$, $G_3(w)$, and $G_4(w)$ and n section surfaces, $S_1(u, v)$, $S_2(u, v)$,..., $S_n(u, v)$. See Fig. 3 for $n = 3$.

(b) Arrange the four guide curves to have the same direction as that shown in Fig. 3. All section surfaces are also adjusted to have the same configuration, and their normal vectors are directed toward the same direction as the four guide curves as shown in Fig. 3.

(c) Reparameterize the four guide curves such that the parameter for each curve ranges from 0.0 to 1.0. Similarly, reparameterize all the section surfaces such that each surface has a parameter range 0.0 to 1.0 along the u and v directions.

(d) Raise the degree of the guide curves of lower degree such that the four guide curves have the same degree, and insert proper knots such that each curve has the same knot vector.

(e) Raise the degree of section surfaces of lower degree such that all the section surfaces have the same degree along the u and v directions, and insert proper knots such that each surface has the same knot vectors along u and v directions.

(f) For the ith section surface, take the following procedures: Find t_1, t_2, t_3, t_4 such that $G_1(t_1) = S_i(0,0)$, $G_2(t_2) = S_i(1,0)$, $G_3(t_3) = S_i(0,1)$, and $G_4(t_4) = S_i(1,1)$. If t_1, t_2, t_3, t_4 are not same, then for the jth guide curve ($j = 1,2,3,4$), split $G_j(w)$ at $w = t_j$, resulting in two splitted curves, $G_j^0(w = 0 : t_j)$ and $G_j^1(w = t_j : 1)$, and then reparameterize $G_j^0(w = 0 : t_j)$ and $G_j^1(w = t_j : 1)$ such that they have the parameter range $[0, (t_1 + t_2 + t_3 + t_4)/4]$ and $[(t_1 + t_2 + t_3 + t_4)/4, 1]$, respectively.

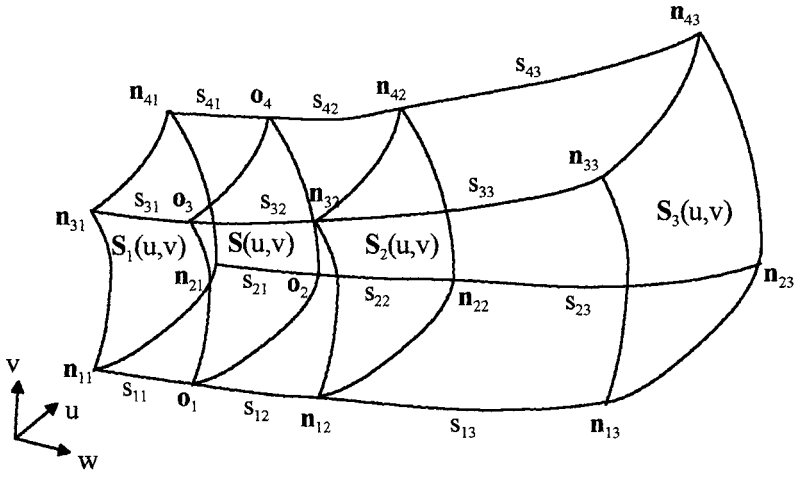


FIGURE 4 A configuration for transforming section surfaces.

Note that each of the four corner points of the ith section surface lies on the corresponding guide curve at the same parameter value, $(t_1 + t_2 + t_3 + t_4)/4$ $(= t_m)$,which means that $S_i(0,0) = G_1(t_m)$, $S_i(1,0) = G_2(t_m)$, $S_i(0,1) = G_3(t_m)$, and $S_i(1,1) = G_4(t_m)$. Thus t_m is the common Greville abscissa (= node value) of each guide curve, and the four corner points are the node points of the corresponding guide curves.

(g) Transform all the section surfaces to form a single merged surface located at the new node points of the four guide curves at which the section surfaces are not originally located. Figure 4 shows the configuration for this transformation.

Let us denote the new node points by $\mathbf{o}_1$, $\mathbf{o}_2$, $\mathbf{o}_3$, $\mathbf{o}_4$, where $\mathbf{o}_i$ is the node point of $G_i(w)(i = 1,2,3,4)$, which can be computed in step (f). Also we denote by $\mathbf{n}_{11}$, $\mathbf{n}_{12}$, $\mathbf{n}_{13}$ the node points of $G_1(w)$ at which the section surfaces have originally been located. Similarly, the node points of $G_2(w)$, $G_3(w)$, and $G_4(w)$ are denoted by $\mathbf{n}_{21}$, $\mathbf{n}_{22}$, $\mathbf{n}_{23}$, $\mathbf{n}_{31}$, $\mathbf{n}_{32}$, $\mathbf{n}_{33}$, and $\mathbf{n}_{41}$, $\mathbf{n}_{42}$, $\mathbf{n}_{43}$, respectively. Now, we compute s_{11},s_{12},s_{13}, which are arc lengths between $\mathbf{o}_1$ and $\mathbf{n}_{11}$, $\mathbf{n}_{12}$, $\mathbf{n}_{13}$ along $G_1(w)$. Similarly, we compute s_{21},s_{22},s_{23}, s_{31},s_{32},s_{33}, and s_{41},s_{42},s_{43} along $G_2(w)$, $G_3(w)$, and $G_4(w)$, respectively.

The transformation consists of two steps. One is the transformation of each section surface. That is, in Fig. 4, $S_1(u, v)$, $S_2(u, v)$, and $S_3(u, v)$ are transformed into $S_1^*(u, v)$, $S_2^*(u, v)$, and $S_3^*(u, v)$ such that the four corner points of each transformed surface are located at $\mathbf{o}_1$, $\mathbf{o}_2$, $\mathbf{o}_3$, and $\mathbf{o}_4$. The details of the transformation will be illustrated later in Fig. 5. The other step is the weighted sum of $S_i^*(u, v)$. First, we will explain the weighted sum. In Fig. 4, we compute the weighting factor s_i of each section surface $S_i(u, v)$ by

$$s_1 = \frac{1}{\frac{s_{11} + s_{21} + s_{31} + s_{41}}{4}}, \qquad s_2 = \frac{1}{\frac{s_{12} + s_{22} + s_{32} + s_{42}}{4}},$$

$$s_3 = \frac{1}{\frac{s_{13} + s_{23} + s_{33} + s_{43}}{4}}$$

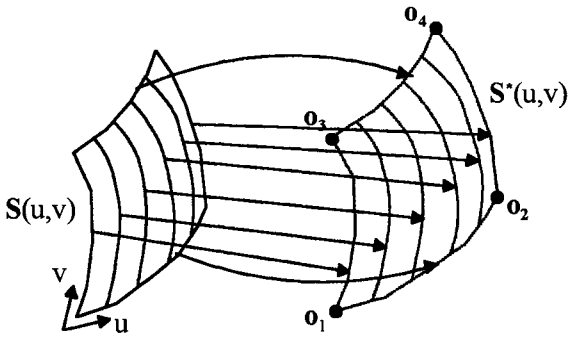


FIGURE 5 Transformation of u- and v-ray curves obtained from a section surface.

from which we know that the weighting factors are determined from the inverse of the average distance between the new node points o_1, o_2, o_3, o_4 and each section surface $S_i(u, v)$. Thus, by using the weighting factors, we compute the single merged surface $S(u, v)$ by

$$S(u, v) = \frac{s_1 S_1^* + s_2 S_2^* + s_3 S_3^*}{s_1 + s_2 + s_3}.$$

Now, we will explain the transformation from $S_i(u, v)$ to $S_i^*(u, v)$ with Fig. 5. For the ith section surface, take the following procedures:

(1) Compute the u-directional node values of $S_i(u, v)$ and then compute the v-ray curves, isoparametric curves along the v direction at each of the u-directional node values. Here, we know that the number of v-ray curves is equal to that of the u-directional node points.

(2) Compute the u-ray curves and iso parametric curves along the u direction, at the staring and ending v parameters of $S_i(u, v)$. Again we know that the number of the u-ray curves is two.

(3) Move the two u-ray curves to the positions o_1, o_2 and o_3, o_4. That is, move both ending points of the lower u-ray to the positions o_1 and o_2, and the upper u-ray to o_3 and o_4, while preserving the tangent vectors of each of the u-ray curves at its end points during the movement. Then evaluate the two u-ray curves at the u-directional node values already calculated in step (1).

(4) Move each of the v-ray curves into two corresponding node points, one being calculated from the upper u-ray and the other from the lower in step (3), while preserving the tangent vectors of the v-ray curve at its end points during the movement. This movement process is applied to all v-ray curves positioned at the u-directional node points.

(5) Compute $S_i^*(u, v)$ by interpolating the node points of the v-ray curves obtained in step (4).

(h) By using the interpolated volume method, we obtain the control grid of a swept volume from the initial section surfaces and their transformed surfaces obtained in step (g). Note that the u- and v-knot vectors of the swept volume are equal to those of anyone of the section surfaces, and the w-knot vector is equal to that of anyone of the guide curves.

C. Gridding Methods of Scattered Data

Many researchers have tried to construct methods for creating structured or unstructed grids from scattered data points. Some methods involve unstructured grids or meshes being dependent on the distribution of the scattered data points. In two dimensions, this would be a method for triangulating a set of scattered points in a plane [15]. The rendering process of the triangular meshes determines the color of each pixel in the image by applying bilinear interpolation to the data value at each node. Although the original data are preserved, the sparseness of the data points results in a pseudo-colored distribution that is difficult to interpret. That is, it is possible to incorrectly visualize the spatial distribution of the data.

Other methods attempt to create new data distributions from the original data for their actual visualization. Most of the methods construct a structured grid configuration, which has a regular Cartesian or curvilinear grid type, calculated from the original scattered data. This section serves only as a very brief introduction to the topic.

The simplest approach, called the "nearest-neighbor gridding" method, is to create a Cartesian grid from the ungridded point data by the following procedure. First, we calculate the smallest box that minimally covers the spatial distribution of the data. Second, a Cartesian (or regular) grid configuration is constructed from the box and stored into n-dimensional arrays. The n-dimensional arrays consist of common 3D spatial grids and 1D model fields, e.g., temperature, pressure, and velocity. These array types of data structure can be found in the Vis-5D system [16], which uses 5D rectangles organized as 2D arrays of 3D spatial grids. The 2D arrays are indexed by time and by model field. Next, we find the nearest point to each grid point in the resultant grid configuration. Finally, we assign that grid point the original point's value, as shown in Fig. 6. Figure 6 illustrates the case where only three grid points are related. Such a technique has a positive side that it preserves the original data values, while the rendering results might not satisfy qualitative display requirements because the discrete spatial structure is preserved.

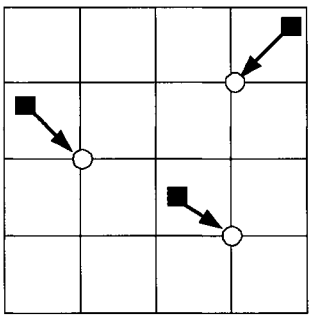


FIGURE 6 Nearest neighbor gridding method.

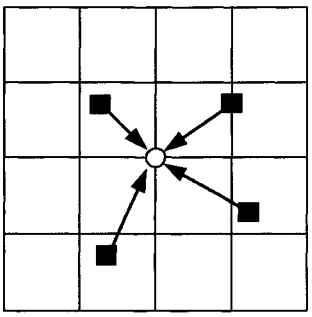


FIGURE 7 Weighted average gridding method.

A potentially more appropriate method, and certainly more accurate than the simple method aforementioned, uses the concept of weighted averaging as shown in Fig. 7. The governing procedures are as follows: First, we compute the smallest box enveloping the original data spatially, and create a Cartesian grid configuration, saved into n-dimensional arrays. These steps are similar to those explained above. Next, for each grid point in the grid configuration, we calculate the weighted average of the scalar values of the original m data points spatially nearest to that grid point. Here, we should obtain a weighting factor applied to each of the m values, that is, $w_i = f(d_i)$, where $d_i (i = 1, \ldots, m)$ is the distance between the grid point and the ith point in the original data points. Figure 7 illustrates a case for $m = 4$. Depending on the characteristics of application fields, we can choose one of various weighting functions. The weighting function $w = d^{-2}$ is commonly recommended in most applications. Note that this approach is based on Shepard's method [17], and many advanced techniques [10] such as radial basis function methods, FEM methods, and multistage methods are still under development.

We need only a scatter data interpolation function for modeling or analyzing the spatial distribution of scattered data, while for effective visualization of the data, it is necessary to perform the meshing or gridding process and produce a 3D array of values or more dimensional arrays from original data. It is worthwhile noting that this gridding process is necessary, not sufficient. The reason comes from that the array data structure does not give directly new values at arbitrary positions in a continuous field and thus new meaningful objects cannot be extracted. For example, streamlines or stream surfaces in a steady fluid flow cannot be extracted. In order to achieve a goal of qualitative visualization or new feature segmentation, we need a more continuous process or a new interpolating function that efficiently computes continuous data fields from the discrete grid distribution. We expect that the hypervolume suggested in this chapter enables us to realize the continuous process significantly. That is, the interpolated volume method in the previous section will provide a way of obtaining the continuous field from the discrete grid points, and from the interpolated volume, the computation for the extraction of any feature objects, and their rendering implementation will be realized in an efficient manner.

Some practical examples of feature segmentation will be illustrated in the next section.

IV. RENDERING METHODS OF VOLUMETRIC DATA

We discuss several volumetric rendering methods for visualizing a three- or more dimensional data set to enhance the structural comprehension of the data set significantly. One of the most commonly used rendering methods is the direct volume rendering algorithm, e.g., ray casting [18] or splatting [19]. Also the tiny cube [20,21], hexahedron [22], and vanishing cube [20,21] methods are well known as visualization techniques of multivariable function objects.

The tiny cube method creates a number of small cubes (or boxes) to be placed in the spatial domain of the data set. These boxes are commonly distributed in a gridded structure, as shown in Fig. 8, with a uniform gap between boxes in each direction, i.e., the u, v, and w directions in the gridded distribution. The faces of these boxes are then rendered by color information calculated from the function values of the volume model used for representing the volumetric data. Here we can use $\mathbf{H}(u, v, w)$ or hypervolume in Eq. (13) as the volume model. It should be noted that the images displayed by this method become difficult to interpret if the number of boxes in each direction is greater than 15.

The hexahedron method is similar to the tiny cube method above. In this method, we assume that the gap distance between boxes is zero, and use the function values at the volume centers of some of the boxes in order to produce the images of the boxes. In this case, a box placed in the interior can be visible if the boxes between the interior box and the viewer are not turned on.

In the vanishing cube method, we create equally spaced planes in each direction instead of the boxes. That is, we create three rectangles for each grid point in the governing volume model, as shown in Fig. 9. Each of the rectangles is colored using bilinear interpolation of the values of the volume model at

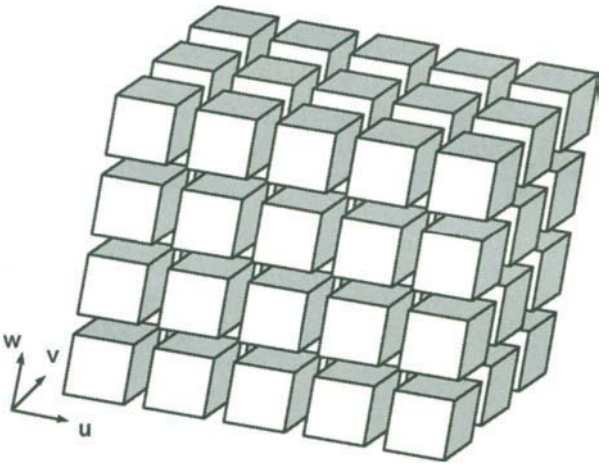


FIGURE 8 Tiny cube method.

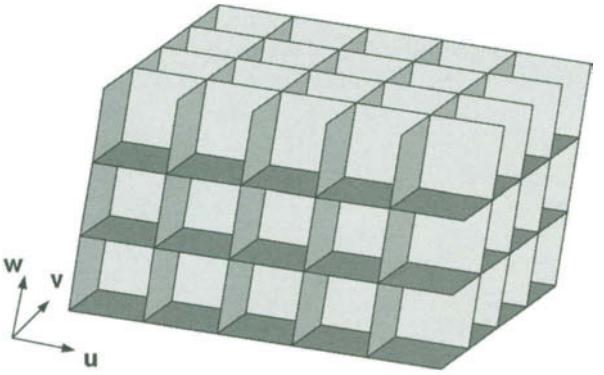


FIGURE 9 Vanishing cube method.

its four vertices. By using this method, the interior planes can be viewed if the exterior planes are rendered as partially transparent. In real practice, the number of planes embedded in the volume model cannot be too large.

Direct volume rendering involves "shooting" rays through an object. To simplify the required summation or integration process [23], the object is usually divided into a large number of uniformly sized boxes called *voxels*. Then for each voxel, the voxel intensity and its gradient magnitude should be calculated in order to classify and render the data set. The classification requires calculating an optical density value for each voxel in the data set, called opacity. The opacity is often used to put a stress on any meaningful surfaces or boundaries between different materials. Typically, the opacities are calculated from either voxel intensities or a combination of voxel intensities and gradient information. Once the data set is classified and the opacity and color are computed, we obtain two new voxel data sets, a color and an opacity data set. Then we simply determine the color and opacity on sampling points (generally not on voxel positions) by commonly trilinear interpolation, but this step may lower the quality of the image. As an alternative, we may directly compute the color and opacity on the sampling positions without using interpolation during the rendering process. However, this step may increase the complexity of the algorithm and require more computational processes. Finally, the rendering images are constructed by using several rendering techniques. The major distinction between the techniques is the order in which the voxels are processed to generate an image, and they are grouped into image–order and object–order algorithms. Examples of image-order algorithms are ray casting and Sabella's method [24]. The splatting algorithm [19], V-buffer algorithm [25], and slice shearing algorithm [26] belong to object–order algorithms. Note that these methods require a voxelization algorithm or an octree structure [18] as a preprocessing step, and have a disadvantage that they are computationally intensive in any case.

As a generalization of the direct volume method, a method based on linear transport theory has been proposed [23]. In this method, the rays are replaced by virtual (light) particles moving through the data field. These particles interact with the 3D field data (which is represented by grid points) and collect information about the light intensity for screen display. The general transport theory model for volume rendering is given by the integro-differential equation. The evaluation of this model requires very expensive computations, and thus

techniques such as Monte Carlo simulation, series expansions, and rendering algorithms on massively parallel machines [27] are usually applied with this model. It should be mentioned that the data value and its derivatives at the evaluation points are currently computed by trilinear interpolation, which sometimes may produce incorrect information and should be improved to reduce computational inaccuracy.

Finally, we remark that most of the rendering algorithms have mainly focused on the construction procedures that produce graphic images, not a volume representation. We expect that the volume representation can reduce the amount of computation for rendering images. In fact, too much computation is required for rendering images in the direct volume rendering methods. The volume representation also provides a great deal of information for the improvement of the rendering quality. We suggest that a hypervolume as the volume representation will play a great role in the enhancement of the rendering algorithms. Main features of the hypervolume were listed in detail in the previous section.

V. APPLICATION TO FLOW VISUALIZATION

A. Related Works

Visualizing 3D fluid flow fields has long been a part of fluid dynamics research. Since the 19th century, an experimental flow visualization for showing the patterns resulting from the injection of ink, smoke, or particles in a fluid has been performed. Current computer-aided visualization techniques also basically simulate this experimental visualization scheme.

We can categorize these flow visualization techniques according to the data dimensionality we wish to visualize—scalar, vector, and tensor—or by the object dimensionality to be visualized—points, curves, surfaces, and volumes—or by the scope of the domain of interest—local and global. Reviews of prior works in the classification can be found in [28,29]. In general, common visualization techniques such as arrow plots, streamlines [30], and particle traces work well for 2D applications, while advanced visualization techniques such as stream surfaces [31], tensor probes [32], vector field topology [33,34], and hyperstreamlines [35] are useful for visualizing the velocity field of 3D data sets. In some cases, cutting planes can be effective for viewing slices of 3D data. For example, the flow around an airplane wing may be well displayed using the cross sections of the wing.

These visualization techniques produce colorful and sometimes even beautiful images. However, there are still problems in the visualization of the vector and the tensor fields. The main reason is the inexistence of a flexible volumetric representation that feeds a variety of flow visualization techniques introduced above and makes it possible to analyze the initial flow data more intuitively and productively for the creation of effective 3D graphical presentations.

In most cases, measured or simulated volume data are typically represented as raw data with no prior definition of interesting objects to be visualized. Thus, it is difficult to provide CFD (computational fluid dynamics) researchers with a feature-based visualization, which is increasingly required in order to reduce

graphical complexity, increase information contents, and detect complex flow structures intuitively. Feature segmentation, which means the extraction of important characteristics from a data set, is needed for further analysis and interpretation and also is required as a preliminary step for an effective feature-based flow visualization. Vector field topology techniques [33,34] and mathematical morphology techniques [36] are well known as examples of feature segmentation, and iconic representations [28,29,37] are often used for displaying feature data already extracted from a data set. In most cases, this feature segmentation successfully describes the characteristics of a fluid flow field and provides elementary information for further analysis in a continuous computational space referred to as a curvilinear coordinate system. However, they have only a discrete data set as an input resource. That is, they have a large problem in that positional coordinate values and field variables' values at arbitrary positions in a continuous physical domain should be derived from the discrete raw data in the domain.

In the visualization program PLOT3D [38], as in many such packages, the values of field variables within each cell are interpolated by a trilinear function at the computational space coordinates of each query point. Here, very expensive computation is required to convert each query point in physical space into its computational space representation. For this step, the neighboring grid points of the query point should be first searched. The computational loads on the searching problem greatly limit the speed of the global visualization. Note that this basic step is frequently executed as an elementary routine for any visualization. In the basic step introduced above, which can be found in most existing visualization techniques, we can find two problems. One is the computational inaccuracy induced by the trilinear interpolation using a product of the fractional displacements of the query point along each computational space dimension within a grid cell as the relative contribution of each vertex of the grid cell. The other is the computational inefficiency caused by searching the neighbors of the query point directly from a raw data at every time step as already explained. There exists tradeoffs between the two problems. If a higher-order interpolating function is used for the improvement of the accuracy, it requires more additional neighbors of a query point and thus more expensive computational loads.

Now, we will describe the practical examples of a hypervolume as a new volumetric data representation for accurately analyzing 3D vector fields in scientific visualization and supporting all information necessary for the creation of effective 3D graphical presentations. This representation allows a unified mathematical basis to be used as a general framework in the visualization systems. This representation is also independent of any visualization technique being used, and thus is immediately applicable to many current systems. The usefulness of the data representation proposed in this paper will be demonstrated with simple examples of feature segmentation.

B. Application Examples

We have applied a hypervolume to several flow visualization examples. We considered the visualization of the streamlines or stream surfaces for various flow fields. First, a hypervolume is constructed by using the interpolation technique

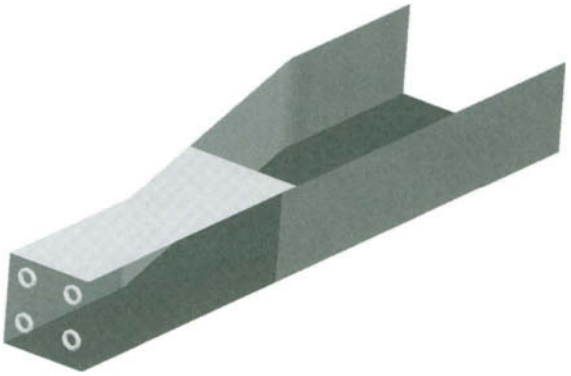


FIGURE 10 A rectangular duct geometry. Reprinted from S. Park and K. Lee, (1997) with permission from Elsevier Science.

developed in this chapter. Then the streamlines or stream surfaces are computed by a procedure to be explained later under Feature Segmentation (see next section). In addition, the pressure as an additional field variable is concurrently derived from the generated hypervolume using the laws of physics. Finally, the computed results are displayed with various different sizes and colors, which map the computed numerical values to the visual images on a computer screen.

Our first example is an internal flow in a rectangular duct as shown in Fig. 10, where several planes are removed to show the duct geometry in detail. Figure 11 shows the four streamtubes in the duct. Note that the starting seed curves of the streamtubes are four circles of the same radius, and the displayed color and radius of the streamtubes indicate the magnitude of flow velocity and the flow pressure, respectively.

Our second example illustrates the capabilities of multiple hypervolumes applied to decomposed flow domains as shown in Fig. 12. Figure 13 shows a

FIGURE 11 Four streamtubes in a rectangular duct where color and radius indicate the flow velocity and pressure, respectively. Reprinted from S. Park and K. Lee, (1997) with permission from Elsevier Science.

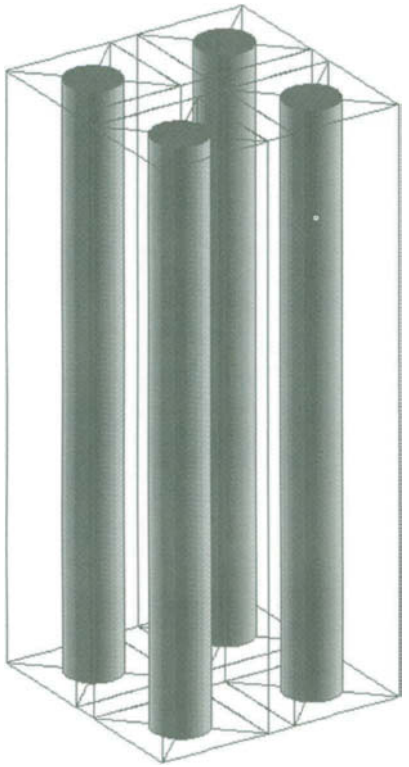


FIGURE 12 Decomposed flow domains in a tube bank. Reprinted from S. Park and K. Lee, (1997) with permission from Elsevier Science.

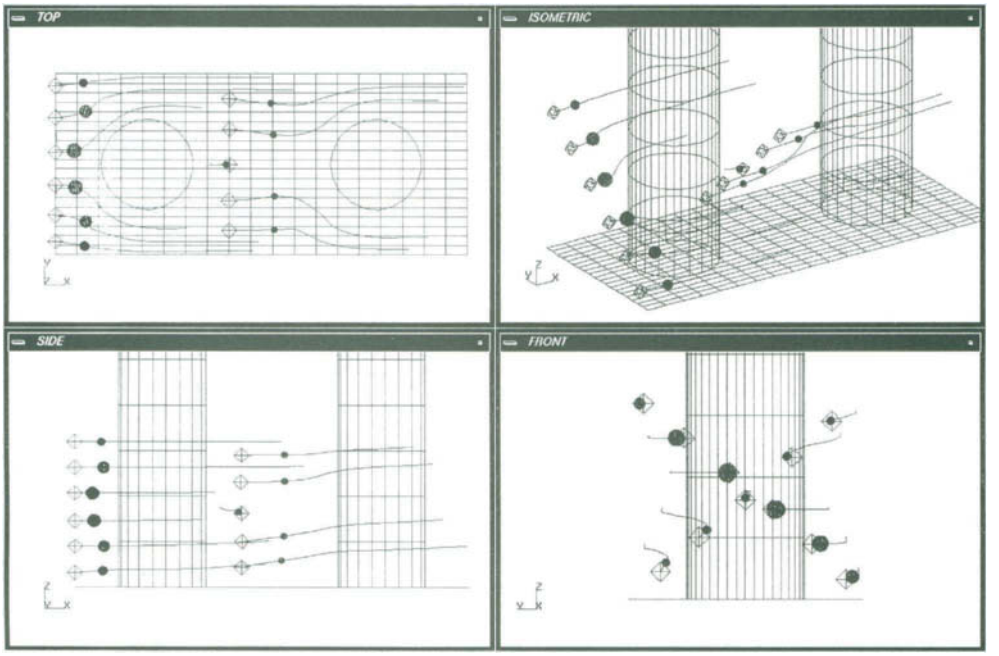


FIGURE 13 Streamlines in a tube bank where the diameter of the particles indicates the pressure. Reprinted from S. Park and K. Lee, (1997) with permission from Elsevier Science.

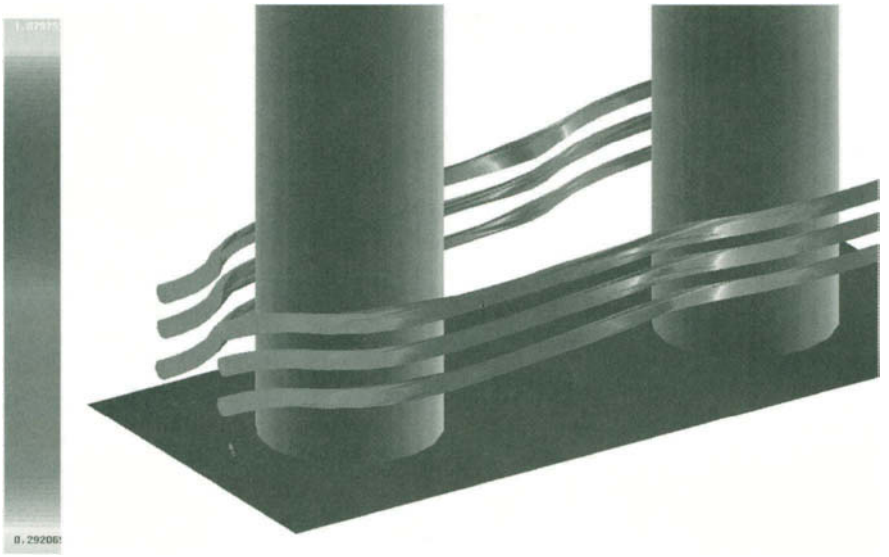


FIGURE 14 Six streamtubes in a tube bank where color and radius indicate the flow velocity and pressure, respectively. Reprinted from S. Park and K. Lee, (1997) with permission from Elsevier Science.

frame at an instant when particles, having left their start points, are located at specific positions along their trajectories. In the figure, the diameter of the particles indicates the instantaneous pressure along the path. Figure 14 shows the six streamtubes in the tube bank in Fig. 12.

Our third example presents the stream surfaces around an airplane wing as shown in Fig. 15. The starting curve is a line, and the line becomes curved along its trajectory in the figure. Note that the color indicates the relative magnitude of a flow velocity.

C. Feature Segmentation

In this section, we will show how a streamline is generated from the hypervolume as an example of feature segmentation. This example will show the capabilities of our hypervolume for a feature-based visualization. As is well known, a streamline in a steady fluid velocity field is a trajectory of a single small particle injected into the flow. The streamline, being integral curves, gives insight into the global structure of the velocity vector field and serves as a basic element for other flow visualizations; e.g., a stream surface can be seen as a set of streamlines. The streamline in a steady state can be computed by

$$\mathbf{x} = \mathbf{x}_0 + \int_0^t \mathbf{v}\, dt, \qquad \mathbf{v} = \mathbf{v}_0 + \int_0^t \mathbf{a}\, dt \tag{34}$$

using a velocity vector field $\mathbf{v}(t)$ and an initial seed point $\mathbf{x}_0(t_0)$

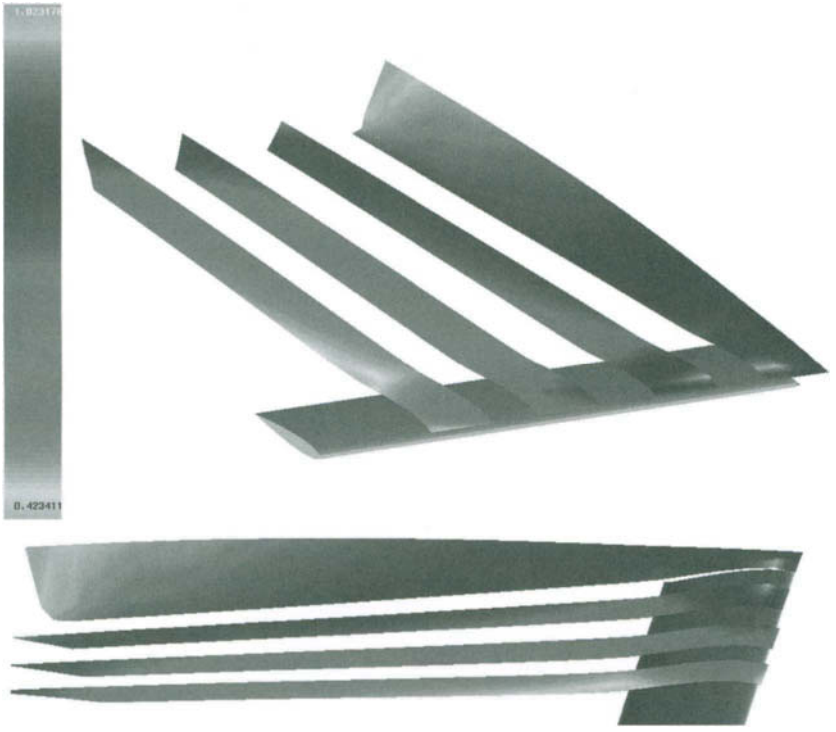


FIGURE 15 Four stream surfaces around an airplane wing where color indicates the flow velocity. Reprinted from S. Park and K. Lee, (1997) with permission from Elsevier Science.

If we assume that the acceleration **a** in Eq. (34) is constant during Δt, then the streamline equation in Eq. (34) can be rewritten as

$$\Delta \mathbf{x} = \mathbf{v}\Delta t + \frac{1}{2}\mathbf{a}(\Delta t)^2. \tag{35}$$

Thus, we can get the streamline if we can compute the term **a** in Eq. (35). The acceleration **a** can be expressed by the velocity **v**, the partial derivatives $\partial \mathbf{v}/\partial u$, $\partial \mathbf{v}/\partial v$, $\partial \mathbf{v}/\partial w$, and the contravariant basic vectors ∇u, ∇v, ∇w as follows.

Since

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{v}}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial \mathbf{v}}{\partial v}\frac{\partial v}{\partial t} + \frac{\partial \mathbf{v}}{\partial w}\frac{\partial w}{\partial t}$$

where

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial u}{\partial z}\frac{\partial z}{\partial t} = \nabla u \cdot \mathbf{v},$$

$$\frac{\partial v}{\partial t} = \frac{\partial v}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial v}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial v}{\partial z}\frac{\partial z}{\partial t} = \nabla v \cdot \mathbf{v},$$

and

$$\frac{\partial w}{\partial t} = \frac{\partial w}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial w}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial w}{\partial z}\frac{\partial z}{\partial t} = \nabla w \cdot \mathbf{v}$$

the acceleration **a** is given by

$$\mathbf{a} = (\nabla u \cdot \mathbf{v})\frac{\partial \mathbf{v}}{\partial u} + (\nabla v \cdot \mathbf{v})\frac{\partial \mathbf{v}}{\partial v} + (\nabla w \cdot \mathbf{v})\frac{\partial \mathbf{v}}{\partial w}. \tag{36}$$

Here we can easily get ∇u, ∇v, ∇w from the geometry volume and $\mathbf{v}$, $\partial \mathbf{v}/\partial u$, $\partial \mathbf{v}/\partial v$, $\partial \mathbf{v}/\partial w$ from the attribute volume suggested in this chapter. Note that the parameters, u, v, and w represent three positional coordinates in the parameter space of a hypervolume.

Therefore, we can compute $\Delta \mathbf{x}$ during a time interval Δt from Eqs. (35) and (36). Note that if the time interval Δt decreases, the assumption that **a** is constant during the Δt is reasonably valid and so we can obtain a more realistic streamline during Δt. Finally, the streamline over a full domain of a fluid flow can be easily obtained as follows:

(1) Find (u_0, v_0, w_0) such that $\mathbf{G}(u_0, v_0, w_0) = \mathbf{x}_0$, where $\mathbf{x}_0$ is an initial seed point and $\mathbf{G}(u, v, w)$ is the geometry volume suggested in this chapter. Note that the Newton–Raphson method for solving this nonlinear equation is applied to form

$$\begin{bmatrix} \dfrac{\partial \mathbf{F}(u_i, v_j, w_k)}{\partial u} & \dfrac{\partial \mathbf{F}(u_i, v_j, w_k)}{\partial v} & \dfrac{\partial \mathbf{F}(u_i, v_j, w_k)}{\partial w} \end{bmatrix} \begin{bmatrix} u_{i+1} - u_i \\ v_{j+1} - v_j \\ w_{k+1} - w_k \end{bmatrix} = [-\mathbf{F}(u_i, v_j, w_k)],$$

where $\mathbf{F}(u, v, w) = \mathbf{G}(u, v, w) - \mathbf{x}_0 = 0$.

The Greville abscissa of the control point distanced nearly from the given $\mathbf{x}_0$ is chosen as the initial guess. The convergence is achieved when the differences of $\Delta u = u_{i+1} - u_i$, $\Delta v = v_{j+1} - v_j$, and $\Delta w = w_{k+1} - w_k$ are under the user-supplied parametric tolerance and the value of $\|\mathbf{F}(u_i, v_j, w_k)\|$ is also under the user-supplied spatial tolerance.

(2) Compute **v** at (u_0, v_0, w_0) directly from $\mathbf{A}(u, v, w)$, where $\mathbf{A}(u, v, w)$ is the attribute volume suggested in this chapter.

(3) Compute **a** at (u_0, v_0, w_0) by using Eq. (36) in which each term can be calculated from $\mathbf{G}(u, v, w)$ and $\mathbf{A}(u, v, w)$.

(4) Compute $\Delta \mathbf{x}$ from **v**, **a**, and a given Δt by using Eq. (35), and then obtain $\mathbf{x} = \mathbf{x}_0 + \Delta \mathbf{x}$.

(5) Assign $\mathbf{x}_0$ to **x** and repeat step (1).

VI. SUMMARY AND CONCLUSIONS

Most existing visualization systems provide a variety of functionalities concerned with the following major subjects: data models, rendering algorithms, and graphical presentation. A data model involves describing a discrete grid distribution or an unstructured mesh with a native form or data structure, depending on the type, dimensions, and organization of the data. From this data model, a rendering algorithm extracts interesting objects and/or computes their color and opacity values by interpolation. Then a graphical presentation displays these rendering data on a typical computer screen with some graphic techniques aimed at producing realistic images. Most visualization systems follow these ordered procedures where it is worth noting that only a data model allows a wide range of visualization possibilities.

Despite many needs for a data model to build a flexible visualization system and support various application developments, a very powerful data model has not yet been established. The data model should describe adequately the full range of data used, allow a simple evaluation for rendering purposes, and derive all the required data, e.g., higher-order derivative data, from the given data set without additional computations. These requirements or expectations will be sufficiently satisfied if the hypervolume proposed in this chapter is chosen.

We have suggested the hypervolume model as a mathematical representation for modeling and rendering volumetric data, and have described two methods for creating a volume model, depending on the spatial distribution of volumetric data. The interpolated volume method can be applied to the gridded distribution of volumetric data, and the swept volume method can be utilized for volumetric modeling of the cross-sectional distribution of data sets. We have also introduced several gridding methods for scattered volumetric data, from which we can obtain discrete grid points to be directly used in the interpolated volume method. In addition, we have mentioned several rendering methods of volumetric data, all of which can be implemented with a hypervolume.

We are currently verifying that the hypervolume can provide a mathematical framework for the analysis of integrated, multidisciplinary simulations that involve interactions between disciplines such as fluid dynamics, structural dynamics, and solid mechanics. The hypervolume model presented in this chapter has a contribution in these senses.

REFERENCES

1. McCormick, B. H., DeFanti, T. A., and Brown, M. D. (Eds.). "Visualization in Scientific Computing." *Computer Graphics* **21**(6): Nov. 1987.
2. Kaufman, A., Cohen, D., and Yagel, R. Volume graphics.*Computer* **26**(7):51–64, 1993.
3. Muraki, S. Volume data and wavelet transform. *IEEE Comput. Graphics Appl.* **13**(4):50–56, 1993.
4. Franke, R., and Nielson, G. Smooth interpolation of large sets of scattered data. *Internat. J. Numer. Methods Engrg.* **15**:1691–1704, 1980.
5. Franke, R., and Nielson, G. Scattered data interpolation and applications: A tutorial and survey. In *Geometric Modeling: Methods and Their Application* (H. Hagen and D. Roller, Eds.), pp. 131–160. Springer-Verlag, Berlin, 1990.
6. Nielson, G. *et al.* Visualization and modeling of scattered multivariate data. *IEEE Comput. Graphics Appl.* **11**(3):47–55, 1991.
7. Nielson, G. A method for interpolating scattered data based upon a minimum norm network. *Math. Comp.* **40**:253–271, 1983.
8. Casale M. S., and Stanton, E. L. An overview of analytic solid modeling. *IEEE Comput. Graphics Appl.* **5**:45–56, 1985.
9. Lasser, D. Bernstein–Bezier representation of volumes. *Comput. Aided Geom. Design* **2**:145–149, 1985.
10. Lasser, D., and Hoschek, J. *Fundamentals of Computer Aided Geometric Design.* A K Peters, Wellesley, MA, 1993.
11. de Boor, C. *A Practical Guide to Splines.* Springer-Verlag, New York, 1978.
12. Tiller, W. Rational B-splines for curve and surface representation. *IEEE Comput. Graphics Appl.* **3**:61–69, 1983.
13. Hartley, P. J., and Judd, C. J. Parametrization and shape of B-spline curves for CAD. *Comput. Aided Design* **12**:235–238, 1980.
14. Farin, G. *Curves and Surfaces for Computer Aided Geometric Design,* pp. 150–151. Academic Press, San Diego, 1990.

15. Agishtein, M. E., and Migda, A. A. Smooth surface reconstruction from scattered data points. *Comput. Graphics (UK)* **15**(1):29–39, 1991.
16. Hibbard, W., and Santek, D. The VIS-5D system for easy interactive visualization. In *Proceedings of Visualization 90* (A. Kaufman, Ed.), pp. 28–35. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990.
17. Shepard, D. A two dimensional interpolation function for irregular spaced data. In *Proceedings 23rd ACM National Conference,* pp. 517–524, 1968.
18. Levoy, M. Efficient ray tracing of volume data. *ACM Trans. Graphics* **9**:245–261, 1990.
19. Westover, L. Footprint evaluation for volume rendering. *Comput. Graphics* **24**(4):367–376, 1990.
20. Nielson, G. M., and Hamann, B. Techniques for the visualization of volumetric data. In *Visualization '90* (A. Kaufman, Ed.), pp. 45–50. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990.
21. Nielson, G. M., Foley, Th. A., and Hamann, B., Lane, D. Visualizing and modeling scattered multivariate data. *IEEE Comput. Graphics Appl.* **11**:47–55, 1991.
22. Pajon, J. L., and Tran, V. B. Visualization of scalar data defined on a structured grid. In *Visualization '90* (A. Kaufman, Ed.), pp. 281–287. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990.
23. Krüger, W. The application of transport theory to visualization of 3D scalar data fields. *Comput. Phys.* (July/Aug): 397–406, 1991.
24. Sabella, P. A rendering algorithm for visualizing 3D scalar fields. *ACM Comput. Graphics* **22**:51–58, 1988.
25. Upson, C., and Keeler, M. V-BUFFER: Visible volume rendering. *ACM Comput. Graphics* **22**:59–64, 1988.
26. Lacroute, P., and Levoy, M. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Comput. Graphics* **28**(4):451–458, 1994.
27. Krüger, W., and Schroder, P. Data parallel volume rendering algorithms for interactive visualization. *Visual Comput.* **9**:405–416, 1993.
28. Hesselink, L., and Delmarcelle, T. Visualization of vector and tensor data sets. In *Frontiers in Scientific Visualization* (L. Rosenblum *et al.*, Eds.). Academic Press, New York, 1994.
29. Delmarcelle, T., and Hesselink, L. A unified framework for flow visualization. In *Engineering Visualization* (R. Gallagher, Ed.). CRC Press, Boca Raton, FL, 1994.
30. Kenwright, D., and Mallinson, G. A streamline tracking algorithm using dual stream functions. In *Proceedings of Visualization '92,* pp. 62–68. IEEE Comput. Soc. Press, Los Alamitos, CA, 1992.
31. Hultquist, J. P. M. Constructing stream surfaces in steady 3D vector fields. In *Proceedings of Visualization '92,* pp. 171–178. IEEE Comput. Soc. Press, Los Alamitos, CA, 1992.
32. de Leeuw, W. C., and van Wijk, J. J. A probe for local flow field visualization. In *Proceedings of Visualization '93,* pp. 39–45. IEEE Comput. Soc. Press, Los Alamitos, CA, 1993.
33. Helman, J., and Hesselink, L. Visualizing vector field topology in fluid flows. *IEEE Comput. Graphics Appl.* **11**:36–46, 1991.
34. Globus, A., Levit, C., and Lasinski, T. A tool for visualizing the topology of 3D vector fields. In *Proceedings of Visualization '91,* pp. 33–40. IEEE Comput. Soc. Press, Los Alamitos, CA, 1991.
35. Delmarcelle, T., and Hesselink, L. Visualizaing second-order tensor fields with hyperstreamlines. *IEEE Comput. Graphics Appl.* **13**:25–33, 1993.
36. Silver, D., and Zabusky, N. J. Quantifying visualizations for reduced modeling in nonlinear science: Extracting structures from data sets. *J. Visual Commun. Image Represent.* **4**:46–61, 1993.
37. Kerlick, G. D. Moving iconic objects in scientific visualization. In *Proceedings of Visualization '90* (A. Kaufman, Ed.), pp. 124–130. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990.
38. Buning, P. G., and Steger, J. L. Graphics and flow visualization in CFD. In *AIAA 7th CFD Conference,* Cincinnati, OH, AIAA Paper 85-1507-CP, pp. 162–170, 1985.
39. Park, S., and Lee, K. High-dimensional trivariate NURBS representation for analyzing and visualizing fluid flow data. *Computers and Graphics* **21**(4):473–482, 1997.

15

THE DEVELOPMENT OF DATABASE SYSTEMS FOR THE CONSTRUCTION OF VIRTUAL ENVIRONMENTS WITH FORCE FEEDBACK

HIROO IWATA

Institute of Engineering Mechanics and Systems, University of Tsukuba, Tsukuba 305-8573, Japan

Database and Data Communication Network Systems, Vol. 2

I. INTRODUCTION

It is well known that sense of touch is inevitable for understanding the real world. Force sensation plays an important role in the manipulation of virtual objects. A haptic interface is a feedback device that generates skin and muscle sensation, including sense of touch, weight, and rigidity. We have been working in research on haptic interfaces in virtual environments for a number of years. We have developed various force feedback devices and their applications. In most cases of haptic interfaces, software of a virtual environment is tightly connected to the control program of force displays. This problem is a hazard for development of further applications of haptic virtual environments. In 1991, we started a project for the development of a modular software tool for a haptic interface. The system was called VECS at the time. We have been improving the software tools to support various force displays and their applications [3,4,6]. Our latest system is composed of seven modules: the device driver of force display, haptic renderer, model manager, primitive manager, autonomy engine, visual display manager, and communication interface. The system is called LHX, named for library for haptics. Various types of force displays can be plugged into LHX. This chapter introduces techniques and applications in the development of a database system for haptic environments using LHX.

A. Haptic Interface

A haptic interface, or force display, is a mechanical device that generates reaction forces from virtual objects. Research activities into haptic interfaces have been rapidly growing recently, although the technology is still in a state of trial-and-error. There are three approaches to implement an haptic interface: tool handling-type force display, exoskeleton-type force display, and object-oriented-type force display. We have developed prototypes in each category.

1. Tool Handling-Type Force Display

Tool handling-type force display is the easiest way to realize force feedback. The configuration of this type is similar to a joystick. Virtual world technology usually employs glove-like tactile input devices. Users feel anxious when they put on one of these devices. If the glove is equipped with a force feedback device, the problem is more severe. This disadvantage obstructs practical use of the haptic interface. Tool handling-type force display is free from being fitted to the user's hand. Even though it cannot generate force between the fingers, it has practical advantages.

We developed a 6-DOF (degree-of-freedom) force display which has a ball grip [5]. The device is called "HapticMaster" and is commercialized by Nissho Electronics Co. (Fig. 1). The HapticMaster is a high-performance force feedback device for desktop use. This device employs a parallel mechanism in which a top triangular platform and a base triangular platform are connected by three sets of pantographs. The top end of the pantograph is connected with a vertex

FIGURE 1 HapticMaster.

of the top platform by a spherical joint. This compact hardware has the ability of carrying a large payload. Each pantograph has three DC motors.

The grip of HapticMaster can be replaced to fit specialized applications. A simulator of laparoscopic surgery, for example, has been developed by attaching a real tool for surgery on the top plate of the HapticMaster.

2. Exoskeleton-Type Force Display

In the field of robotics research, master manipulators are used in teleoperation. Most master manipulators, however, have large hardware with high costs, which restricts their application areas. In 1989, we developed a compact master manipulator as a desktop force display [2]. The core element of the device is a 6-DOF parallel manipulator, in which three sets of pantograph link mechanisms are employed. Three actuators are set coaxially with the first joint of the thumb, forefinger, and middle finger of the operator.

We improved the device by increasing the degrees-of-freedom for each finger. We have developed a new haptic interface that allows 6-DOF motion for three independent fingers: thumb, index finger and middle finger [12]. The device has three sets of 3-DOF pantographs, at the top of which three 3-DOF gimbals are connected. A thimble is mounted at the end of each gimbal. The thimble is carefully designed to fit the finger tip and is easy to put on or take

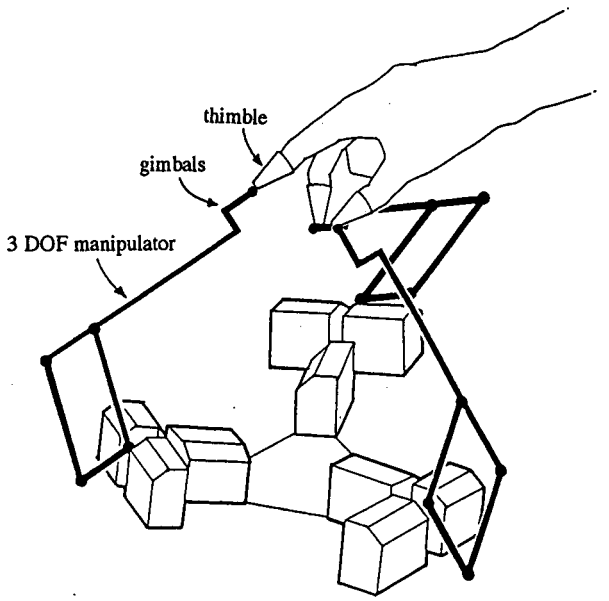


FIGURE 2 Force display for three fingers.

off. The device applies 3-DOF force at each finger tip. The user can grasp and manipulate virtual objects by their three fingers. Figure 2 illustrates the mechanical configuration of the device. This device is easily realized by replacing the top platform of the HapticMaster with three gimbals with thimbles.

3. Object-Oriented-Type Force Display

Object-oriented-type force display is a radical idea of design of a haptic interface. The device moves and deforms to present shapes of virtual objects. A user of the device can make contact with a virtual object by its surface. It allows natural interaction as compared to that of exoskeleton and tool handling type. However, it is fairly difficult to implement. Furthermore, its ability to simulate virtual objects is limited. Because of these characteristics, object-oriented type is effective for specific applications. We focused on 3D shape modeling as an application of our object-oriented-type force display. We have developed a prototype named Haptic Screen [13]. The device employs an elastic surface made of rubber. An array of actuators is set under the elastic surface. The surface deforms by the actuators. Each actuator has force sensors. Hardness of the surface is made variable by these actuators and sensors. Deformation of a virtual object occurs according to the force applied by the user. Figure 3 illustrates the mechanical configuration of the Haptic Screen.

B. Issues in Haptic Software

Software tool for development of a virtual environment is one of the key technologies in the field of virtual reality. There are commercially available software tools such as WorldToolKit, dVS, and Super Scape VRT [7–9]. Those softwares are not designed to support a haptic interface.

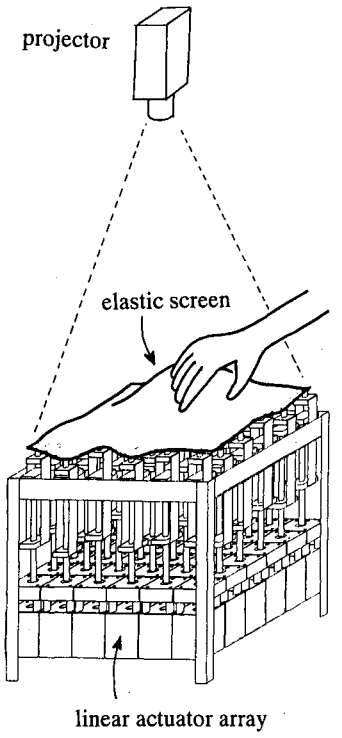


FIGURE 3 Haptic Screen.

Recently research on haptic interfaces has been growing rapidly, although software for a haptic interface is at an early state. Sensable Device developed GHOST as a software tool for their commercial haptic interface named PHANToM [11]. Another example of haptic software is found in the work by the University of North Carolina at Chapel Hill. They developed Armlib, which supports the PHANToM and ARGONE remote manipulator [10].

Issues in haptic software can be summarized in the following aspects, and LHX is designed to deal with these issues:

(1) *Wide applicability of the mechanical configuration of a haptic interface.* The PHANToM and ARGONE remote manipulator have similar mechanisms. The design of a haptic interface has wide valuation as to the mechanical configuration as discussed later. One of the focuses of LHX is how to support various configurations of a haptic interface.

(2) *Construction of a haptic user interface.* Haptic software should support easy construction of a user environment. LHX has primitives for construction of a user interface with force feedback.

(3) *Network capability.* Collaboration in virtual space is one of the major applications of virtual reality. LHX has network capability for constructing a haptic virtual environment shared by multiple users.

(4) *Connection with visual display.* In most applications, a haptic interface is connected to a visual display. LHX includes a visual display interface.

II. LHX

A. Basic Structure of LHX

In order to deal with the issues in haptic software discussed in the previous section, LHX is composed of seven modules: a device driver of force display, a haptic renderer, a model manager, a primitive manager, an autonomy engine, a visual display manager, and a communication interface. By dividing these modules, force displays and virtual environments are easily reconfigured. Figure 4 shows the basic structure of LHX.

Functions of those modules are as follows:

1. Device Driver

A device driver manages sensor input and actuator output for a haptic interface. Various types of haptic interface can be connected to LHX by changing the device driver. We developed a device driver of above-mentioned force displays so that they can be connected to LHX.

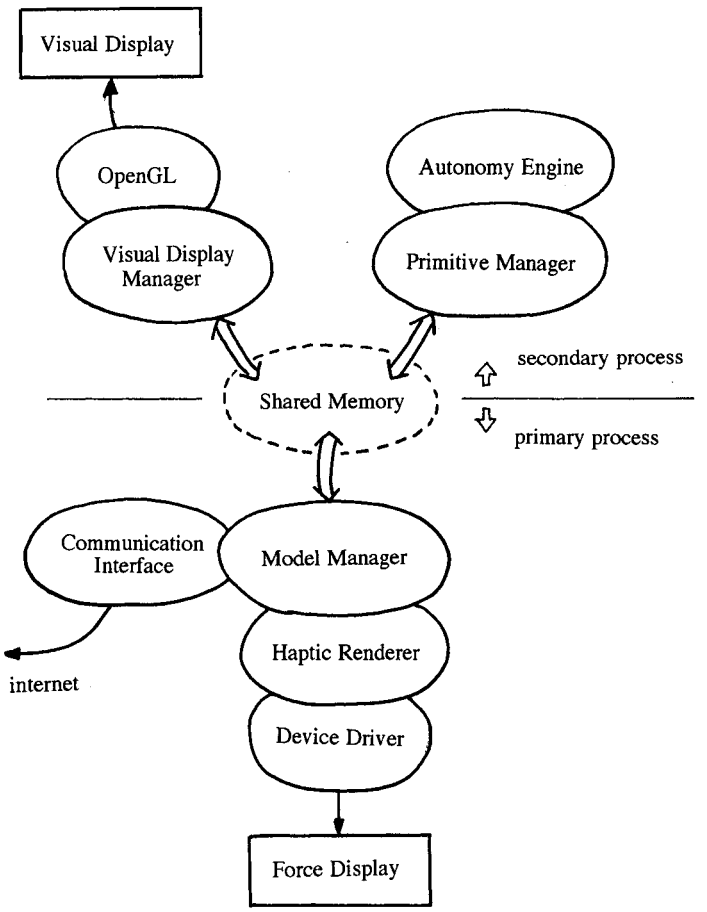


FIGURE 4 Basic structure of LHX.

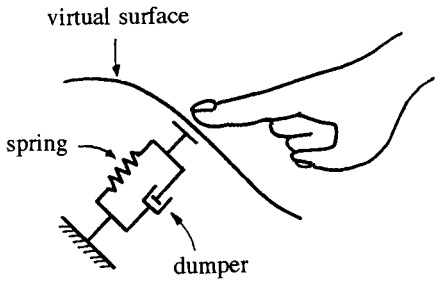


FIGURE 5 Surface haptic renderer.

2. Haptic Renderer

Currently rendering means generation of a visual image. However, force sensation also needs rendering. Hardness, weight, and viscosity of virtual objects are generated by haptic rendering. We have developed a software package for haptic rendering. Haptic render of LHX has three categories according to three types of force displays:

a. Tool Handling-Type Renderer

LHX supports two haptic renders: surface renderer and volume renderer. Surface renderer is implemented by a spring and dumper model (Fig. 5). Volume renderer is implemented by mapping voxel data to force and torque (Fig. 6).

b. Exoskeleton-Type Renderer

Exoskeleton-Type force display applies force to multiple fingers. A renderer of this category is composed of multiple surface renderers of tool handling type.

c. Object-Oriented-Type Renderer

An object-oriented-type renderer determines the stiffness of the surface of Haptic Screen according to the physical model of the virtual object.

3. Model Manager

A model of virtual objects is implemented in the model manager module of LHX. The shapes and attributes of virtual objects are defined in this module.

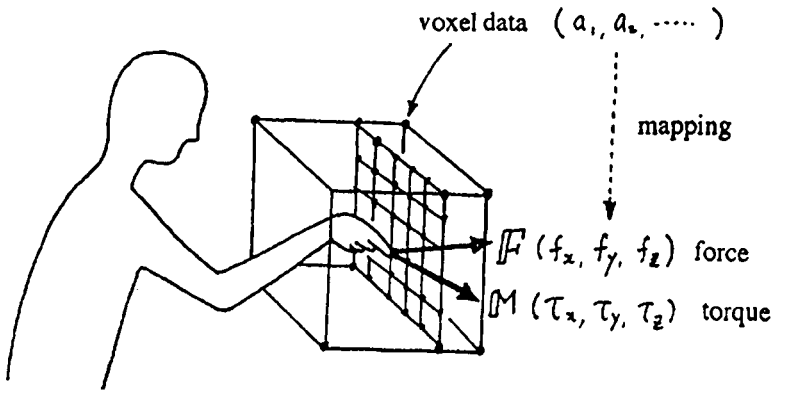


FIGURE 6 Volume haptic renderer.

Users of the LHX program determine the methods for interaction between the virtual objects and the operators.

4. Primitive Manager

Primitives of virtual objects are stored in the primitive manager. Primitives include cube, sphere, cylinder, free-form surface, and 3D voxels. Haptic icons for the user interface are also included. This module supervises the ID code of each primitive. Users of LHX interactively generate or erase primitives. Working primitives are placed in shared memory.

5. Autonomy Engine

An autonomy engine determines the behavior of virtual objects. Physical laws of the virtual world are contained in this module. Gravity, elasticity, and viscosity are currently implemented. Collisions between primitives are detected in real time. This module defines "time" of a virtual environment. The time of a virtual environment increases independently from the user. This function enables autonomous growth of virtual objects.

6. Communication Interface

LHX has a network interface by which multiple force displays are connected to each other. Multiple users can simultaneously interact in the same virtual environment. This function enables easy construction of a groupware program. LHX supports TCP/IP so that the system can use the existing Internet.

7. Visual Display Manager

The visual display manager generates graphic images of a virtual environment. This module translates the haptic model into OpenGL format. HMD, stereo shutter glasses, and a spherical screen are supported as visual displays.

B. Implementation of LHX

LHX is currently implemented in SGI and Windows NT workstations. Considering the connection of a haptic interface to a visual image generator, SGI and Windows NT workstations are the most promising platforms. C++ is used for its implementation. Since our force displays are interfaced with PCs, the device driver module is implemented in a PC. The host workstation and PC are connected by RS-232C.

LHX is composed of two processes: visual feedback and force feedback. The visual feedback process runs the visual display manager, primitive manager, and autonomy engine. The force feedback process runs the other modules. Shared memory is used for communication between these processes. The required update rate of force feedback is much higher than that of visual feedback. Images can be seen continuously at an update rate of 10 Hz. On the other hand, force feedback requires 40 Hz at least. In LHX the force feedback process has a higher priority than the visual feedback process. LHX enables a high update rate of force display in a complex virtual environment.

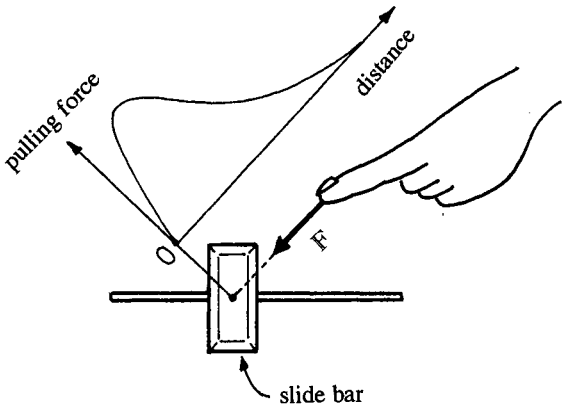


FIGURE 7 Haptic icon.

C. Haptic User Interface

Unlike a conventional 2D user interface, operation in a 3D virtual environment is complicated. It is usually difficult to point a button, which is floating in 3D space. We therefor have developed 3D icons with force feedback. When the user's hand comes close to a haptic icon, it is pulled toward the center of the icon. Figure 7 indicates the pulling force around a haptic icon. The user does not need to point the button precisely due to this pulling force, so that the performance of the pointing task is improved.

Primitives of LHX include haptic icons for user interface. Figure 8 shows a typical user interface of LHX. The user of the system sees his/her virtual hand, virtual objects, and virtual control panels. Virtual control panels include buttons or slide bars. In Fig. 8, four buttons are located on the left-hand side and a slide bar is located on the right. A virtual object seen at the center is a free-form surface. Command input and parameter settings are done through these devices. If the virtual hand comes close to these buttons, the hand is pulled toward the center of the buttons by force display. This applied force assists the user to operate the control panels. If the user grasps the button, the color of the button changes and the command is inputted.

III. APPLICATIONS OF LHX: DATA HAPTIZATION

A. Basic Idea of Haptization

Scientific visualization is a major application area of virtual reality. The recent evolution of computer graphics technology enables real-time interactive presentation of scientific data. Simulations and scientific experiments often produce data in the form of a large number of values within a three-dimensional coordinate space. This information is often hard to comprehend in numerical form. Volume visualization is a powerful tool for investigators of those data.

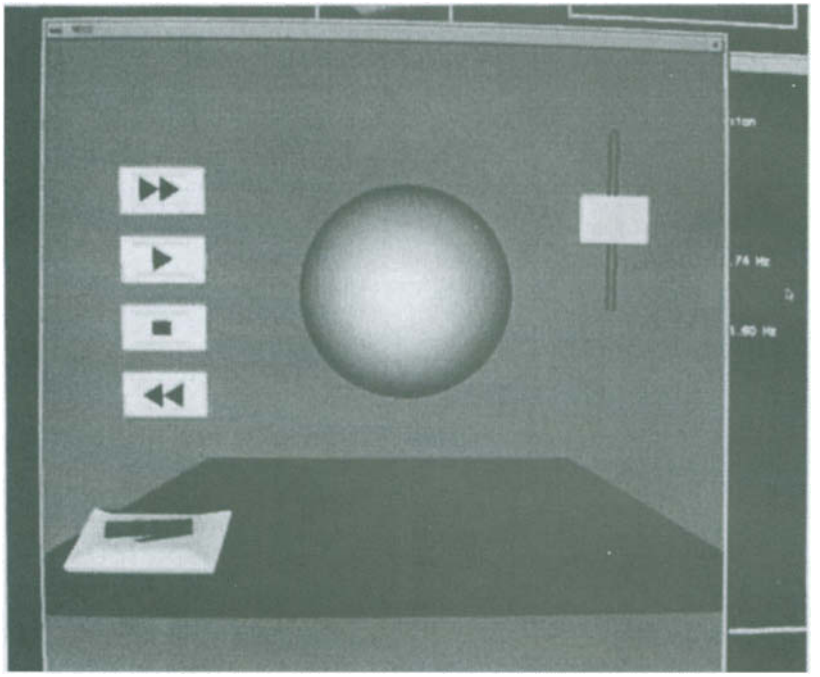


FIGURE 8 User interface of LHX.

Visual information essentially consists of a two-dimensional image. A three-dimensional scene is recognized by binocular parallax cues or motion parallax. Complex 3D objects are often difficult to comprehend because of occlusion. A possible method for visual representation of such objects is semi-transparent graphics. However, multiple objects are overlapped in the image. This drawback leads to difficulty in distinguishing objects.

Visual representation of higher-dimensional and multiparameter data sets is a much harder problem. A typical technique for visualizing those data is iconification. For example, a vector field of fluid dynamics is visualized by streamline. Effectiveness of the technique depends on icon design. Inadequate design of icons leads to a misunderstanding of volume data. Moreover, higher-dimensional values, such as four or five, are difficult to be mapped to icons.

The major objective of our research is representation of volume data by force sensation. Force sensation plays an important role in the recognition of 3D objects. An example of haptic representation of scientific data is found in the work of Brooks *et al.* [1]. A complex molecular docking task is assisted by a force reflective master manipulator. In this work, force display is used for magnifying the length and scale of molecules. We are proposing the haptic mapping of general physical fields.

Force sensation contains six-dimensional information: three-dimensional force and three-dimensional torque. Therefore, higher-dimensional data can be represented by force sensation. The basic idea of volume haptization is mapping voxel data to force and/or torque (Fig. 6). The Haptic Master is used for

volume haptization. The force display is combined with real-time visual images of volume data.

B. Methods of Haptization

Volume data consist of scalar, vector, or tensor data found in a three-dimensional space. Data consisting of single scalar values is the simplest case. Often multiple parameters occur at the same voxel, some of which are sets of scalars and vectors. For example, data from computational fluid dynamics may consist of a scalar for density and vectors for velocity. Visualization of such multiparameter data sets is a difficult problem. A combination of visualization techniques can be used, but there is a danger of the image becoming confusing.

We propose representation of volume data by force sensation. Our force display has an ability to apply six-dimensional force and torque at the fingertips. Values at each voxel can be mapped to force and/or torque. Visual information has an advantage of presenting whole images of objects. On the other hand, haptic information has an advantage in presenting complex attributes of local regions. In our system, a visual image of volume data is represented by direct volume rendering using semi-transparent graphics.

Methods of haptization can be classified into the following three categories:

1. Haptic Representation of Scalar Data

There are two possibilities for mapping scalar data to force/torque. One is mapping scalar values to torque vectors:

$$T_z = a[S(x, y, z)], \tag{1}$$

where $S(x, y, z)$ is a scalar value at each voxel and a is a scaling factor. In this case, the direction of these torque vectors is the same. The user's hand is twisted at each voxel. The other method is mapping gradients of scalar values to three-dimensional force vectors (Fig. 9):

$$F = a[-\text{grad}\, S(x, y, z)]. \tag{2}$$

This formula converts the scalar field to a three-dimensional potential field. The user's hand is pulled toward a low-potential area. This method magnifies the

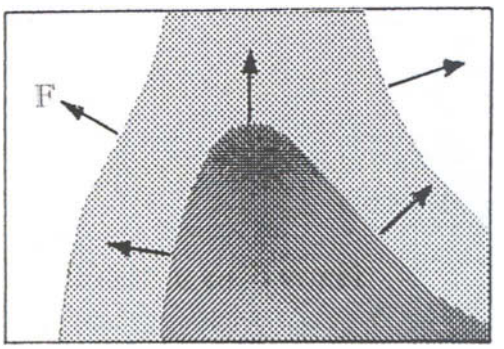


FIGURE 9 Haptic representation of density.

transition area of density data. As for medical imaging such as ultrasound scanning, voxel data are classified according to density. Representation of gradients by force will be effective in such applications.

2. Haptic Representation of Vector/Tensor Data

Vector data have three components, so it can be directly mapped to force:

$$F = aV(x, y, z), \tag{3}$$

where $V(x, y, z)$ are vector data at each voxel. Tensor data is given by a matrix that has nine components. These components cannot be directly mapped to a haptic channel. Some components must be selected according to the user's interest:

$$F \text{ or } T = a(T_{ij}, T_{kl}, T_{mn}), \tag{4}$$

where T_{ij} is a selected component of tensor data at each voxel.

In the case of data from computational fluid dynamics, velocity is mapped to force and one component of vorticity is mapped to torque whose axis has the same direction as the velocity vector (Fig. 10). Velocity and vorticity are simultaneously represented by force display.

3. Haptic Representation of Multiparameter Data Sets

Multiparameter data sets that consist of one or more scalars and vectors can be mapped to force and torque. For example, velocity is mapped to force and density is mapped to one component of torque:

$$F = aV(x, y, z) \tag{5}$$

$$T_z = b[S(x, y, z)]. \tag{6}$$

Representation of multiparameter data sets is rather difficult in the selection of components of a haptic channel. If two different data such as density and

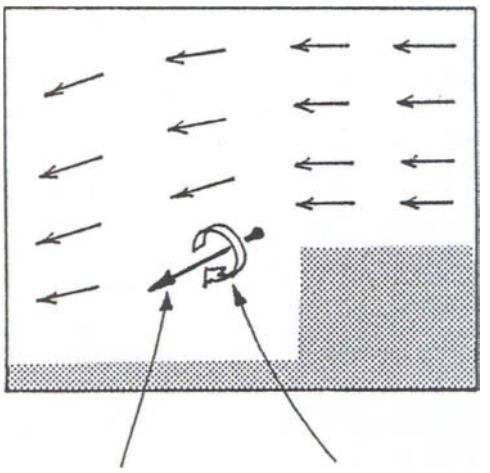


FIGURE 10 Haptic representation of flow field.

temperature are mapped to two components of torque, it may confuse the user. In such case, one scalar data may possibly be represented by auditory channel.

C. Volume Haptics Library

1. Structure of the Volume Haptics Library

LHX includes "volume haptics library." The library supports management of massive volumetric data, and mapping methods of parameters to force. The volume haptics library provides the following three function sets in order to realize environment of volume haptization. Figure 11 shows the structure of these function sets.

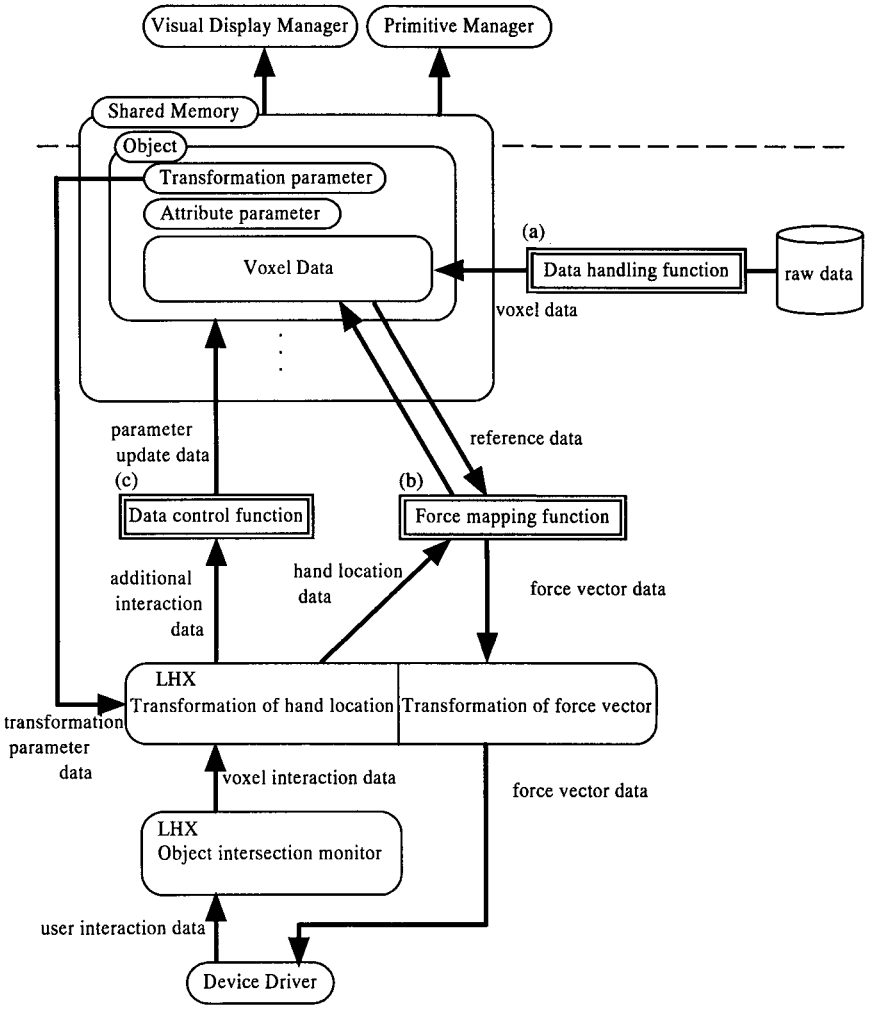


FIGURE 11 Structure of volume haptics library.

a. Data Handling Function

Volumetric data should be stored in the memory in order to represent all of the data as visual/haptic at once. To maintain a high update rate in visual/haptic servo loop, the data management function preloads all of the volumetric data into shared memory and reconstructs original data before the servo loop begins. The data management function also serves as a preprocessor of data for compression or extraction.

b. Mapping Method

We should define how to generate haptic feedback from stored volumetric data. We implemented a force/torque mapping method, which is generated from scalar/vector voxel data to the volume render function. Therefore, minimum change of the function makes it possible to exchange another method of haptic representation. In Fig. 11, this function receives hand location data based on a voxel's local coordinates. Then the function refers to shared memory in order to generate the force vector.

c. Data Control Function

In order to treat volumetric data as a primitive, it is necessary to attach the primitive attribute to the volumetric data. The data control function gives such additional primitive parameters so that LHX can utilize various functions such as position transformation or collision detection. As the user moves/rotates the volumetric data, the data control function updates the transformation parameter, and then LHX looks up the parameter to change the coordinate system of the user's hand.

2. Application of Volume Haptics Library

Two applications have been developed by using the three functions mentioned in the previous section. The first one is a multidimensional data browser that presents virtual 4D or higher-dimensional space represented by visual and haptic sensation [16]. In principle force sensation contains six-dimensional information: three-dimensional force and three-dimensional torque. Therefore, six-dimensional data can be represented by force sensation. Our multidimensional space where the data are exposed as volumetric data are geometrically generated by scanning a 3D cube. The user's hand can essentially move in 3D space. We therefore use rotational motion of the hand for scanning a 3D cube in a multidimensional cube. The 3D cube is cutout of the volume of the multidimensional cube, which moves by rotational motion around the roll and the pitch axes of the user's hand. Force display presents the potential field, which indicates the axis of rotation. The user can easily separate rotational motion from transnational motion by a haptic guide.

The second application is a noninvasive surgery support system. This system proposes a method of haptic representation of a noninvasive area to support the system of microsurgery [17]. In case the virtual tool approaches the noninvasive area, force is applied to the operator's hand. Generated force is determined by volume data of the CT image. Figure 12 shows haptic representation of CT data.

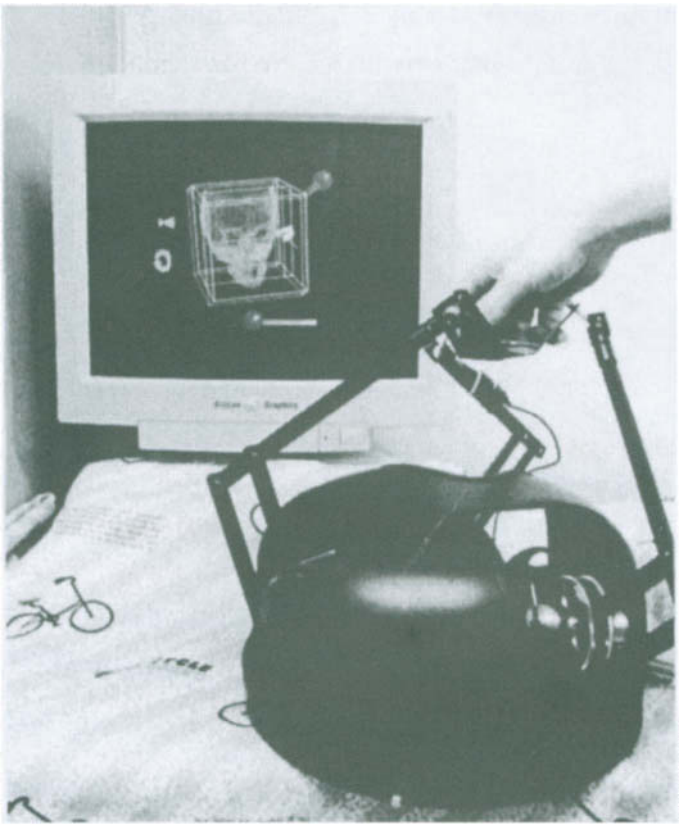


FIGURE 12 Haptic representation of CT data.

IV. APPLICATIONS OF LHX: 3D SHAPE DESIGN USING AUTONOMOUS VIRTUAL OBJECT

A. Shape Design and Artificial Life

The design of 3D shapes is a major application area of virtual reality. Direct manipulation of 3D shapes has been realized in virtual environments. In this case, the shape of the object is originally determined by a human designer. This chapter proposes autonomously growing objects in a virtual environment. Designers are often inspired by the shapes of living creatures.

The creation of living systems by computer software is called artificial life. The major objective of research on artificial life is duplicating the emergent mechanics of biology. We have tried to introduce the emergent mechanics in a virtual environment. The user of our system can interact with artificial life in its morphological process. Parameters of congenital characteristics can be interactively changed. An artificial life shape can be directly manipulated by the user during its growing process. In this way, the autonomous shape and the intentional shape are mixed. Artificial life is implemented as a user application of LHX.

B. Methods for Interaction with Autonomous Virtual Objects

Interaction with autonomous virtual objects can be done through the following three parameters:

1. Time

Our virtual environment has a time clock supervised by the kernel of LHX. Growth or autonomous deformation occurs according to this time clock. The user can freely change the speed of time, and he/she can also reverse it. Direct manipulation of time increases the freedom of the design process.

2. Congenital Characteristics

Autonomous 3D shapes have congenital characteristics, which determine their morphological process. The user can arbitrary change the congenital characteristics. The variation of congenital characteristics results in different shapes. Manipulation of congenital characteristics leads to unexpected forms of virtual objects. In our system, congenital characteristics can be changed in the virtual objects growing process.

3. Acquired Characteristics

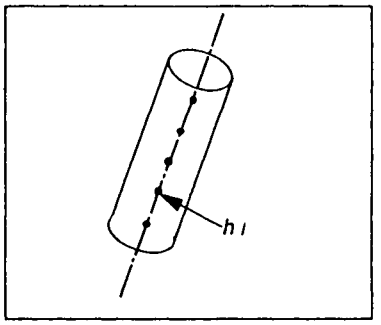
Acquired characteristics are obtained by direct manipulation of virtual objects by the user, who can deform autonomous virtual objects at any time.

LHX supports implementation of these parameters.

C. Direct Manipulation of Tree-Like Artificial Life

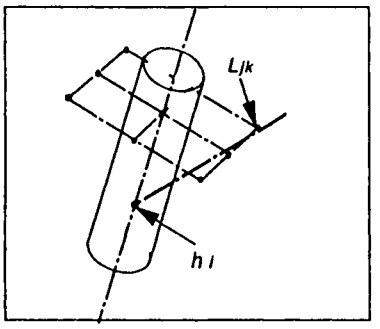
A tree-like virtual object is selected as artificial life. The principle of growth of the tree is illustrated in Fig. 13. A cylinder in the figure indicates the trunk of the tree. The length and diameter of the cylinder increases according to the time clock of LHX. Ramification occurs in preset frequency. At first, the ramification point h_i is randomly selected from five points as shown in Fig. 13a. Then growth direction L_{jk} is randomly selected from eight point as shown in Fig. 13b. After these selection, a new cylinder appears (Fig. 13c). This principle is applied to the new cylinder. Figure 14 shows an example of the growth of a tree.

The user can directly manipulate the tree-like artificial life by three parameters: time, congenital characteristics, and acquired characteristics. The time of the virtual environment is controlled by four buttons: start, stop, fast forward, and reverse. These buttons are set on the left hand-side of the virtual environment. Operation of the buttons is similar to that of a VCR. Congenital characteristics are determined by the frequency of ramification, which is controlled by a slide bar located on the right-hand side of the virtual environment. Acquired characteristics are determined by chopping off the branches by the user's virtual hand. Branches can be chopped off at any ramification point while the tree is growing. The user can intentionally form the final shape of the tree. This procedure is similar to "bonsai" or dwarf-tree culture.



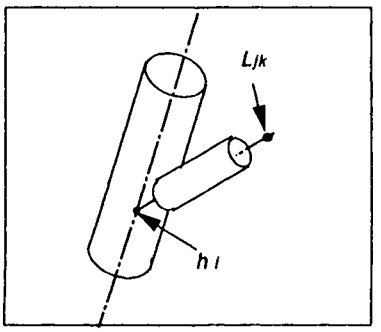
(a)

SELECT
RAMIFICATION
POINT *h i*



(b)

SELECT GROWTH
DIRECTION *Ljk*



(c)

SET BRANCH

FIGURE 13 Principles of growth.

D. Manipulation of Autonomous Free-Form Surface

1. Functions of Autonomous Free-Form Surface

Industrial designers often make a simple mockup using urethane or styrene boards in order to study the theme of the form at the early stage of design development. This process is called form study. We developed an autonomous free-from surface in a virtual environment as a tool for form study [4]. The original shape of the surface is a sphere. The user can freely deform the surface.

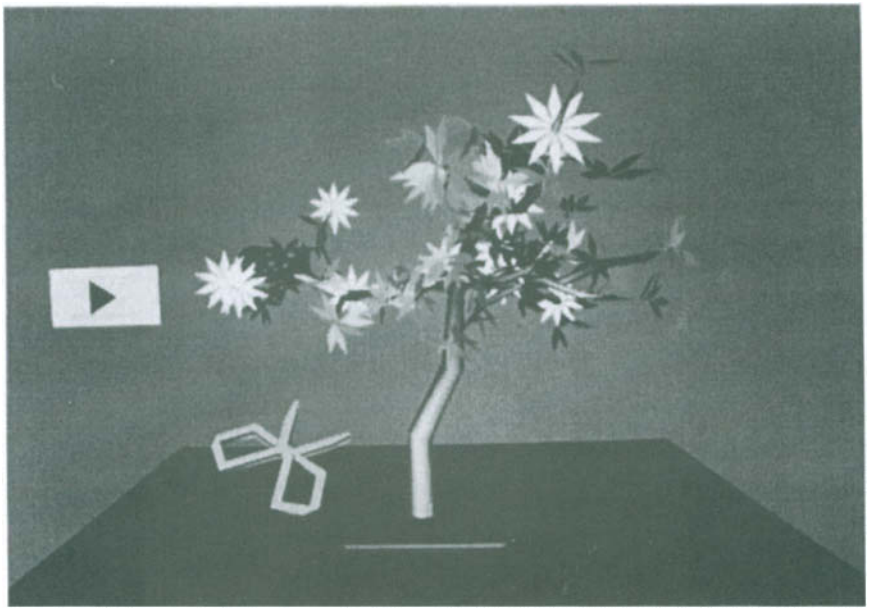


FIGURE 14 Virtual tree.

The basic theme of the form is created through deformation of the sphere. We implemented three functions in the autonomous surface:

a. Restoration

Each lattice of the surface is connected by four springs (Fig. 15). These springs generate surface tension. If the surface is pulled or pushed by the user, the surface tension restores the surface to the original sphere (Fig. 16). Unexpected shapes are found during the autonomous deformation process. The user can manipulate the surface while it is autonomously deforming. In

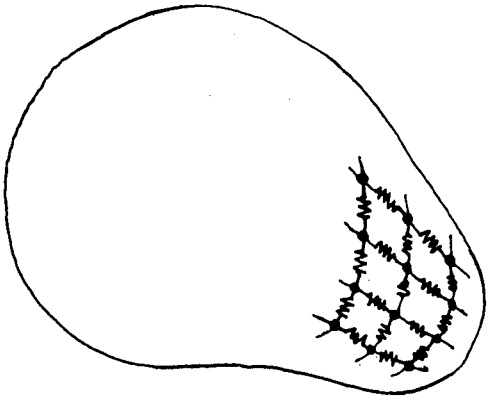


FIGURE 15 Mechanism of surface tension.

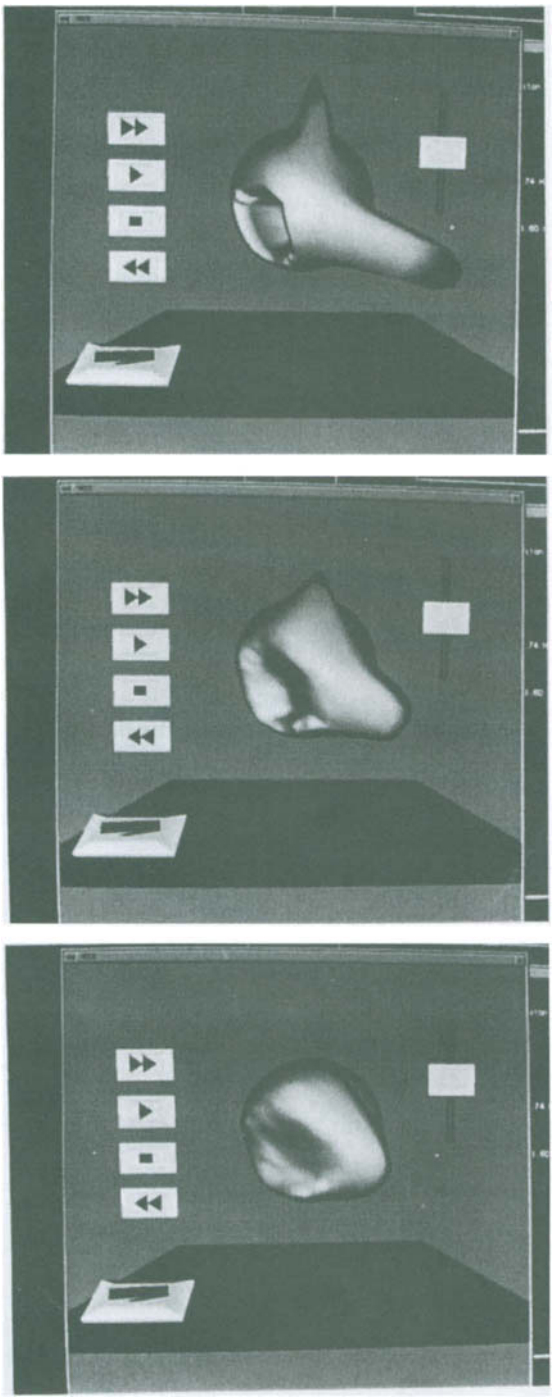


FIGURE 16 An example of restoration.

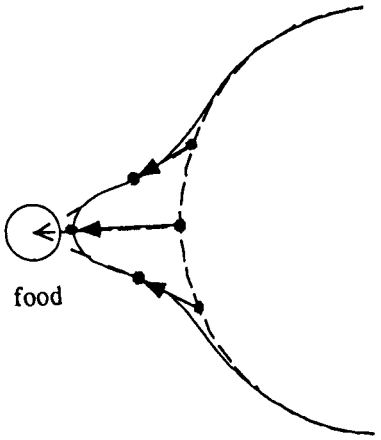


FIGURE 17 Mechanism of reaching action to food.

this case, congenital characteristics are determined by the spring constant of surface tension. The user can change the spring constant by a slide bar. If the spring constant is large, deformation spreads to a wider area.

b. Reaching Action to Food

The user can put out "food" for the surface. If the food comes near, the surface reaches out to it. The motion of each lattice is caused according to the distance between the surface and the food. Transition from the original surface is determined by a negative exponential function (Fig. 17). If the food is large, a large area of the surface deforms.

c. Avoidance from Enemy

The user can put out "enemy" for the surface. If the enemy comes near, the surface avoids it. The motion of each lattice is caused according to the distance between the surface and the enemy. Transition from the original surface is

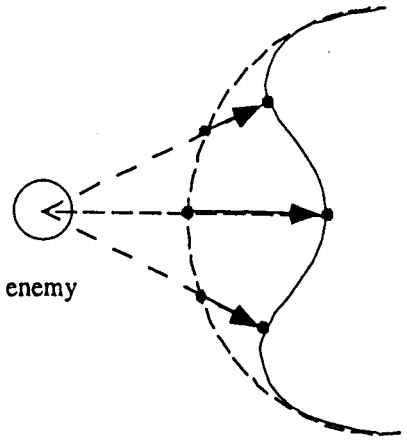


FIGURE 18 Mechanism of avoidance from enemy.

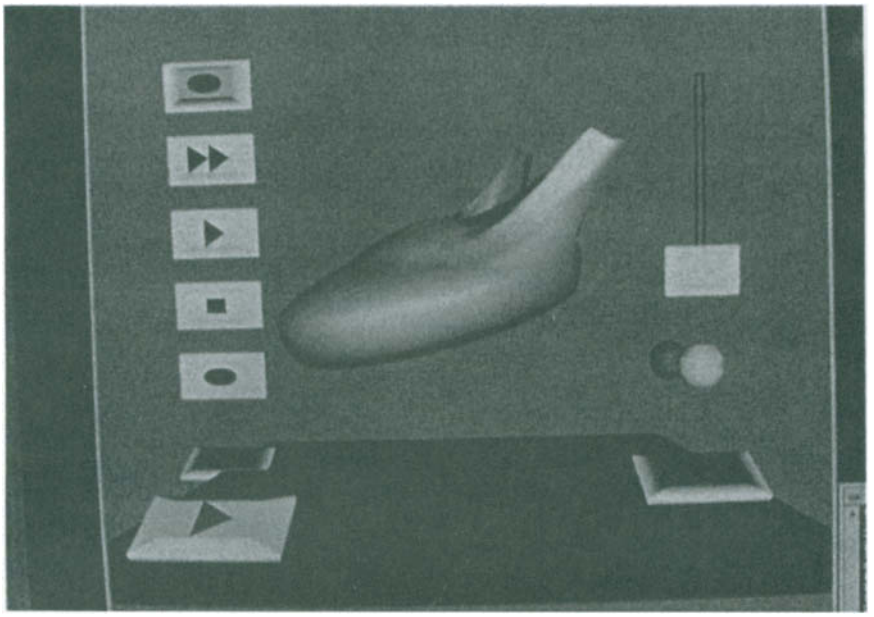


FIGURE 19 An example of a pattern for the form study of car design.

determined by a negative exponential function (Fig. 18). If the enemy is large, a large area of the surface deforms.

2. Usability Study

As a usability tests of the autonomous free-form surface, we examined the modeling task of a complex curved surface. We prepared three patterns, which represent shapes of form study for car design. Figure 19 shows an example of the patterns. Subjects are instructed to make these patterns from a spherical surface.

Two conditions are set for the experiment:

(1) *With function of restoration.* The surface autonomously deforms according to its surface tension.

(2) *Without function of restoration.* In this case, the surface is passive. The subjects make shapes by direct manipulation of the free-form surface. A deformed surface is determined by a sine curve.

We took six volunteer subjects from the students of our university. We examined the accuracy of the modeling task. The deformation distance at each lattice from the original surface was calculated. The total deformation distance was normalized by that of the target pattern. If the error value is 1, the task was perfect. Figure 20 shows the normalized deformation distance of the two conditions. The value of condition (1) is nine times smaller than that of condition (2). The required time for each task was approximately 2 min. for both conditions. Subjects often abandoned making shapes under condition (2). The results show that an autonomous free-form surface improves design tasks.

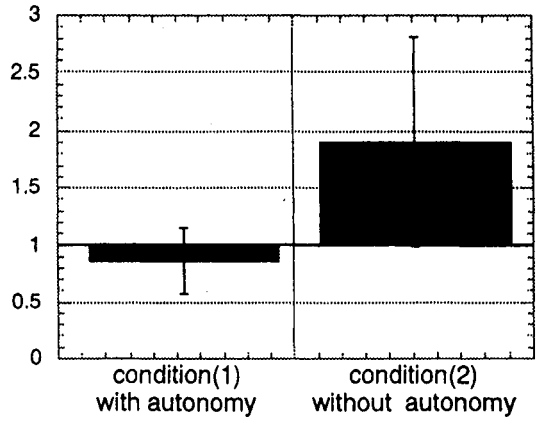


FIGURE 20 Accuracy of modeling.

V. OTHER APPLICATIONS OF LHX

A. Surgical Simulator

Laparoscopic surgery requires training in a virtual environment. We are collaborating with Olympus Co. on developing a training simulator for laparoscopy [15]. Two specialized HapticMasters are used for both hands. LHX supports two force displays simultaneously.

Primitives of LHX include autonomous free-form surfaces. The surface has functions similar to that of living creatures. The free-form surface has surface tension, which enables restoration to the original shape. We developed virtual tissue by using an autonomous free-form surface. An operator can feel viscoelasticity of the virtual tissue. The virtual tissue can be cut at any place in realtime.

B. Shared Haptic World

Existing communication media supports only visual and auditory information. We have tried to use haptic information as a tool for communication. We used the network capability of LHX to connect two force displays in the same virtual environment. The two users of our system were able to feel the reaction force simultaneously. They can cooperatively manipulate virtual objects. One user can grasp the other user's virtual hand, and feel the force applied by the other user. This function is beneficial for trainer–trainee interaction.

In the case of using the Internet as a communication line, there is a time delay between the two force displays. We developed visual and haptic aid to assist the work under a time delay. Users of the system were able to feel viscosity while moving the hands. The force realizes natural collaboration under a time delay.

At SIGGRAPH'95, we connected two force displays by Internet. One was located in Los Angels and the other in Tsukuba, Japan. We succeeded in cooperative work between those remote sites [6]. The time delay was constantly 0.4 s.

C. HapticWeb

The "HapticWeb" is a WWW client, which enables a user to feel the rigidity or weight of a virtual object. HapticWeb uses the haptic renderer of LHX. It realizes force feedback from the VRML data set. Users can feel the rigidity or weight of virtual objects stored in a WWW server. The system was demonstrated at SIGGRAPH'96.

We also developed an authoring tool for the HapticWeb. Parameters of the rigidity and weight of virtual objects are presented by 3D icons. The user can change the rigidity or weight of virtual objects by manipulating slide bars.

We observed the behavior of users at SIGGRAPH'96. A total of 647 people experienced our system. Of these, 637 people (98%) could feel force feedback, 446 people (69%) found haptic icons, and 642 people (99%) of them could manipulate the slide bars. The result shows that the design of the haptic icon was successful.

VI. CONCLUSION

This chapter introduced techniques and applications in the development of a database system for a haptic environment. We developed a software infrastructure for a haptic interface named LHX. It achieved the reproductivity of software of virtual environments with force feedback. The software tool has been improved through various applications including haptization of scientific data and 3D shape manipulation. Future work will be the further development of practical software tools such as the volume haptic library.

REFERENCES

1. Brooks, F. P. *et al.* Project GROPE—Haptic displays for scientific visualization. *ACM SIGGRAPH Comput. Graphics* 24(4):177–185, 1990.
2. Iwata, H. Artificial reality with force-feedback: Development of desktop virtual space with compact master manipulator. *ACM SIGGRAPH Comput. Graphics* 24(4):165–170, 1990.
3. Iwata, H., and Yano, H. Artificial life in haptic virtual environment. In *Proceedings of ICAT'93,* pp. 91–96, 1993.
4. Iwata, H., and Yano, H. Interaction with autonomous free-form surface. In *Proceedings of ICAT'94,* pp. 27–32, 1994.
5. Iwata, H. Desktop force display. In *SIGGRAPH'94 Visual Proceedings,* 1994.
6. Yano, H., and Iwata, H. Cooperative work in virtual environment with force feedback. In *Proceedings of ICAT/VRST'95,* pp. 203–210, 1995.
7. *World Tool Kit User's Guide,* SENSE8.
8. dVS, http://www.ptc.com/products/division/index.htm.
9. ViScape, available at http://www.superscape.com.
10. Mark, W. R. *et al.* Adding force feedback to graphics system: Issues and solutions. In *Proceedings of SIGGRAPH'96,* pp. 447–452, 1996.
11. Brochure of SensAble Technologies. GHOST: General Haptics Open Software Toolkit, 1996.
12. Iwata, H., and Hayakawa, K. Force display for grasping virtual object by three fingers. In *Proceedings of the Eleventh Symposium on Human Interface,* pp. 395–400, 1995. [In Japanese.]
13. Iwata, H., and Ichigaya, A. Haptic screen. In *Proceedings of the Virtual Society of Japan Annual Conference,* Vol. 1, pp. 7–10, 1996. [In Japanese.]

14. Iwata, H., and Noma, H. Volume haptization. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality,* pp. 91–96, 1993.
15. Asano, T., Yano, H., and Iwata, H. Basic technology of simulation system for laparoscopic surgery in virtual environment with force display. In *Medicine Meets Virtual Reality,* pp. 207–215, IOS Press.
16. Hashimoto, W., and Iwata, H. Multi-dimensional data browser with haptic sensation. In *Transactions of the Virtual Reality Society of Japan,* Vol. 2, No. 3, pp. 9–16, 1997. [In Japanese.]
17. Hashimoto, W., and Iwata, H. Haptic representation of non-invasive region for surgery based on volumetric Data. In *Transactions of the Virtual Reality Society of Japan,* Vol. 3, No. 4, pp. 197–202, 1998. [In Japanese.]

16

DATA COMPRESSION IN INFORMATION RETRIEVAL SYSTEMS

SHMUEL TOMI KLEIN

Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel

I. INTRODUCTION

As can be seen from the title, we shall concentrate on techniques that are at the crossroads of two disciplines: *data compression* (DC) and *information retrieval* (IR). Each of these encompass a large body of knowledge that has evolved over the past decades, each with its own philosophy and its own scientific community. Nevertheless, their intersection is particularly interesting, the various files of large full-text IR systems providing a natural testbed for new compression methods, and DC enabling the proliferation of improved retrieval algorithms.

A chapter about data compression in a book published at the beginning of the twenty-first century might at a first glance seem anachronistic. Critics will say that storage space is getting cheaper every day; tomorrow it will be

Database and Data Communication Network Systems, Vol. 2

almost given for free, so who needs complicated methods to save a few bytes. What these critics overlook is that for data storage, supply drives demand: our appetite for getting ever-increasing amounts of data into electronic storage grows just as steadily as does the standard size of the hard disk in our current personal computer. Most users know that whatever the size of their disks, they will fill up sooner or later, and generally sooner than they wish.

However, there are also other benefits to be gained from data compression, beyond the reduction of storage space. One of the bottlenecks of our computing systems is still the slow data transfer from external storage devices. Similarly, for communication applications, the problem is not storing the data but rather squeezing it through some channel. However, many users are competing for the same limited bandwidth, effectively reducing the amount of data that can be transferred in a given time span. Here, DC may help reduce the number of I/O operations to and from secondary memory, and for communication it reduces the actual amount of data that must pass through the channel. The additional time spent on compression and decompression is generally largely compensated for by the savings in transfer time.

For these reasons, research in DC is not dying out, but just the opposite is true, as evidenced by the recent spurt of literature in this area. An international Data Compression Conference has convened annually since 1990, and many journals, including even popular ones such as *Byte, Dr. Dobbs, IEEE Spectrum, Datamation, PC Magazine,* and others, have repeatedly published articles on compression recently.

It is true that a large part of the research concentrates on image compression. Indeed, pictorial data are storage voracious so that the expected profit of efficient compression is substantial. The techniques generally applied to images belong to the class of *lossy* compression, because they concentrate on how to throw away part of the data, without too much changing its general appearance. For instance, most humans do not really see any difference between a picture coded with 24 bits per pixel, allowing more than 16 million colors, and the same picture recoded with 12 bits per pixel, giving "only" about 4000 different color shades. Of course, most image compression techniques are much more sophisticated, but we shall not deal with them in the present survey. The interested reader is referred to the large literature on lossy compression, e.g., [1].

Information retrieval is concerned, on the one hand, with procedures for helping a user satisfy his information needs by facilitating his access to large amounts of data, on the other hand, with techniques for evaluating his (dis)satisfaction with whatever data the system provided. We shall concentrate primarily on the algorithmic aspects of IR. A functional full-text retrieval system is constituted of a large variety of files, most of which can and should be compressed. Some of the methods described below are of general applicability, and some are specially designed for an IR environment.

Full-text information retrieval systems may be partitioned according to the level of specificity supported by their queries. For example, in a system operating at the *document* level, queries can be formulated as to the presence of certain keywords in each document of the database, but not as to their exact locations within the document. Similarly, one can define the *paragraph* level and *sentence* level, each of which is a refinement of its predecessor. The highest specificity

level is the *word* level, in which the requirement is that the keywords appear within specified distances of each other. With such a specificity level, one could retrieve all the occurrences of A and B such that there are at least two but at most five words between them. In the same way, the paragraph and sentence levels permit also appropriate distance constraints; e.g., at the sentence level one could ask for all the occurrences of A and B in the same or adjacent sentences.

Formally, a typical query consists of an optional level indicator, m keywords, and $m-1$ distance constraints, as in

$$level : A_1\,(l_1, u_1)\,A_2\,(l_2, u_2)\cdots A_{m-1}(l_{m-1}, u_{m-1})\,A_m. \tag{1}$$

The l_i and u_i are (positive or negative) integers satisfying $l_i \leq u_i$ for $1 \leq i < m$, with the couple (l_i, u_i) imposing lower and upper limits on the distance from A_i to A_{i+1}. Negative distance means that A_{i+1} may appear before A_i in the text. The distance is measured in words, sentences, or paragraphs, as prescribed by the level indicator. In case the latter is omitted, word level is assumed; in this case, constraints of the form $A\ (1, 1)\ B$ (meaning that A should be followed immediately by B) are omitted. Also, if the query is on the document level, then the distances are meaningless and should be omitted (the query degenerates then into a conjunction of the occurrences of all the keywords in the query).

In its simplest form, the keyword A_i is a single word or a (usually very small) set of words given explicitly by the user. In more complex cases a keyword A_i in (1) will represent a set of words $A_i = \bigcup_{j=1}^{n_i} A_{ij}$, all of which are considered synonymous to A_i in the context of the given query. For example, a variable-length-don't-care-character $*$ can be used, which stands for an arbitrary, possibly empty, string. This allows the use of prefix, suffix, and infix truncation in the query. Thus A_i could be `comput*`, representing, among others, the words `computer`, `computing`, `computerize`, etc., or it could be `*mycin`, which retrieves a large class of antibiotics; infix truncation also can be useful for spelling foreign names, such as `Ba*tyar`, where $*$ could be matched by `h`, `k`, `kh`, `ch`, `sh`, `sch`, etc.

Another possibility for getting the variants of a keyword is from the use of a *thesaurus* (`month` representing `January`, `February`, etc.), or from some morphological processing (`do` representing `does`, `did`, `done`, etc.). Although these grammatical variants can be easily generated in some languages with simple morphology like English, sophisticated linguistic tools are needed for languages such as Hebrew, Arabic, and many others. One of the derivatives of the 2-character word `daughter` in Hebrew, for example, is a 10-character string meaning `and when our daughters`, and it shares only *one* common letter with its original stem; a similar phenomenon occurs in French with the verb `faire`, for example.

For all these cases, the families A_i are constructed in a preprocessing stage. Algorithms for generating the families identified by truncated terms can be found in [2], and for the families of grammatical variants in [3].

This general definition of a query with distance constraints allows great flexibility in the formulation of the query. For example, the query `solving (1,3) differential equations` will retrieve sentences containing `solving differential equations`, as well as `solving these differential equations` and `solving the required differential equations`, but not `solving`

these systems of differential equations. The query true (−2,2) false can be used to retrieve the phrases true or false and false or true; since these words appear frequently in some mathematical texts, searching for true and false in the same sentence could generate noise. A lower bound greater than 1 in the distance operators is needed, for example, when one wishes to locate phrases in which some words $X_1, X_2, \ldots$ appear, but the juxtaposition of these words $X_1 X_2 \ldots$ forms an idiomatic expression that we do not wish to retrieve. For example, ... the security of the council members assembled here ... should be retrieved by the query security (2,4) council. Note however that (1) implies that one can impose distance constraints only on adjacent keywords. In the query A (1,5) B (2,7) C, the pair (2,7) refers to the distance from B to C. If we wish to impose positive bounds on the distances from A to both B and C, this can be done by using negative distances, C (−7,−2) A (1,5) B, but this procedure cannot be generalized to tying more than two keywords to A.

A well-known problem in retrieval systems is the handling of "negative" keywords, i.e., words, the *absence* of which, in a specified distance from a specified context, is required. A negation operator (represented here by the minus sign −) is particularly useful for excluding known homonyms so as to increase precision. For example, searching for references to the former U.S. President, one could submit the query Reagan (−2,1) −Donald. Another interesting example would be to use the constraints $(l_i, u_i) = (0,0)$ in order to restrict some large families of keywords, as in the example comput* (0,0) −computer*, which would retrieve computing, computation, etc, but not computer or computers. The general definition of a query as given in (1) should therefore include the possibility of negating some—but not all—of the keywords while specifying their appropriate distance constraints.

Queries of type (1) can be of course further combined by the Boolean operators of AND, OR, and NOT, but we shall restrict our attention here to queries of type (1), since they are quite common, on the one hand, and on the other, their efficient processing is anyway a prerequisite to the efficient processing of more complicated ones.

At the end of the search process, the solutions are presented to the user in the form of a list of the identifying numbers or the titles of the documents that contain at least one solution, possibly together with the text of the sentence (or the paragraph) in which this solution occurs. The exact details of the display depend on the specific system, on the target population, and on the human–interface design of the system.

The way to process such queries depends on the size of the database. When the size of the text is small, say up to a few hundred kilobytes, the problem of efficiently accessing the data can generally be solved by some brute-force method that scans the whole text in reasonable time. Such a method is commonly used in text editors. At the other extreme, for very large databases spanning hundreds of megabytes, a complete scan is not feasible. The usual approach in that case is to use so-called *inverted files*.

Every occurrence of every word in the database can be uniquely characterized by a sequence of numbers that give its exact position in the text; typically, in a word-level retrieval system, such a sequence would consist of the document number, the paragraph number (in the document), the sentence number (in the paragraph), and the word number (in the sentence). These are the *coordinates*

of the occurrence. For every word W, let $C(W)$ be the ordered list of the coordinates of all its occurrences in the text. The problem of processing a query of type (1) consists then, in its most general form, of finding all the m-tuples $(a_1, \ldots, a_m)$ of coordinates satisfying

$$\forall i \in \{1, \ldots, m\} \quad \exists j \in \{1, \ldots, n_i\} \quad \text{with} \quad a_i \in C(A_{ij})$$

and

$$l_i \leq d(a_i, a_{i+1}) \leq u_i \quad \text{for } 1 \leq i < m,$$

where $d(x, y)$ denotes the distance from x to y on the given level. Every m-tuple satisfying these two equations is called a *solution.*

In the inverted files approach, processing (1) does not involve directly the original text files, but rather the auxiliary *dictionary* and *concordance* files. The concordance contains, for each distinct word W in the database, the ordered list $C(W)$ of all its coordinates in the text; it is accessed via the dictionary that contains for every such word a pointer to the corresponding list in the concordance. For each keyword A_i in (1) and its attached variants A_{ij}, the lists $C(A_{ij})$ are fetched from the concordance and merged to form the combined list $C(A_i)$. Beginning now with A_1 and A_2, the two lists $C(A_1)$ and $C(A_2)$ are compared, and the set of all pairs of coordinates (a_1, a_2) that satisfy the given distance constraints (l_1, u_1) at the appropriate level is constructed. (Note that a unique a_1 can satisfy the requirements with different a_2, and vice versa.) $C(A_2)$ is now purged from the irrelevant coordinates, and the procedure is repeated with A_2 and A_3, resulting in the set $\{(a_1, a_2, a_3)\}$ of partial solutions of (1). Finally, when the last keyword A_m is processed in this way, we have the required set of solutions.

Note that it is not really necessary to always begin the processing with the first given keyword A_1 in (1), going all the way in a left-to-right mode. In some cases, it might be more efficient to begin it with a different keyword A_j, and to proceed with the other keywords in some specified order.

The main drawback of the inverted files approach is its huge overhead: the size of the concordance is comparable to that of the text itself and sometimes larger. For the intermediate range, a popular technique is based on assigning *signatures* to text fragments and to individual words. The signatures are then transformed into a set of bitmaps, on which Boolean operations, induced by the structure of the query, are performed. The idea is first to effectively reduce the size of the database by removing from consideration segments that cannot possibly satisfy the request, then to use pattern-matching techniques to process the query, but only over the—hopefully small—remaining part of the database [4]. For systems supporting retrieval only at the document level, a different approach to query processing might be useful. The idea is to replace the concordance of a system with ℓ documents by a set of *bitmaps* of fixed length ℓ. Given some fixed ordering of the documents, a bitmap $B(W)$ is constructed for every distinct word W of the database, where the ith bit of $B(W)$ is 1 if W occurs in the ith document, and is 0 otherwise. Processing queries then reduces to performing logical OR/AND operations on binary sequences, which is easily done on most machines, instead of merge/collate operations on more general sequences. Davis and Lin [5] were apparently the first to propose the use of bitmaps for secondary key retrieval. It would be wasteful to store the bitmaps in their original form, since they are usually very sparse (the great majority of

the words appear in very few documents), and we shall review various methods for the compression of such large sparse bit-vectors. However, the concordance can be dropped only if *all* the information we need is kept in the bitmaps. Hence, if we wish to extend this approach to systems supporting queries also at the paragraph, sentence, or word level, the length of each map must equal the number of paragraphs, sentences, or words respectively, a clearly infeasible scheme for large systems. Moreover, the processing of distance constraints is hard to implement with such a data structure.

In [6], a method in which, basically, the concordance and bitmap approaches are combined is presented. At the cost of marginally expanding the inverted files' structure, compressed bitmaps are *added* to the system; these maps give *partial* information on the location of the different words in the text and their distribution. This approach is described in more detail in Section V.

Most of the techniques below were tested on two real-life full-text information retrieval systems, both using the inverted files approach. The first is the *Trésor de la Langue Française* (TLF) [7], a database of 680 MB of French language texts (112 million words) made up from a variety of complete documents including novels, short stories, poetry, and essays, by many different authors. The bulk of the texts are from the 17th through 20th centuries, although smaller databases include texts from the 16th century and earlier. The other system is the *Responsa Retrieval Project* (RRP) [8], 350 MB of Hebrew and Aramaic texts (60 million words) written over the past ten centuries. For the sake of conciseness, detailed experimental results have been omitted throughout.

Table 1 shows roughly what one can expect from applying compression methods to the various files of a full-text retrieval system. The numbers correspond to TLF. Various smaller auxiliary files are not mentioned here, including grammatical files, and thesauri.

For the given example, the overall size of the system, which was close to 2 Gbytes, could be reduced to fit onto a single CD-Rom.

The organization of this chapter is as follows. The subsequent sections consider, in turn, compression techniques for the file types mentioned above, namely, the text, dictionaries, concordances, and bitmaps. For text compression, we first shortly review some background material. While concentrating on Huffman coding and related techniques, arithmetic coding and dictionary-based text compression are also mentioned. For Huffman coding, we focus in particular on techniques allowing fast decoding, since decoding is more important than encoding in an information retrieval environment. For dictionary and

TABLE 1 Files in a Full-Text System

File	Full size (MB)	Compressed size (MB)	Compression (%)
Text	700	245	65
Dictionary	30	18	40
Concordance	400	240	40
Bitmaps	800	40	95
Total		543	

concordance compression the prefix omission method and various variants are suggested. Finally, we describe the usefulness of bitmaps for the enhancement of IR systems and then show how these large structures may in fact be stored quite efficiently.

The choice of the methods to be described is not meant to be exhaustive. It is a blend of techniques that reflect the personal taste of the author rather than some well-established core curriculum in information retrieval and data compression. The interested reader will find pointers to further details in the appended references.

II. TEXT COMPRESSION

We are primarily concerned with information retrieval; therefore this section will be devoted to text compression, as the text is still the heart of any large full-text IR system. We refer to text written in some natural language, using a fixed set of letters called an *alphabet*. It should, however, be noted that the methods below are not restricted to textual data alone, and are in fact applicable to any kind of file. For the ease of discourse, we shall still refer to texts and characters, but these terms should not be understood in their restrictive sense.

Whatever text of other file we wish to store, our computers insist on talking only binary, which forces us to transform the data using some binary *encoding*. The resulting set of elements, called *codewords*, each corresponding to one of the characters of the alphabet, is called a *code*. The most popular and easy to use codes are *fixed-length* codes, for which all the codewords consist of the same number of bits. One of the best-known fixed-length codes is the American Standard Code for Information Interchange (ASCII), for which each codeword is 1 byte (8 bits) long, providing for the encoding of $2^8 = 256$ different elements.

A fixed-length code has many advantages; most obviously, the encoding and decoding processes are straightforward. Encoding is performed by concatenating the codewords corresponding to the characters of the message; decoding is done by breaking the encoded string into blocks of the given size, and then using a decoding table to translate the codewords back into the characters they represent. For example, the ASCII representation of the word `Text` is

$$01010100011001010111100001110100,$$

which can be broken into

$$01010100 \mid 01100101 \mid 01111000 \mid 01110100.$$

From the compression point of view, such a code may be wasteful. A first attempt to reduce the space of an ASCII encoding is to note that if the actual character set used is of size n, only $\lceil \log_2 n \rceil$ bits are needed for each codeword. Therefore a text using only the 26 letters of the English alphabet (plus up to six special characters, such as space, period, comma, etc.) could be encoded using just five bits per codeword, saving already 37.5%. However, even for larger alphabets an improvement is possible if the frequency of occurrence of the different characters is taken into account.

As is well known, not all the letters appear with the same probability in natural language texts. For English, `E`, `T`, and `A` are the most frequent, appearing

A	0	A	11	A	11	A	1
B	01	B	110	B	011	B	00
C	110	C	1100	C	0011	C	010
D	1011	D	1101	D	1011	D	0110
		E	11000	E	00011	E	0111
	(a)		(b)		(c)		(d)

FIGURE 1 Examples of codes: (a) Non-UD, (b) UD nonprefix, (c) prefix noncomplete, and (d) complete.

about 12, 10, and 8% respectively, while J, Q, and Z occur each with probability less than 0.1%. Similar phenomena can be noted in other languages. The skewness of the frequency distributions can be exploited if one is ready to abandon the convenience of fixed-length codes, and trade processing ease for better compression by allowing the codewords to have *variable length*. It is then easy to see that one may gain by assigning shorter codewords to the more frequent characters, even at the price of encoding the rare characters by longer strings, as long as the *average* codeword length is reduced. Encoding is just as simple as with fixed-length codes and still consists in concatenating the codeword strings. There are however a few technical problems concerning the decoding that must be dealt with.

A *code* has been defined above as a set of codewords, which are binary strings, but not every set of strings gives a useful code. Consider, for example, the four codewords in column (a) of Fig. 1. If a string of 0's is given, it is easily recognized as a sequence of A's. Similarly, the string 010101 can only be parsed as BBB. However, the string 010110 has two possible interpretations: 0 | 1011 | 0 = ADA or 01 | 0 | 110 = BAC. This situation is intolerable, because it violates our basic premise of reversibility of the encoding process. We shall thus restrict attention to codes for which *every* binary string obtained by concatenating codewords can be parsed only in one way, namely into the original sequence of codewords. Such codes are called *uniquely decipherable* (UD).

At first sight, it seems difficult to decide whether a code is UD, because infinitely many potential concatenations must be checked. Nevertheless, efficient algorithms solving the problem do exist [9]. A necessary, albeit not sufficient, condition for a code to be UD is that its codewords should not be too short. A precise condition has been found by McMillan [10]: any binary UD code with codewords lengths $\{\ell_1, \ldots, \ell_n\}$ satisfies

$$\sum_{i=1}^{n} 2^{-\ell_i} \leq 1. \tag{2}$$

For example, referring to the four codes of Figure 1, the sum is 0.9375, 0.53125, 0.53125, and 1 for codes (a) to (d) respectively. Case (a) is also an example showing that the condition is not sufficient.

However, even if a code is UD, the decoding of certain strings may not be so easy. The code in column (b) of Fig. 1 is UD, but consider the encoded string 11011111110: a first attempt to parse it as 110 | 11 | 11 | 11 | 10 = BAAA10 would fail, because the tail 10 is not a codeword; hence only when trying to

decode the fifth codeword do we realize that the first one is not correct, and that the parsing should rather be 1101 | 11 | 11 | 110 = DAAB. In this case, a codeword is not immediately recognized as soon as all its bits are read, but only after a certain *delay*. There are codes for which this delay never exceeds a certain fixed number of bits, but the example above is easily extended to show that the delay for the given code is unbounded.

We would like to be able to recognize a codeword as soon as all its bits are processed, that is, with no delay at all; such codes are called *instantaneous*. A special class of instantaneous codes is known as the class of prefix codes: a code is said to have the *prefix property*, and is hence called a *prefix code*, if none of its codewords is a prefix of any other. It is unfortunate that this definition is misleading (shouldn't such a code be rather called a nonprefix code?), but it is widespread and therefore we shall keep it. For example, the code in Fig. 1(a) is not prefix because the codeword for A (0) is a prefix of the codeword for B (01). Similarly, the code in (b) is not prefix, since all the codewords start with 11, which is the codeword for A. On the other hand, codes (c) and (d) are prefix.

It is easy to see that any prefix code is instantaneous and therefore UD. Suppose that while scanning the encoded string for decoding, a codeword x has been detected. In that case, there is no ambiguity as in the example above for code (b), because if there were another possible interpretation y that can be detected later, it would imply that x is a prefix of y, contradicting the prefix property.

In our search for good codes, we shall henceforth concentrate on prefix codes. In fact, we incur no loss by this restriction, even though the set of prefix codes is a proper subset of the UD codes: it can be shown that given any UD code whose codeword lengths are $\{\ell_1, \ldots, \ell_n\}$, one can construct a prefix code with the same set of codeword lengths [11]. As example, note that the prefix code (c) has the same codeword lengths as code (b). In this special case, (c)'s codewords are obtained from those of code (b) by reversing the strings; now every codeword terminates in 11, and the substring 11 occurs only as suffix of any codeword. Thus no codeword can be the proper prefix of any other. Incidently, this also shows that code (b), which is not prefix, is nevertheless UD.

There is a natural one-to-one correspondence between binary prefix codes and binary trees. Let us assign labels to the edges and vertices of a binary tree in the following way:

- every edge pointing to a left child is assigned the label 0, and every edge pointing to a right child is assigned the label 1;
- the root of the tree is assigned the empty string Λ;
- every vertex v of the tree below the root is assigned a binary string, which is obtained by concatenating the labels on the edges of the path leading from the root to vertex v.

It follows from the construction that the string associated with vertex v is a prefix of the string associated with vertex w if and only if v is a vertex on the path from the root to w. Thus, the set of strings associated with the *leaves* of any binary tree satisfies the prefix property and may be considered as a prefix code. Conversely, given any prefix code, one can easily construct the corresponding

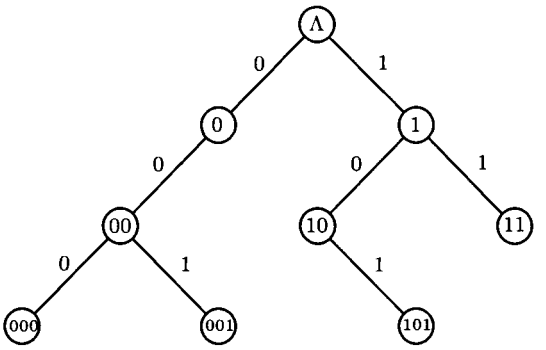


FIGURE 2 Tree corresponding to code {11, 101, 001, 000}.

binary tree. For example, the tree corresponding to the code {11, 101, 001, 000} is depicted in Fig. 2.

The tree corresponding to a code is a convenient tool for decompression. One starts with a pointer to the root and another one to the encoded string, which acts as a guide for the traversal of the tree. While scanning the encoded string from left to right, the tree-pointer is updated to point to the left, respectively right, child of the current node, if the next bit of the encoded string is a 0, respectively a 1. If a leaf of the tree is reached, a codeword has been detected, it is sent to the output, and the tree-pointer is reset to point to the root.

Note that not all the vertices of the tree in Fig. 2 have two children. From the compression point of view, this is a waste, because we could, in that case, replace certain codewords by shorter ones, without violating the prefix property, i.e., build another UD code with a strictly smaller average codeword length. For example, the node labeled 10 has only a right child, so the codeword 101 could be replaced by 10; similarly, the vertex labeled 0 has only a left child, so the codewords 000 and 001 could be replaced by 00 and 01, respectively. A tree for which all internal vertices have two children is called a *complete* tree, and accordingly, the corresponding code is called a complete code. A code is complete if and only if the lengths $\{\ell_i\}$ of its codewords satisfy Eq. (2) with equality, i.e., $\sum_{i=1}^{n} 2^{-\ell_i} = 1$.

A. Huffman Coding

To summarize what we have seen so far, we have restricted the class of codes under consideration in several steps. Starting from general UD codes, we have passed to instantaneous and prefix codes and finally to complete prefix codes, since we are interested in good compression performance. The general problem can thus be stated as follows: we are given a set of n nonnegative weights $\{w_1, \ldots, w_n\}$, which are the frequencies of occurrence of the letters of some alphabet. The problem is to generate a complete binary variable-length prefix code, consisting of codewords with lengths ℓ_i bits, $1 \leq i \leq n$, with optimal compression capabilities, i.e., such that the *total length* of the encoded text

$$\sum_{i=1}^{n} w_i \ell_i \tag{3}$$

is minimized. It is sometimes convenient to redefine the problem in terms of relative frequencies. Let $W = \sum_{i=1}^{n} w_i$ be the total number of characters in the text; one can then define $p_i = w_i / W$ as the *probability of occurrence* of the ith letter. The problem is then equivalent to minimizing the *average codeword length* (ACL) $\sum_{i=1}^{n} p_i \ell_i$.

Let us for a moment forget about the interpretation of the ℓ_i as codeword lengths, and try to solve the minimization problem analytically without restricting the ℓ_i to be integers, but still keeping the constraint that they must satisfy the McMillan equality $\sum_{i=1}^{n} 2^{-\ell_i} = 1$. To find the set of ℓ_i's minimizing (3), one can use Langrange multipliers. Define a function $L(\ell_1, \ldots, \ell_n)$ of n variables and with a parameter λ by

$$L(\ell_1, \ldots, \ell_n) = \sum_{i=1}^{n} w_i \ell_i - \lambda \left(\sum_{i=1}^{n} 2^{-\ell_i} - 1 \right),$$

and look for local extrema by setting the partial derivatives to 0,

$$\frac{\partial L}{\partial \ell_i} = w_i + \lambda 2^{-\ell_i} \ln 2 = 0,$$

which yields

$$\ell_i = -\log_2 \left(\frac{w_i}{-\lambda \ln 2} \right). \tag{4}$$

To find the constant λ, substitute the values for ℓ_i derived in (4) in the McMillan equality

$$1 = \sum_{i=1}^{n} 2^{-\ell_i} = \frac{1}{-\lambda \ln 2} \sum_{i=1}^{n} w_i = \frac{W}{-\lambda \ln 2},$$

from which one can derive $\lambda = -W/\ln 2$. Plugging this value back into (4), one finally gets

$$\ell_i = -\log_2 \left(\frac{w_i}{W} \right) = -\log_2 p_i.$$

This quantity is known as *the information content* of a symbol with probability p_i, and it represents the minimal number of bits in which the symbol could be coded. Note that this number is not necessarily an integer. Returning to the sum in (3), we may therefore conclude that the lower limit of the total size of the encoded file is given by

$$-\sum_{i=1}^{n} w_i \log_2 p_i = W \left(-\sum_{i=1}^{n} p_i \log_2 p_i \right). \tag{5}$$

The quantity $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ has been defined by Shannon [12] as the *entropy* of the probability distribution $\{p_1, \ldots, p_n\}$, and it gives a lower bound to the weighted average codeword length.

In 1952, Huffman [13] proposed the following algorithm which solves the problem:

1. If $n = 1$, the codeword corresponding to the only weight is the null-string; return.
2. Let w_1 and w_2, without loss of generality, be the two smallest weights.

3. Solve the problem recursively for the $n-1$ weights $w_1+w_2, w_3, \ldots, w_n$; let α be the codeword assigned to the weight w_1+w_2.
4. The code for the n weights is obtained from the code for $n-1$ weights generated in point 3 by replacing α by the two codewords $\alpha 0$ and $\alpha 1$; return.

In the straightforward implementation, the weights are first sorted and then every weight obtained by combining the two that are currently the smallest is inserted in its proper place in the sequence so as to maintain order. This yields an $O(n^2)$ time complexity. One can reduce the time complexity to $O(n \log n)$ by using two queues, one, Q_1, containing the original elements, the other, Q_2, the newly created combined elements. At each step, the two smallest elements in $Q_1 \cup Q_2$ are combined and the resulting new element is inserted at the end of Q_2, which remains in order [14].

THEOREM. *Huffman's algorithm yields an optimal code.*

Proof. By induction on the number of elements n. For $n=2$, there is only one complete binary prefix code, which therefore is optimal, namely {0,1}; this is also a Huffman code, regardless of the weights w_1 and w_2.

Assume the truth of the theorem for $n-1$. Let T_1 be an optimal tree for $\{w_1, \ldots, w_n\}$, with ACL $M_1 = \sum_{i=1}^{n} w_i l_i$.

CLAIM 1. *There are at least two elements on the lowest level of* T_1.

Proof. Suppose there is only one such element and let $\gamma = a_1 \cdots a_m$ be the corresponding binary codeword. Then by replacing γ with $a_1 \cdots a_{m-1}$ (i.e., dropping the last bit) the resulting code would still be prefix, and the ACL would be smaller, in contradiction to T_1's optimality. ■

CLAIM 2. *The codewords c_1 and c_2 corresponding to the smallest weights w_1 and w_2 have maximal length (the nodes are on the lowest level in T_1).*

Proof. Suppose the element with weight w_2 is on level m, which is not the lowest level ℓ. Then there is an element with weight $w_x > w_2$ at level ℓ. Thus the tree obtained by switching w_x with w_2 has an ACL of

$$M_1 - w_x\ell - w_2 m + w_x m + w_2 \ell < M_1,$$

which is impossible since T_1 is optimal. ■

CLAIM 3. *Without loss of generality one can assume that the smallest weights w_1 and w_2 correspond to sibling nodes in T_1.*

Proof. Otherwise one could switch elements without changing the ACL. ■

Consider the tree T_2 obtained from T_1 by replacing the sibling nodes corresponding to w_1 and w_2 by their common parent node α, to which the weight w_1+w_2 is assigned. Thus the ACL for T_2 is $M_2 = M_1 - (w_1+w_2)$.

CLAIM 4. *T_2 is optimal for the weights $(w_1+w_2), w_3, \ldots, w_n$.*

Proof. If not, let T_3 be a better tree with $M_3 < M_2$. Let β be the node in T_3 corresponding to (w_1+w_2). Consider the tree T_4 obtained from T_3 by splitting

β and assigning the weight w_1 to β's left child and w_2 to its right child. Then the ACL for T_4 is

$$M_4 = M_3 + (w_1 + w_2) < M_2 + (w_1 + w_2) = M_1,$$

which is impossible, since T_4 is a tree for n elements with weights $w_1, \ldots, w_n$ and T_1 is optimal among all those trees.

Using the inductive assumption, T_2, which is an optimal tree for $n-1$ elements, has the same ACL as the Huffman tree for these weights. However, the Huffman tree for $w_1, \ldots, w_n$ is obtained from the Huffman tree for $(w_1 + w_2)$, $w_3, \cdots, w_n$ in the same way as T_1 is obtained from T_2. Thus the Huffman tree for the n elements has the same ACL as T_1; hence it is optimal. ■

B. Huffman Coding without Bit Manipulations

In many applications, compression is by far not as frequent as decompression. In particular, in the context of static IR systems, compression is done only once (when building the database), whereas decompression directly affects the response time for on-line queries. We are thus more concerned with a good *decoding* procedure. Despite their optimality, Huffman codes are not always popular with programmers as they require bit manipulations and are thus not suitable for smooth programming and efficient implementation in most high-level languages.

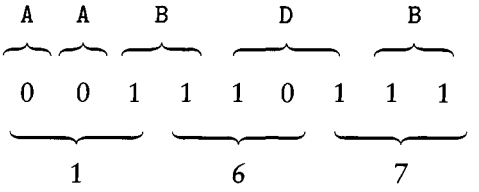
This section presents decoding routines that directly process only bit blocks of fixed and convenient size (typically, but not necessarily, integral bytes), making it therefore faster and better adapted to high-level languages programming, while still being efficient in terms of space requirements. In principle, byte decoding can be achieved either by using specially built tables to isolate each bit of the input into a corresponding byte or by extracting the required bits while simulating shift operations.

1. Eliminating the Reference to Bits

We are given an alphabet Σ, the elements of which are called *letters*, and a message ($\equiv$ sequence of elements of Σ) to be compressed, using variable-length codes. Let L denote the set of N items to be encoded. Often $L = \Sigma$, but we do not restrict the codewords necessarily to represent single letters of Σ. Indeed, the elements of L can be pairs, triplets, or any n-grams of letters, they can represent words of a natural language, and they can finally form a set of items of completely different nature, provided that there is an unambiguous way to decompose a given file into these items (see, for example, [15]). We call L an *alphabet* and its elements *characters*, where these terms should be understood in a broad sense. We thus include also in our discussion applications where N, the size of the alphabet, can be fairly large.

We begin by compressing L using the variable-length Huffman codewords of its different characters, as computed by the conventional Huffman algorithm. We now partition the resulting bit string into k-bit blocks, where k is chosen so as to make the processing of k-bit blocks, with the particular machine and high-level language at hand, easy and natural. Clearly, the boundaries of these blocks do not necessarily coincide with those of the codewords: a k-bit block

may contain several codewords, and a codeword may be split into two (or more) adjacent k-bit blocks. As an example, let $L = \{\texttt{A},\texttt{B},\texttt{C},\texttt{D}\}$, with codewords $\{0,11,100,101\}$ respectively, and choose $k = 3$. Consider the following input string, its coding and the coding's partition into 3-bit blocks:



The last line gives the integer value $0 \le i < 2^3$ of the block.

The basic idea for all the methods is to use these k-bit blocks, which can be regarded as the binary representation of integers, as *indices* to some tables prepared in advance in the preprocessing stage.

In this section we first describe two straightforward—albeit not very efficient—methods for implementing this idea.

For the first method, we use a table $\mathcal{B}$ of 2^k rows and k columns. In fact, $\mathcal{B}$ will contain only zeros and ones, but as we want to avoid bit manipulations, we shall use one byte for each of the $k2^k$ elements of this matrix. Let $i = I_1 \cdots I_k$ be the binary representation of length k (with leading zeros) of i, for $0 \le i < 2^k$, then $\mathcal{B}(i, j) = I_j$, for $1 \le j \le k$; in other words, the ith line of $\mathcal{B}$ contains the binary representation of i, one bit per byte. The matrix $\mathcal{B}$ will be used to decompose the input string into individual bits, without any bit manipulation. Figure 3(a) depicts the matrix $\mathcal{B}$ for $k = 3$.

The value 0 or 1 extracted from $\mathcal{B}$ is used to decode the input, using the Huffman tree of the given alphabet. The Huffman tree of the alphabet L of our small example is in Fig. 4(a).

A Huffman tree with N leaves (and $N-1$ internal nodes) can be kept as a table $\mathcal{H}$ with $N-1$ rows (one for each internal node) and two columns. The internal nodes are numbered from 0 to $N-2$ in arbitrary order, but for convenience the root will always be numbered zero. For example, in Fig. 4(a),

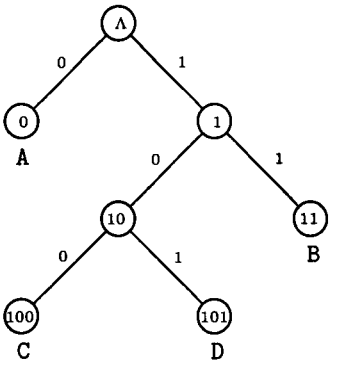
$\mathcal{B}$	1	2	3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

(a)

$\mathcal{S}$	1	2
0	0	0
1	2	0
2	4	0
3	6	0
4	0	1
5	2	1
6	4	1
7	6	1

(b)

FIGURE 3 Tables for Huffman decoding.



(a)

$\mathcal{H}$	0	1
0	-A	1
1	2	-B
2	-C	-D

(b)

FIGURE 4 Example of Huffman code. (a) Tree form and (b) table form.

the indices of the internal nodes containing Λ, 1, and 10 will be 0, 1 and 2 respectively. The two elements stored in the ith row of Table $\mathcal{H}$ are the left and right children of the internal node indexed i. Each child can be either another internal node, in which case its index is stored, or a leaf, corresponding to one of the characters of the alphabet, in which case this character is stored. We thus need an additional bit per element, indicating whether it is an internal node or a leaf, but generally, one can use the sign bit for that purpose: if the element is positive, it represents the index of an internal node; if it is negative, its absolute value is the representation of a character. Figure 4(b) shows Table $\mathcal{H}$ corresponding to the Huffman tree of Fig. 4(a). The Huffman decoding routine can then be formulated as follows:

Byte Decoding algorithm

```
ind ← 0        [pointer to table H]
repeat
    n ← integer value of next input block
    for j = 1 to k
        newind ← H(ind, B(n, j))      [left or right child of current node]
        if newind > 0 then ind ← newind
        else
            output(−newind)
            ind ← 0
    end
until input is exhausted
```

Another possibility is to replace table $\mathcal{B}$ by the following table $\mathcal{S}$, again with 2^k rows, but only two columns. For $0 \leq i < 2^k$, $\mathcal{S}(i, 1)$ will contain $2i \bmod 2^k$, and $\mathcal{S}(i, 2)$ will contain the leftmost bit of the k-bit binary representation of i. In the algorithm, the assignment to *newind* must be replaced by

$$newind \leftarrow H(ind, S(n, 2))$$
$$n \leftarrow S(n, 1).$$

The first statement extracts the leftmost bit and the second statement shifts the k-bit block by one bit to the left. Figure 3(b) shows Table $\mathcal{S}$ for $k = 3$. Hence we have reduced the space needed for the tables from $k2^k + 2(N-1)$ to $2^{k+1} + 2(N-1)$, but now there are three table accesses for every bit of the input, instead of only two accesses for the first method.

Although there is no reference to bits in these algorithms and their programming is straightforward, the number of table accesses makes their efficiency rather doubtful; their only advantage is that their space requirements are linear in N (k is a constant), while for all other time-efficient variants to be presented below, space is at least $\Omega(N \log N)$. However, for these first two methods, the term 2^k of the space complexity is dominant for small N, so that they can be justified—if at all—only for rather large N.

2. Partial-Decoding Tables

Recall that our goal is to permit a block-per-block processing of the input string for some fixed block-size k. Efficient decoding under these conditions is made possible by using a set of m auxiliary tables, which are prepared in advance for every given Huffman code, whereas Tables $\mathcal{B}$ and $\mathcal{S}$ above were independent of the character distribution.

The number of entries in each table is 2^k, corresponding to the 2^k possible values of the k-bit patterns. Each entry is of the form (W, j), where W is a sequence of characters and j $(0 \le j < m)$ is the index of the next table to be used. The idea is that entry i, $0 \le i < 2^k$, of Table 0 contains, first, the longest possible decoded sequence W of characters from the k-bit block representing the integer i (W may be empty when there are codewords of more than k bits). Usually some of the last bits of the block will not be decipherable, being the prefix P of more than one codeword; j will then be the index of the table corresponding to that prefix (if $P = \Lambda$, then $j = 0$). Table j is constructed in a similar way except for the fact that entry i will contain the analysis of the bit pattern formed by the prefixing of P to the binary representation of i. We thus need a table for every possible proper prefix of the given codewords; the number of these prefixes is obviously equal to the number of internal nodes of the appropriate Huffman tree (the root corresponding to the empty string and the leaves corresponding to the codewords), so that $m = N - 1$.

More formally, let P_j, $0 \le j < N-1$, be an enumeration of all the proper prefixes of the codewords (no special relationship needs to exist between j and P_j, except for the fact that $P_0 = \Lambda$). In Table j corresponding to P_j, the ith entry, $T(j, i)$, is defined as follows: let B be the bit string composed of the juxtaposition of P_j to the left of the k-bit binary representation of i. Let W be the (possibly empty) longest sequence of characters that can be decoded from B, and P_ℓ the remaining undecipherable bits of B; then $T(j, i) = (W, \ell)$.

Referring again to the simple example given above, there are three possible proper prefixes: Λ,1,10, hence three corresponding tables indexed 0,1,2 respectively, and these are given in Fig. 5. The column headed "Pattern" contains for every entry the binary string decoded in Table 0; the binary strings decoded by Tables 1 and 2 are obtained by prefixing "1," respectively "10," to the strings in "Pattern."

Entry	Pattern for Table 0	Table 0		Table 1		Table 2	
		W	ℓ	W	ℓ	W	ℓ
0	000	AAA	0	CA	0	CAA	0
1	001	AA	1	C	1	CA	1
2	010	A	2	DA	0	C	2
3	011	AB	0	D	1	CB	0
4	100	C	0	BAA	0	DAA	0
5	101	D	0	BA	1	DA	1
6	110	BA	0	B	2	D	2
7	111	B	1	BB	0	DB	0

FIGURE 5 Partial-decoding tables.

For the input example given above, we first access Table 0 at entry 1, which yields the output string AA, Table 1 is then used with entry 6, giving the output B, and finally Table 2 at entry 7 gives output DB.

The utterly simple decoding subroutine (for the general case) is as follows ($M(i)$ denotes the ith block of the input stream, j is the index of the table currently being used, and $T(j, \ell)$ is the ℓth entry of table j):

Basic Decoding Algorithm

$j \leftarrow 0$
for $i \leftarrow 1$ **to** length of input **do**
    (output, j) $\leftarrow T(j, M(i))$
end

As mentioned before, the choice of k is largely governed by the machine-word structure and the high-level language architecture. A natural choice in most cases would be $k = 8$, corresponding to a byte context, but $k = 4$ (half-byte) or $k = 16$ (half-word) are also conceivable. The larger k is, the greater the number of characters that can be decoded in a single iteration, thus transferring a substantial part of the decoding time to the preprocessing stage. The size of the tables, however, grows exponentially with k, and with every entry occupying (for $N \leq 256$ and $k = 8$) 1 to 8 bytes, each table may require between 1 and 2 Kbytes of internal memory. For $N > 256$, we need more than one byte for the representation of a character, so that the size of a table will be even larger, and for larger alphabets these storage requirements may become prohibitive. We now develop an approach that can help reduce the number of required tables and their size.

3. Reducing the Number of Tables: Binary Forests

The storage space needed by the partial-decoding tables can be reduced by relaxing somewhat the approach of the previous section, and using the

conventional Huffman decoding algorithm no more than once for every block, while still processing only k-bit blocks. This is done by redefining the tables and adding some new data structures.

Let us suppose, just for a moment, that after deciphering a given block B of the input that contains a "remainder" P (which is a prefix of a certain codeword), we are somehow able to determine the correct complement of P and its length ℓ, and accordingly its corresponding encoded character. More precisely, since a codeword can extend into more than two blocks, ℓ will be the length of the complement of P in the next k-bit block which contains also other codewords; hence $0 \leq \ell < k$. In the next iteration (decoding of the next k-bit block not yet entirely deciphered), table number ℓ will be used, which is similar to Table 0, but *ignores* the first ℓ bits of the corresponding entry, instead of *prefixing* P to this entry as in the previous section.

Therefore the number of tables reduces from $N-1$ (about 30 in a typical single-letter natural-language case, or 700–900 if we use letter pairs) to only k (8 or 16 in a typical byte or half-word context), where entry i in Table ℓ, $0 \leq \ell < k$, contains the decoding of the $k-\ell$ rightmost bits of the binary representation of i. It is clear, however, that Table 1 contains two exactly equal halves, and in general table ℓ ($0 \leq \ell < k$) consists of 2^ℓ identical parts. Retaining then in each table only the first $2^{k-\ell}$ entries, we are able to compress the needed k tables into the size of only two tables. The entries of the tables are again of the form (W, j); note however that j is not an index to the next table, but an identifier of the remainder P. It is only after finding the correct complement of P and its length ℓ we can access the right table ℓ.

For the same example as before one obtains the tables of Fig. 6, where table t decodes the bit strings given in "Pattern," but ignoring the t leftmost bits, $t = 0$, 1, 2, and $l = 0, 1, 2$ corresponds respectively to the proper prefixes Λ, 1, 10.

The algorithm will be completed if we can find a method for identifying the codeword corresponding to the remainder of a given input block, using of course the following input block(s). We introduce the method through an example.

Entry	Pattern	Table 0		Table 1		Table 2	
	for Table 0	W	ℓ	W	ℓ	W	ℓ
0	000	AAA	0	AA	0	A	0
1	001	AA	1	A	1	–	1
2	010	A	2	–	2		
3	011	AB	0	B	0		
4	100	C	0				
5	101	D	0				
6	110	BA	0				
7	111	B	1				

FIGURE 6 Substring Translate Tables.

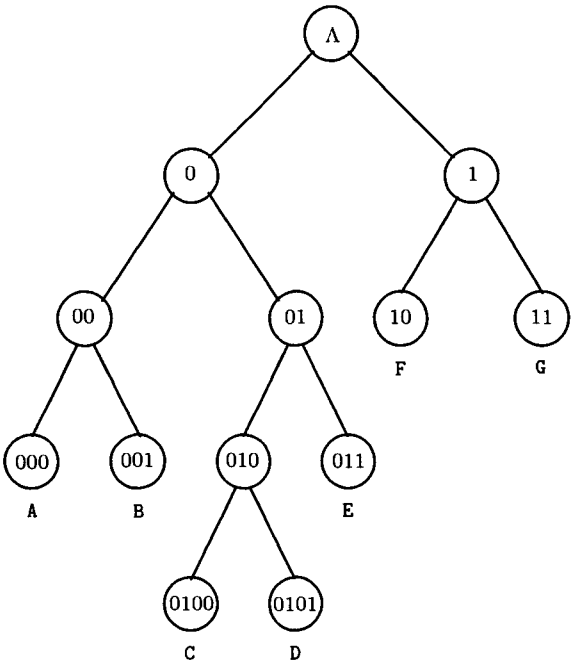


FIGURE 7 The Huffman tree H.

Figure 7 shows a typical Huffman tree H for an alphabet L of $N = 7$ characters. Assume now $k = 8$ and consider the following adjacent blocks of input: 00101101 00101101. The first block is decoded into the string BE and the remainder $P = 01$. Starting at the internal node containing 01 and following the first bits of the following block, we get the codeword C, and length $l = 2$ for the complement of P, so that Table 2 will be used when decoding the next block; ignoring the first 2 bits, this table translates the binary string 101101.

For the general case, let us for simplicity first assume that the depth of H, which is the length of the longest codeword, is bounded by k. Given the nonempty remainder P of the current input block, we must access the internal node corresponding to P, and proceed downwards turning left (0) or right (1) as indicated by the first few bits of the next k-bit block, until we reach a leaf. This leaf contains the next character of the output. The number of edges traversed is the index of the table to be used in the next iteration.

Our goal is to simulate this procedure without having to follow a "bit-traversal" of the tree. The algorithm below uses a binary forest instead of the original Huffman tree H. For the sake of clarity, the construction of the forest is described in two steps.

First, replace H by $N - 2$ smaller trees H_i, which are induced by the proper subtrees rooted at the internal nodes of H, corresponding to all nonempty proper prefixes of the codewords. The nodes of the trees of the forest contain binary strings: Λ for the roots, and for each other node v, a string obtained by concatenating the labels of the edges on the path from the root to v, as in the Huffman tree, but padded at the right by zeroes so as to fill a k-bit block. In addition, each leaf contains also the corresponding decoded character. The

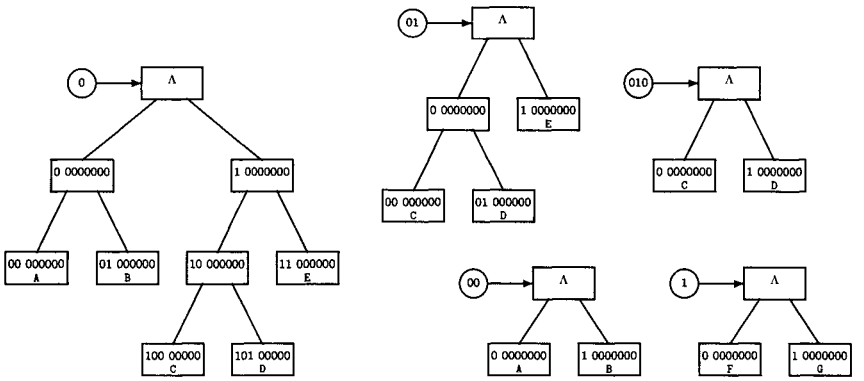


FIGURE 8 Forest of proper prefixes.

string in node v is denoted by VAL(v). Figure 8 depicts the forest obtained from the tree of our example, where the pointer to each tree is symbolized by the corresponding proper prefix. The idea is that the identifier of the remainder in an entry of the tables described above is in fact a pointer to the corresponding tree. The traversal of this tree is guided by the bits of the next k-bit block of the input, which can directly be compared with the contents of the nodes of the tree, as will be described below.

Consider now also the possibility of long codewords, which extend over several blocks. They correspond to long paths so that the depth of some trees in the forest may exceed k. During the traversal of a tree, passing from one level to the next lowest one is equivalent to advancing one bit in the input string. Hence when the depth exceeds k, all the bits of the current k-bit block were used, and we pass to the next block. Therefore the above definition of VAL(v) applies only to nodes on levels up to k; this definition is generalized to any node by the following: VAL(v) for a node v on level j, with $ik < j \leq (i+1)k$, $i \geq 0$, is the concatenation of the labels on the edges on the path from level ik to v.

In the second step, we compress the forest as could have been done with any Huffman tree. In such trees, every node has degree 0 or 2; i.e., they appear in pairs of siblings (except the root). For a pair of sibling nodes (a, b), VAL(a) and VAL(b) differ only in the jth bit, where j is the level of the pair (here and in what follows, the level of the root of a tree is 0), or more precisely, $j = (\text{level} - 1) \bmod k + 1$. In the compressed tree, every pair is represented by a unique node containing the VAL of the right node of the pair, the new root is the node obtained from the only pair in level 1, and the tree structure is induced by the noncompressed tree. Thus a tree of ℓ nodes shrinks now to $(\ell - 1)/2$ nodes. Another way to look at this "compression" method is to take the tree of internal nodes, and store it in the form of a table as described in the previous section. We use here a tree-oriented vocabulary, but each tree can equivalently be implemented as a table. Figure 9 is the compressed form of the forest of Fig. 8.

We can now compare directly the values VAL stored in the nodes of the trees with the k-bit blocks of the Huffman encoded string. The VAL values have the following property: let v be a node on level j of one of the trees in the compressed forest, with $ik < j \leq (i+1)k$, $i \geq 0$ as above, and let $I(B)$ be the

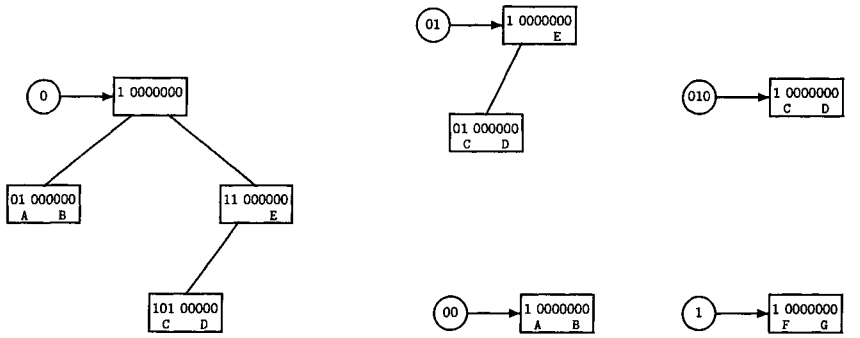


FIGURE 9 Compressed forest.

integer value of the next k-bit block B. Then

$$I(B) < \text{VAL}(v) \quad \text{if and only if} \quad \text{bit}(1 + j \bmod k) \text{ of } B \text{ is } 0.$$

Thus after accessing one of the trees, the VAL of its root is compared with the next k-bit block B. If B, interpreted as a binary integer, is smaller, it must start with 0 and we turn left; if B is greater or equal, it must start with 1 and we turn right. These comparisons are repeated at the next levels, simulating the search for an element in a binary search tree [16, Section 6.2.2]. This leads to the modified algorithm below. Notations are like before, where ROOT(t) points to the tth tree of the forest, and every node has three fields: VAL, a k-bit value, and LEFT and RIGHT, each of which is either a pointer to the next level or contains a character of the alphabet. When accessing Table j, the index is taken modulo the size of the table, which is 2^{k-j}.

Revised Decoding Algorithm

```
i ← 1
j ← 0
repeat
      (output, tree-nbr) ← T(j, S(i) mod 2^(k−j))
      i ← i + 1
      j ← 0
      if tree-nbr ≠ 0 then TRAVERSE ( ROOT(tree-nbr) )
until input is exhausted
```

where the procedure TRAVERSE is defined by

```
TRAVERSE (node)
    repeat
        if S(i) < VAL(node) then
                node ← LEFT(node)
        else    node ← RIGHT(node)
        if node is a character C then output C
        j ← j + 1     [j is the number of bits in S(i) which are 'used up']
        if j = k then
            j ← 0
            i ← i + 1     [advance to next k-bit block]
    until a character was output
end
```

Any node v of the original (compressed) Huffman tree H' generates several nodes in the forest, the number of which is equal to the level of v in H'. Hence the total number of nodes in the forest is exactly the *internal path length* of the original (uncompressed) Huffman tree H, as defined by Knuth [17]. This quantity is between $O(N \log N)$ (for a full binary tree) and $O(N^2)$ (for a degenerate tree), and at the average, with all possible shapes of Huffman trees equally likely, proportional to $N\sqrt{N}$.

Therefore even in the worst case, the space requirements are reasonable in most practical applications with small N. If, for large N and certain probability distributions, $O(N^2)$ is prohibitive, it is possible to keep the space of the forest bounded by $O(N \log N)$, if one agrees to abandon the optimality of the Huffman tree. This can be done by imposing a maximal length of $K = O(\log N)$ to the codewords. If K does not exceed the block-size k, the decoding algorithm can even be slightly simplified, since in the procedure TRAVERSE there is no need to check whether the end of the block was reached. Another advantage of bounding the depth of the Huffman tree is that this tends to lengthen the shortest codeword. Since the number of characters stored at each entry in the partial-decoding tables is up to $1 + \lceil (k-1)/s \rceil$, where s is the length of the shortest codeword, this can reduce the space required to store each table. An algorithm for the construction of an optimal tree with bounded depth in time and space $O(KN)$ can be found in [18]. Nevertheless, it might often not seem worthwhile to spend so much efforts obtaining an *optimal* code of bounded length. As an alternative one can use a procedure proposed in [19], which gives a suboptimal average codeword length, but uses less space and is much faster. Moreover, the codes constructed by this method are often very near to optimal.

4. Huffman Codes with Radix $r > 2$

The number of tables can also be reduced by the following simple variants which, similar to the variants with bounded codeword length, yield compression factors slightly lower than those of the methods described above. Let us apply the Huffman algorithm with radix r, $r > 2$, the details of which can be found in Huffman's original paper [13]. In such a variant, one combines at each step, except perhaps the first, the r smallest weights (rather than only the smallest two in the binary algorithm) and replaces them with their sum. The number of weights combined in the first step is chosen so that the number h of weights remaining after this step verifies $h \equiv 1 (\mathrm{mod}\ r-1)$. In the corresponding r-ary tree, every internal node has r sons, except perhaps one on the next-to-lowest level of the tree, which has between 2 and r sons. If we choose $r = 2^\ell$, we can encode the alphabet in a first stage using r different symbols; then every symbol is replaced by a binary code of ℓ bits. If in addition ℓ divides k, the "borders" of the k-bit blocks never split any ℓ-bit code. Hence in the partial-decoding tables, the possible remainders are sequences of one or more r-ary symbols. There is therefore again a correspondence between the possible remainders and the internal nodes of the r-ary Huffman tree, only that their number now decreased to $\lceil (n-1)/(r-1) \rceil$. Moreover, there may be some savings in the space needed for a specific table. As we saw before, the space for each table depends on the length s of the shortest codeword, so this can be k with the binary algorithm when $s = 1$, but at most $\lceil k/2 \rceil$ in the 4-ary case.

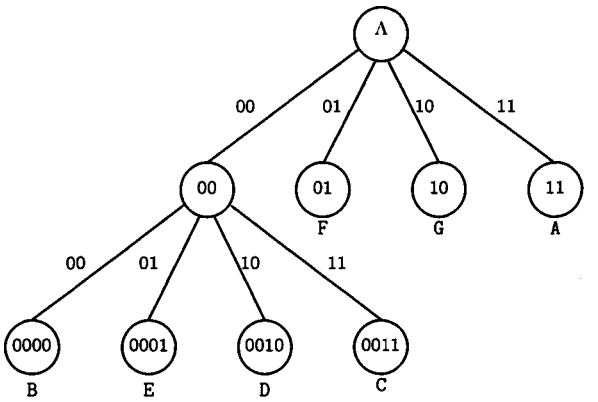


FIGURE 10 Quaternary Huffman tree.

Due to the restrictions on the choice of r, there are only few possible values. For example, for $k = 8$, one could use a quaternary code ($r = 2^2$), where every codeword has an even number of bits and the number of tables is reduced by a factor of 3, or a hexadecimal code ($r = 2^4$), where the codeword length is a multiple of 4 and the number of tables is divided by 15. Note that for alphabets with $N \leq 31$, the hexadecimal code can be viewed as the classical method using "restricted variability" (see, for example, [20]): assign 4-bit encodings to the 15 most frequent characters and use the last 4-bit pattern as "escape character" to indicate that the actual character is encoded in the next 4 bits. Thus up to 16 least frequent characters have 8-bit encodings, all of which have their first 4 bits equal to the escape character.

Referring to the Huffman tree given in Fig. 7, suppose that a character corresponding to a leaf on level ℓ appears with probability $2^{-\ell}$, then the corresponding 2^2-ary tree is given in Fig. 10. Note that the only proper prefixes of even length are Λ and 00, so that the number of tables dropped from 6 to 2.

However, with increasing r, compression will get worse, so that the right tradeoff must be chosen according to the desired application.

C. Space-Efficient Decoding of Huffman Codes

The data structures needed for the decoding of a Huffman encoded file (a Huffman tree or lookup table) are generally considered negligible overhead relative to large texts. However, not all texts are large, and if Huffman coding is applied in connection with a Markov model [21], the required Huffman forest may become itself a storage problem. Moreover, the "alphabet" to be encoded is not necessarily small, and may, e.g., consist of all the different words in the text, so that Huffman trees with thousands and even millions of nodes are not uncommon [22]. We try here to reduce the necessary internal memory space by devising efficient ways of encoding these trees. In addition, the suggested data structure also allows a speed-up of the decompression process, by reducing the number of necessary bit comparisons.

1. Canonical Huffman Codes

For a given probability distribution, there might be quite a large number of different Huffman trees, since interchanging the left and right subtrees of any internal node will result in a different tree whenever the two subtrees are different in structure, but the weighted average path length is not affected by such an interchange. There are often also other optimal trees, which cannot be obtained via Huffman's algorithm. One may thus choose one of the trees that has some additional properties. The preferred choice for many applications is the *canonical* tree, defined by Schwartz and Kallick [23], and recommended by many others (see, e.g., [24,25]).

Denote by $(p_1, \ldots, p_n)$ the given probability distribution, where we assume that $p_1 \geq p_2 \geq \cdots \geq p_n$, and let ℓ_i be the length in bits of the codeword assigned by Huffman's procedure to the element with probability p_i; i.e., ℓ_i is the depth of the leaf corresponding to p_i in the Huffman tree. A tree is called canonical if, when scanning its leaves from left to right, they appear in nondecreasing order of their depth (or equivalently, in nonincreasing order, as in [26]). The idea is that Huffman's algorithm is only used to generate the lengths $\{\ell_i\}$ of the codewords, rather than the codewords themselves; the latter are easily obtained as follows: the ith codeword consists of the first ℓ_i bits immediately to the right of the "binary point" in the infinite binary expansion of $\sum_{j=1}^{i-1} 2^{-\ell_j}$, for $i = 1, \ldots, n$ [27]. Many properties of canonical codes are mentioned in [24,28].

The following will be used as a running example in this section. Consider the probability distribution implied by Zipf's law, defined by the weights $p_i = 1/(i H_n)$, for $1 \leq i \leq n$, where $H_n = \sum_{j=1}^{n} (1/j)$ is the nth harmonic number. This law is believed to govern the distribution of the most common words in a large natural language text [29]. A canonical code can be represented by the string $\langle n_1, n_2, \ldots, n_k \rangle$, called a *source,* where k denotes, here and below, the length of the longest codeword (the depth of the tree), and n_i is the number of codewords of length i, $i = 1, \ldots, k$. The source corresponding to Zipf's distribution for $n = 200$ is $\langle 0, 0, 1, 3, 4, 8, 15, 32, 63, 74 \rangle$. The code is depicted in Fig. 11.

We shall assume, for the ease of description in the current section, that the source has no "holes"; i.e., there are no three integers $i < j < \ell$ such that $n_i \neq 0, n_\ell \neq 0$, but $n_j = 0$. This is true for many, but not all, real-life distributions.

One of the properties of canonical codes is that the codewords having the same length are the binary representations of consecutive integers. For example, in our case, the codewords of length 9 bits are the binary integers in the range from 110011100 to 111011010. This fact can be exploited to enable efficient decoding with relatively small overhead: once a codeword of ℓ bits is detected, one can get its relative index within the sequence of codewords of length ℓ by simple subtraction.

The following information is thus needed: let $m = \min\{i \mid n_i > 0\}$ be the length of the shortest codeword, and let $base(i)$ be the integer value of the first codeword of length i. We then have

$$base(m) = 0$$

$$base(i) = 2(base(i-1) + n_{i-1}) \qquad \text{for } m < i \leq k.$$

0	000
1	0010
2	0011
3	0100
4	01010
5	01011
6	01100
7	01101
8	011100
9	011101
10	011110
11	011111
12	100000
13	100001
14	100010
15	100011
16	1001000
17	1001001
18	1001010
19	1001011
...	...
29	1010101
30	1010110
31	10101110
32	10101111
33	10110000
...	...
61	11001100
62	11001101
63	110011100
64	110011101
...	...
124	111011001
125	111011010
126	1110110110
127	1110110111
...	...
198	1111111110
199	1111111111

FIGURE 11 Canonical Huffman code for Zipf-200.

Let $B_s(k)$ denote the standard s-bit binary representation of the integer k (with leading zeros, if necessary). Then the jth codeword of length i, for $j = 0, 1, \ldots, n_i - 1$, is $B_i(base(i) + j)$. Let $seq(i)$ be the sequential index of the first codeword of length i:

$$seq(m) = 0$$

$$seq(i) = seq(i-1) + n_{i-1} \qquad \text{for } m < i \leq k.$$

Suppose now that we have detected a codeword w of length ℓ. If $I(w)$ is the integer value of the binary string w (i.e., $w = B_\ell(I(w))$), then $I(w) - base(\ell)$ is the relative index of w within the block of codewords of length ℓ. Thus $seq(\ell) + I(w) - base(\ell)$ is the relative index of w within the full list of codewords. This can be rewritten as $I(w) - diff(\ell)$, for $diff(\ell) = base(\ell) - seq(\ell)$. Thus all one needs is the list of integers $diff(\ell)$. Table 2 gives the values of n_i, $base(i)$, $seq(i)$, and $diff(i)$ for our example.

TABLE 2 Decode Values for Canonical Huffman Code for Zipf-200

i	n_i	$base(i)$	$seq(i)$	$diff(i)$
3	1	0	0	0
4	3	2	1	1
5	4	10	4	6
6	8	28	8	20
7	15	72	16	56
8	32	174	31	143
9	63	412	63	349
10	74	950	126	824

We suggest in the next section a new representation of canonical Huffman codes, which not only is space-efficient, but may also speed up the decoding process, by permitting, at times, the decoding of more than a single bit in one iteration. Similar ideas, based on tables rather than on trees, were recently suggested in [26].

2. Skeleton Trees for Fast Decoding

The following small example, using the data above, shows how such savings are possible. Suppose that while decoding, we detect that the next codeword starts with 1101. This information should be enough to decide that the following codeword ought to be of length 9 bits. We should thus be able, after having detected the first 4 bits of this codeword, to read the following 5 bits as a block, without having to check after each bit if the end of a codeword has been reached. Our goal is to construct an efficient data structure that permits similar decisions as soon as they are possible. The fourth bit was the earliest possible in the above example, since there are also codewords of length 8 starting with 110.

Decoding with sk-trees

The suggested solution is a binary tree, called below an *sk-tree* (for skeleton tree), the structure of which is induced by the underlying Huffman tree, but which has generally significantly fewer nodes. The tree will be traversed like a regular Huffman tree. That is, we start with a pointer to the root of the tree, and another pointer to the first bit of the encoded binary sequence. This sequence is scanned, and after having read a zero (respectively, a 1), we proceed to the left (respectively, right) child of the current node. In a regular Huffman tree, the leaves correspond to full codewords that have been scanned, so the decoding algorithm just outputs the corresponding item, resets the tree pointer to the root and proceeds with scanning the binary string. In our case, however, we visit the tree only up to the depth necessary to identify the length of the current codeword. The leaves of the sk-tree then contain the lengths of the corresponding codewords.

The formal decoding process using an sk-tree is depicted in Fig. 12. The variable *start* points to the index of the bit at the beginning of the current

```
{
    tree_pointer ⟵ root
    i ⟵ 1
    start ⟵ 1
    while i < length_of_string
    {
        if string[i] = 0        tree_pointer ⟵ left(tree_pointer)
        else                    tree_pointer ⟵ right(tree_pointer)
        if value(tree_pointer)> 0
        {
            codeword ⟵ string[start ··· (start + value(tree_pointer) −1)]
            output ⟵ table[ I(codeword)−diff[ value(tree_pointer)] ]
            tree_pointer ⟵ root
            start ⟵ start + value(tree_pointer)
            i ⟵ start
        }
        else                    i ⟵ i+1
    }
}
```

FIGURE 12 Decoding procedure using sk-tree.

codeword in the encoded string, which is stored in the vector *string*[]. Each node of the sk-tree consists of three fields: a *left* and a *right* pointer, which are not null if the node is not a leaf, and a *value* field, which is zero for internal nodes, but contains the length in bits of the current codeword, if the node is a leaf. In an actual implementation, we can use the fact that any internal node has either zero or two sons, and store the *value*-field and the *right*-field in the same space, with *left* = *null* serving as flag for the use of the *right* pointer. The procedure also uses two tables: $table[j]$, $0 \leq j < n$, giving the jth element (in nonincreasing order of frequency) of the encoded alphabet; and $diff[i]$ defined above, for i varying from m to k, that is, from the length of the shortest to the length of the longest codeword.

The procedure passes from one level in the tree to the one below according to the bits of the encoded string. Once a leaf is reached, the next *codeword* can be read in one operation. Note that not all the bits of the input vector are individually scanned, which yields possible time savings.

Figure 13 shows the sk-tree corresponding to Zipf's distribution for $n = 200$. The tree is tilted by 45°, so that left (right) children are indicated by arrows pointing down (to the right). The framed leaves correspond to the last codewords of the indicated length. The sk-tree of our example consists of only 49 nodes, as opposed to 399 nodes of the original Huffman tree.

Construction of sk-trees

While traversing a standard canonical Huffman tree to decode a given codeword, one may stop as soon as one gets to the root of any full subtree of depth h, for $h \geq 1$, i.e., a subtree of depth h that has 2^h leaves, since at this stage it is known that exactly h more bits are needed to complete the codeword. One way to look at sk-trees is therefore as standard Huffman trees from which all

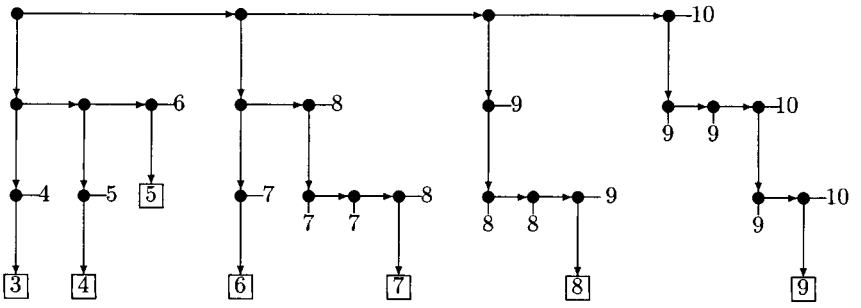


FIGURE 13 Sk-tree for Zipf-200 distribution.

full subtrees of depth $h \geq 1$ have been pruned. A more direct and much more efficient construction is as follows.

The one-to-one correspondence between the codewords and the paths from the root to the leaves in a Huffman tree can be extended to define, for any binary string $S = s_1 \cdots s_e$, the path $P(S)$ induced by it in a tree with given root r_0. This path will consist of $e + 1$ nodes $r_i, 0 \leq i \leq e$, where for $i > 0$, r_i is the left (respectively, right) child of r_{i-1}, if $s_i = 0$ (respectively, if $s_i = 1$). For example, in Fig. 13, $P(111)$ consists of the four nodes represented as bullets in the top line. The skeleton of the sk-tree will consist of the paths corresponding to the last codeword of every length. Let these codewords be denoted by $L_i, m \leq i \leq k$; they are, for our example, 000, 0100, 01101, 100011, etc. The idea is that $P(L_i)$ serves as "demarcation line": any node to the left (respectively, right) of $P(L_i)$, i.e., a left (respectively, right) child of one of the nodes in $P(L_i)$, corresponds to a prefix of a codeword with length $\leq i$ (respectively, $> i$).

As a first approximation, the construction procedure thus takes the tree obtained by $\bigcup_{i=m}^{k-1} P(L_i)$ (there is clearly no need to include the longest codeword L_k, which is always a string of k 1's); and adjoins the missing children to turn it into a complete tree in which each internal node has both a left and a right child. The label on such a new leaf is set equal to the label of the closest leaf following it in an in-order traversal. In other words, when creating the path for L_i, one first follows a few nodes in the already existing tree, then one branches off creating new nodes; as to the labeling, the missing right child of any node in the path will be labeled $i + 1$ (basing ourselves on the assumption that there are no holes), but only the missing left children of any *new* node in the path will be labeled i.

A closer look then implies the following refinement. Suppose a codeword L_i has a zero in its rightmost position, i.e., $L_i = \alpha 0$ for some string α of length $i - 1$. Then the first codeword of length $i + 1$ is $\alpha 10$. It follows that only when getting to the ith bit can one decide if the length of the current codeword is i or $i + 1$. However, if L_i terminates in a string of 1's, $L_i = \beta 01^a$, with $a > 0$ and $|\beta| + a = i - 1$, then the first codeword of length $i + 1$ is $\beta 10^{a+1}$, so the length of the codeword can be deduced already after having read the bit following β. It follows that one does not always need the full string L_i in the sk-tree, but only its prefix up to and not including the rightmost zero. Let $L_i^* = \beta$ denote this prefix. The revised version of the above procedure starts with the tree obtained by $\bigcup_{i=m}^{k-1} P(L_i^*)$. The nodes of this tree are depicted as bullets in Fig. 13. For each path $P(L_i^*)$ there is a leaf in the tree, and the left child of this leaf is the new

terminal node, represented in Fig. 13 by a box containing the number i. The additional leaves are then filled in as explained above.

Space Complexity

To evaluate the size of the sk-tree, we count the number of nodes added by path $P(L_i^*)$, for $m \leq i < k$. Since the codewords in a canonical code, when ordered by their corresponding frequencies, are also alphabetically sorted, it suffices to compare L_i to L_{i-1}. Let $\gamma(m) = 0$, and for $i > m$, let $\gamma(i)$ be the longest common prefix of L_i and L_{i-1}, e.g., $\gamma(7)$ is the string 10 in our example. Then the number of nodes in the sk-tree is given by

$$size = 2\left(\sum_{i=m}^{k-1} \max(0, |L_i^*| - |\gamma(i)|)\right) - 1,$$

since the summation alone is the number of internal nodes (the bullets in Fig. 13).

The maximum function comes to prevent an extreme case in which the difference might be negative. For example, if $L_6 = 010001$ and $L_7 = 0101111$, the longest common prefix is $\gamma(7) = 010$, but since we consider only the bits up to and not including the rightmost zero, we have $L_7^* = 01$. In this case, indeed, no new nodes are added for $P(L_7^*)$.

An immediate bound on the number of nodes in the sk-tree is $O(\min(n, k^2))$, since on the one hand, there are up to $k-1$ paths $P(L_i^*)$ of lengths $\leq k-2$, but on the other hand, it cannot exceed the number of nodes in the underlying Huffman tree, which is $2n-1$. To get a tighter bound, consider the nodes in the upper levels of the sk-tree belonging to the full binary tree F with $k-1$ leaves and having the same root as the sk-tree. The depth of F is $d = \lceil \log_2(k-1) \rceil$, and all its leaves are at level d or $d-1$. The tree F is the part of the sk-tree where some of the paths $P(L_i^*)$ must be overlapping, so we account for the nodes in F and for those below separately. There are at most $2k-1$ nodes in F; there are at most $k-1$ disjoint paths below it, with path $P(L_i^*)$ extending at most $i - 2 - \lfloor \log_2(k-1) \rfloor$ nodes below F, for $\log_2(k-1) < i \leq k$. This yields as bound for the number of nodes in the sk-tree

$$2k + 2\left(\sum_{i=1}^{k-2-\lfloor \log_2(k-1) \rfloor} i\right) = 2k + (k - 2 - \lfloor \log_2(k-1) \rfloor)(k - 1 - \lfloor \log_2(k-1) \rfloor).$$

There are no savings in the worst case, e.g., when there is only one codeword of each length (except for the longest, for which there are always at least two). More generally, if the depth of the Huffman tree is $\Omega(n)$, the savings might not be significant, but such trees are optimal only for some very skewed distributions. In many applications, like for most distributions of characters or character pairs or words in most natural languages, the depth of the Huffman tree is $O(\log n)$, and for large n, even the constant c, if the depth is $c \log_2 n$, must be quite small. For suppose the Huffman tree has a leaf on depth d. Then by [30, Theorem 1], the probability of the element corresponding to this leaf is $p < 1/F_{d+1}$, where F_j is the jth Fibonacci number, and we get from [17, Exercise 1.2.1–4] that $p < (1/\phi)^{d-1}$, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. Thus if $d > c \log_2 n$,

we have

$$p < \left(\frac{1}{\phi}\right)^{c \log_2 n} = n^{-c \log_2(1/\phi)} = n^{-0.693c}.$$

To give a numeric example, a Huffman tree corresponding to the different words in English, as extracted from 500 MB (87 million words) of the *Wall Street Journal* [31], had $n = 289,101$ leaves. The probability for a tree of this size to have a leaf at level $3 \log_2 n$ is less than 4.4×10^{-12}, which means that even if the word with this probability appears only once, the text must be at least 4400 billion words long, enough to fill about 35,000 CD-Roms! However, even if the original Huffman tree would be deeper, it is sometimes convenient to impose an upper limit of $B = O(\log n)$ on the depth, which often implies only a negligible loss in compression efficiency [19]. In any case, given a logarithmic bound on the depth, the size of the sk-tree is about

$$\log n (\log n - \log \log n).$$

D. Arithmetic Coding

We have dealt so far only with Huffman coding, and even shown that they are optimal under certain constraints. However, this optimality has often been overemphasized in the past, and it is not always mentioned that Huffman codes have been shown to be optimal only for *block codes*: codes in which each new character is encoded by a fixed bit pattern made up of an integral number of bits.

The constraint of the integral number of bits had probably been considered as obvious, since the possibility of coding elements in fractional bits is quite surprising. *Arithmetic codes* overcome the limitations of block codes. In fact, arithmetic codes have had a long history [32,33], but became especially popular after Witten *et al.*'s paper [34] in 1987.

The approach taken by arithmetic coding is quite different from that of Huffman coding. Instead of using the probabilities of the different characters to generate codewords, it defines a *process* in the course of which a binary number is generated. Each new character of the text to be encoded allows a more precise determination of the number. When the last character is processed, the number is stored or transmitted.

The encoding process starts with the interval [0,1), which will be narrowed repeatedly. We assign to each character a subinterval, the size of which is proportional to the probability of occurrence of the character. Processing a certain character x is then performed by replacing the current interval by the subinterval corresponding to x. Refer to the example in Fig. 14. We assume our alphabet consists of the four characters {A, B, C, D}, appearing with probabilities 0.4, 0.3, 0.1, and 0.2, respectively. We arbitrarily choose a corresponding partition of the interval [0, 1), for example, [0, 0.1) for C, [0.1, 0.4) for B, [0.4, 0.8) for A, and finally [0.8, 1) for D. This partition is depicted as the leftmost bar in Fig. 14.

Suppose now that the text we wish to encode is BDAAC. The first character is B, so the new interval after the encoding of B is [0.1,0.4). This interval is now partitioned similarly to the original one; i.e., the first 10% are assigned to C, the

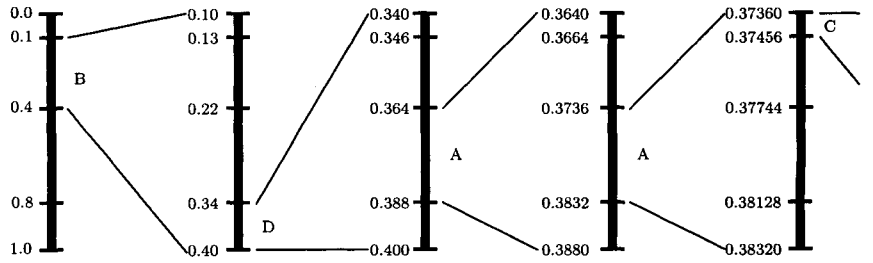


FIGURE 14 Example of arithmetic coding.

next 30% to B, etc. The new subdivision can be seen next to the second bar from the left. The second character to be encoded is D, so the corresponding interval is [0.34, 0.40). Repeating now the process, we see that the next character, A, narrows the chosen subinterval further to [0.364,0.388), and the next A to [0.3736, 0.3832), and finally the last C to [0.37360, 0.37456).

To allow unambiguous decoding, it is this last interval that should be transmitted. This would, however, be rather wasteful: as more characters are encoded, the interval will get narrower, and many of the leftmost digits of its upper limit will overlap with those of its lower limit. In our example, both limits start with 0.37. One can overcome this inefficiency and transmit only a single number if some additional information is given. For instance, if the number of characters is also given to the decoder or, as is customary, a special end-of-file character is added at the end of the message, it suffices to transmit any single number within the final interval. In our case, the best choice would be $y = 0.3740234375$, because its binary representation 0.0101111111 is the shortest among the numbers of the interval.

Decoding is then just the inverse of the above process. Since y is between 0.1 and 0.4, we know that the first character must be B. If so, the interval has been narrowed to [0.1, 0.4). We thus seek the next subinterval that contains y, and find it to be [0.34, 0.40), which corresponds to D, etc. Once we get to [0.37360, 0.37456), the process must be stopped by some external condition; otherwise we could continue this decoding process indefinitely, for example, by noting that y belongs to [0.373984, 0.374368), which could be interpreted as if the following character were A.

As has been mentioned, the longer the input string, the more digits or bits are needed to specify a number encoding the string. Compression is achieved by the fact that a frequently occurring character only slightly narrows the current interval. The number of bits needed to represent a number depends on the required precision. The smaller the given interval, the higher the precision necessary to specify a number in it; if the interval size is p, $\lceil -\log_2 p \rceil$ bits might be needed.

To evaluate the number of bits necessary by arithmetic coding, we recall the notation used in Section II. A. The text consists of characters $x_1 x_2 \cdots x_W$, each of which belongs to an alphabet $\{a_1, \ldots, a_n\}$. Let w_i be the number of occurrences of letter a_i, so that $W = \sum_{i=1}^{n} w_i$ is the total length of the text, and let $p_i = w_i / W$ be the probability of occurrence of letter a_i, $1 \leq i \leq n$. Denote by p_{x_j} the probability associated with the jth character of the text.

After having processed the first character, x_1, the interval has been narrowed to size p_{x_1}; after the second character, the interval size is $p_{x_1} p_{x_2}$; etc. We get that the size of the final interval after the whole text has been processed is $p_{x_1} p_{x_2} \cdots p_{x_W}$. Therefore the number of bits needed to encode the full text is

$$-\log_2 \left(\prod_{j=1}^{W} p_{x_j} \right) = -\sum_{j=1}^{W} \log_2 p_{x_j} = -\sum_{i=1}^{n} w_i \log_2 p_i$$

$$= W \left(-\sum_{i=1}^{n} p_i \log_2 p_i \right) = W H,$$

where we get the second equality by summing over the letters of the alphabet with their frequency instead of summing over the characters of the text, and where H is the entropy of the given probability distribution. Amortizing this per character, we get that the average number of bits needed to encode a single character is just H, which has been shown in Eq. (5) to be the information theoretic lower bound.

We conclude that from the point of view of compression, arithmetic coding has an optimal performance. However, our presentation and the analysis are oversimplified: they do not take into account the overhead incurred by the end_of_file character nor the fractions of bits lost by alignment for each block to be encoded. It can be shown [28] that although these additions are often negligible relative to the average size of a codeword, they might be significant relative to the *difference* between the codeword lengths for Huffman and arithmetic codes. There are also other technical problems, such as the limited precision of our computers, which does not allow the computation of a single number for a long text; there is thus a need for incremental transmission, which further complicates the algorithms, see [34].

Despite the optimality of arithmetic codes, Huffman codes may still be the preferred choice in many applications: they are much faster for encoding and especially decoding, they are less error prone, and after all, the loss in compression efficiency, if any, is generally very small.

E. Dictionary-Based Text Compression

The text compression methods we have seen so far are called *statistical* methods, as they exploit the skewness of the distribution of occurrence of the characters. Another family of compression methods is based on *dictionaries,* which replace variable-length substrings of the text by (shorter) pointers to a dictionary in which a collection of such substrings has been stored. Depending on the application and the implementation details, each method can outperform the other.

Given a fixed amount of RAM that we would allocate for the storage of a dictionary, the selection of an optimal set of strings to be stored in the dictionary turns out to be a difficult task, because the potential strings are overlapping. A similar problem is shown to be NP-complete in [35], but more restricted versions of this problem of optimal dictionary construction are tractable [36].

For IR applications, the dictionary ought to be fixed, since the compressed text needs to be accessed randomly. For the sake of completeness, however,

we mention also *adaptive* techniques, which are the basis of most popular compression methods. Many of these are based on two algorithms designed by Ziv and Lempel [37,38].

In one of the variants of the first algorithm [37], often referred to as LZ77, the dictionary is in fact the previously scanned text, and pointers to it are of the form (d, ℓ), where d is an offset (the number of characters from the current location to the previous occurrence of a substring matching the one that starts at the current location), and ℓ is the length of the matching string. There is therefore no need to store an explicit dictionary. In the second algorithm [38], the dictionary is dynamically expanded by adjoining substrings of the text that could not be parsed. For more details on LZ methods and their variants, the reader is referred to [25].

Even once the dictionary is given, the compression scheme is not yet well defined, as one must decide how to *parse* the text into a sequence of dictionary elements. Generally, the parsing is done by a *greedy* method; i.e., at any stage, the longest matching element from the dictionary is sought. A greedy approach is fast, but not necessarily optimal. Because the elements of the dictionary are often overlapping, and particularly for LZ77 variants, where the dictionary is the text itself, a different way of parsing might yield better compression. For example, assume the dictionary consists of the strings $D = \{$`abc`, `ab`, `cdef`, `d`, `de`, `ef`, `f`$\}$ and that the text is $S =$ `abcdef`; assume further that the elements of D are encoded by some fixed-length code, which means that $\lceil \log_2(|D|) \rceil$ bits are used to refer to any of the elements of D. Then parsing S by a greedy method, trying to match always the longest available string, would yield `abc-de-f`, requiring three codewords, whereas a better partition would be `ab-cdef`, requiring only two.

The various dictionary compression methods differ also by the way they encode the elements. This is most simply done by a fixed length code, as in the above example. Obviously, different encoding methods might yield different optimal parsings. Returning to the above example, if the elements `abc`, `d`, `de`, `ef`, `f`, `ab`, `cdef` of D are encoded respectively by 1, 2, 3, 4, 5, 6, and 6 bits, then the parsing `abc-de-f` would need 9 bits for its encoding, and for the encoding of the parsing `ab-cdef`, 12 bits would be needed. The best parsing, however, for the given codeword lengths, is `abc-d-ef`, which neither is a greedy parsing nor does it minimize the number of codewords, and requires only seven bits.

The way to search for the optimal parsing is by reduction to a well-known graph theoretical problem. Consider a text string S consisting of a sequence of n characters $S_1 S_2 \cdots S_n$, each character S_i belonging to a fixed alphabet Σ. Substrings of S are referenced by their limiting indices; i.e., $S_i \cdots S_j$ is the substring starting at the ith character in S, up to and including the jth character. We wish to compress S by means of a dictionary D, which is a set of character strings $\{\sigma_1, \sigma_2, \ldots\}$, with $\sigma_i \in \Sigma^+$. The dictionary may be explicitly given and finite, as in the example above, or it may be potentially infinite, e.g., for the Lempel–Ziv variants, where any previously occurring string can be referenced.

The compression process consists of two independent phases: parsing and encoding. In the *parsing* phase, the string S is broken into a sequence of consecutive substrings, each belonging to the dictionary D, i.e., an increasing sequence

of indices $i_0 = 0, i_1, i_2, \ldots$ is found, such that

$$S = S_1 S_2 \cdots S_n = S_1 \cdots S_{i_1} S_{i_1+1} \cdots S_{i_2} \cdots,$$

with $S_{i_j+1} \cdots S_{i_{j+1}} \in D$ for $j = 0, 1, \ldots$. One way to assure that at least one such parsing exists is to force the dictionary D to include each of the individual characters of Σ. The second phase is based on an *encoding* function $\lambda : D \longrightarrow \{0,1\}^*$, which assigns to each element of the dictionary a binary string, called its encoding. The assumption on λ is that it produces a code that is UD. This is most easily obtained by a fixed length code, but as has been seen earlier, a sufficient condition for a code being UD is to choose it as a prefix code.

The problem is the following: given the dictionary D and the encoding function λ, we are looking for the optimal partition of the text string S, i.e., the sequence of indices $i_1, i_2, \ldots$ is sought, that minimizes $\sum_{j \geq 0} |\lambda(S_{i_j+1} \cdots S_{i_{j+1}})|$.

To solve the problem, a directed, labeled graph $G = (V, E)$ is defined for the given text S. The set of vertices is $V = \{1, 2, \ldots, n, n+1\}$, with vertex i corresponding to the character S_i for $i \leq n$, and $n+1$ corresponding to the end of the text; E is the set of directed edges: an ordered pair (i, j), with $i < j$, belongs to E if and only if the corresponding substring of the text, that is, the sequence of characters $S_i \cdots S_{j-1}$, can be encoded as a single unit. In other words, the sequence $S_i \cdots S_{j-1}$ must be a member of the dictionary, or more specifically for LZ77, if $j > i+1$, the string $S_i \cdots S_{j-1}$ must have appeared earlier in the text. The label L_{ij} is defined for every edge $(i, j) \in E$ as $|\lambda(S_i \cdots S_{j-1})|$, the number of bits necessary to encode the corresponding member of the dictionary, for the given encoding scheme at hand. The problem of finding the optimal parsing of the text, relative to the given dictionary and the given encoding scheme, therefore reduces to the well-known problem of finding the shortest path in G from vertex 1 to vertex $n+1$. In our case, there is no need to use Dijkstra's algorithm, since the directed graph contains no cycles, all edges being of the form (i, j) with $i < j$. Thus by a simple dynamic programming method, the shortest path can be found in time $O(|E|)$.

Figure 15 displays a small example of a graph, corresponding to the text abbaabbabab and assuming that LZ77 is used. The edges connecting vertices i to $i+1$, for $i = 1, \ldots, n$, are labeled by the character S_i.

As an example of an encoding scheme, we refer to the on-the-fly compression routine recently included in a popular operating system. It is based on [39], a variant of LZ77, using hashing on character pairs to locate (the beginning of)

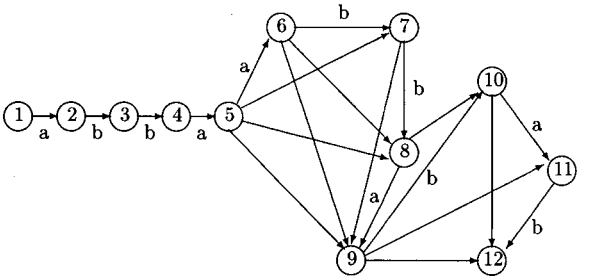


FIGURE 15 Graph corresponding to text abbaabbabab.

recurrent strings. The output of the compression process is thus a sequence of elements, each being either a single (uncompressed) character or an offset-length pair (d, ℓ). The elements are identified by a flag bit, so that a single character is encoded by a zero, followed by the 8-bit ASCII representation of the character, and the encoding of each (d, ℓ) pair starts with a 1. The sets of possible offsets and lengths are split into classes as follows: let $B_m(n)$ denote the standard m-bit binary representation of n (with leading zeros if necessary), then, denoting the encoding scheme by λ_M,

$$\lambda_M(\text{offset } d) = \begin{cases} 1B_6(d-1) & \text{if } 1 \le d \le 64 \\ 01B_8(d-65) & \text{if } 64 < d \le 320 \\ 11B_{12}(d-321) & \text{if } 320 < d \le 4416 \end{cases}$$

$$\lambda_M(\text{length } \ell) = \begin{cases} 0 & \text{if } \ell = 2 \\ 1^{j+1}0\ B_j(\ell - 2 - 2^j) & \text{if } 2^j \le \ell - 2 < 2^{j+1}, \\ & \text{for } j = 0, 1, 2, \ldots. \end{cases}$$

For example, the first few length encodings are $0, 10, 1100, 1101, 111000, 111001, 111010, 111011, 11110000$, etc. Offsets are thus encoded by 8, 11, or 15 bits, and the number of bits used to encode the lengths ℓ is 1 for $\ell = 2$ and $2\lceil \log_2(\ell - 1) \rceil$ for $\ell > 2$.

III. DICTIONARIES

All large full-text retrieval systems make extensive use of dictionaries of all kinds. They are needed to quickly access the concordance, they may be used for compressing the text itself, and they generally provide some useful additional information that can guide the user in the choice of his keywords.

Dictionaries can of course be compressed as if they were regular text, but taking their special structure into account may lead to improved methods [40]. A simple, yet efficient, technique is the *prefix omission method* (POM), a formal definition of which can be found in [2], where it is called *front-end compression.*

The method is based on the observation that consecutive entries in a dictionary mostly share some leading letters. Let x and y be consecutive dictionary entries and let m be the length (number of letters) of their longest common prefix. Then it suffices to store this common prefix only once (with x) and to omit it from the following entry, where instead the length m will be kept. This is easily generalized to a longer list of dictionary entries, as in the example in Fig. 16.

Note that the value given for the prefix length does not refer to the string that was actually stored, but rather to the corresponding full-length dictionary entry. The compression and decompression algorithms are immediate.

If the dictionary entries are coded in standard format, with one byte per character, one could use the first byte of each entry in the compressed dictionary to store the value of m. There will mostly be a considerable gain, since the average length of common prefixes of consecutive entries in large dictionaries is generally much larger than 1. Even when the entries are already compressed,

dictionary entry	*prefix length*	*stored suffix*
FORM	0	FORM
FORMALLY	4	ALLY
FORMAT	5	T
FORMATION	6	ION
FORMULATE	4	ULATE
FORMULATING	8	ING
FORTY	3	TY
FORTHWITH	4	HWITH

FIGURE 16 Example of the prefix omission method.

for example, by a character-by-character Huffman code, one would still achieve some savings. For convenience, one could choose a fixed integer parameter k and reserve the first k bits of every entry to represent values of m for $0 \leq m < 2^k$, where k is not necessarily large enough to accommodate the longest omitted prefix. In the above example, k could for example be chosen as 3, and the entry corresponding to FORMULATING would then be (7, TING).

A standard dictionary does, however, not provide the flexibility required by sophisticated systems. For instance, a prominent feature would be the possibility of processing truncated terms of several kinds by means of a variable-length don't-care character *. Examples of the use of * for prefix, suffix, and infix truncation have been given in Section I.

Suffix truncation can be handled by the regular dictionary. To enable prefix truncation, the problem is that the relevant terms are scattered throughout the file and therefore hard to locate. A possible solution is to adjoin an *inverse* dictionary to the system: for each term, form its reversed string, then sort the reversed strings lexicographically. To search, e.g., `for *ache`, we would access the inverse dictionary with the string `ehca`, retrieve the entries prefixed by it (they form a contiguous block), e.g., `ehcadaeh` and `ehcahtoot`, and reverse these strings again to get our terms, e.g., `headache` and `toothache`. The solution of the inverse dictionary cannot be extended to deal with prefix and suffix truncation simultaneously.

An elegant method allowing the processing of any kind of truncation is the *permuted dictionary* suggested in [2]. Given a dictionary, the corresponding permuted dictionary is obtained by the following sequence of steps:

1. append to each term a character / which does not appear in any term;
2. for a term x of length n characters, form $n + 1$ new terms by cyclically shifting the string $x/$ by k characters, $0 \leq k \leq n$;
3. sort the resulting list alphabetically.

Figure 17 shows these steps for the dictionary consisting of the strings JACM, JASIS, and IPM. The first column lists the terms with the appended /. In the second column, the permuted terms generated by the same original term appear consecutively, and the third column is sorted. The last column shows how the permuted dictionary can be compressed by POM.

original	*permuted*	*sorted*	compressed	
			m	*suffix*
JACM	JACM/	/IPM	0	/IPM
JASIS	ACM/J	/JACM	1	JACM
IPM	CM/JA	/JASIS	3	SIS
	M/JAC	ACM/J	0	ACM/J
	/JACM	ASIS/J	1	SIS/J
	JASIS/	CM/JA	0	CM/JA
	ASIS/J	IPM/	0	IPM/
	SIS/JA	IS/JAS	1	S/JAS
	IS/JAS	JACM/	0	JACM/
	S/JASI	JASIS/	2	SIS/
	/JASIS	M/IP	0	M/IP
	IPM/	M/JAC	2	JAC
	PM/I	PM/I	0	PM/I
	M/IP	S/JASI	0	S/JASI
	/IPM	SIS/JA	1	IS/JA

FIGURE 17 Example of the permuted dictionary.

The key for using the permuted dictionary efficiently is a function `get`(x), which accesses the file and retrieves all the strings having x as prefix. These strings are easily located since they appear consecutively, and the corresponding original terms are recovered by a simple cyclic shift. To process truncated terms, all one needs is to call `get()` with the appropriate parameter. Figure 18 shows in its leftmost columns how to deal with suffix, prefix, infix, and simultaneous prefix and suffix truncations. The other columns then bring an example for each of these categories: first the query itself, then the corresponding call to `get()`, the retrieved entries from the permuted dictionary, and the corresponding reconstructed terms.

IV. CONCORDANCES

Every occurrence of every word in the database can be uniquely characterized by a sequence of numbers that give its exact position in the text. Typically, such a sequence would consist of the document number d, the paragraph number p (in the document), the sentence number s (in the paragraph), and the word number w (in the sentence). The quadruple (d, p, s, w) is the *coordinate* of the occurrence, and the corresponding fields will be called for short d-field, p-field,

X*	get(/X)	JA*	get(/JA)	/JACM, /JASIS	JACM, JASIS
*X	get(X/)	*M	get(M/)	M/IP, M/JAC	IPM, JACM
X*Y	get(Y/X)	J*S	get(S/J)	S/JASI	JASIS
X	get(X)	*A*	get(A)	ACM/J, ASIS/J	JACM, JASIS

FIGURE 18 Processing truncated terms with permuted dictionary.

s-field, and w-field. In the following, we assume for the ease of discussion that coordinates of every retrieval system are of this form; however, all the methods can also be applied to systems with a different coordinate structure, such as book–page–line–word, etc. The concordance contains, for every word of the dictionary, the lexicographically ordered list of all its coordinates in the text; it is accessed via the dictionary that contains for every word a pointer to the corresponding list in the concordance. The concordance is kept in compressed form on secondary storage, and parts of it are fetched when needed and decompressed. The compressed file is partitioned into equi-sized blocks such that one block can be read by a single I/O operation.

Since the list of coordinates of any given word is ordered, adjacent coordinates will often have the same d-field, or even the same d- and p-fields, and sometimes, especially for high-frequency words, identical d-, p-, and s-fields. Thus POM can be adapted to the compression of concordances, where to each coordinate a *header* is adjoined, giving the *number of fields* that can be copied from the preceding coordinate; these fields are then omitted. For instance in our model with coordinates (d, p, s, w), it would suffice to keep a header of 2 bits. The four possibilities are don't copy any field from the previous coordinate, copy the d-field, copy the d- and p-fields, and copy the d-, p-, and s-fields. Obviously, different coordinates cannot have all four fields identical.

For convenient computer manipulation, one generally chooses a fixed length for each field, which therefore must be large enough to represent the maximal possible values. However, most stored values are small; thus there is usually much wasted space in each coordinate. In some situations, some space can be saved at the expense of a longer processing time, as in the following example.

At RRP, the maximal length of a sentence is 676 words! Such long sentences can be explained by the fact that in the Responsa literature punctuation marks are often omitted or used very scarcely. At TLF, there is even a "sentence" of more than 2000 words (a modern poem). Since on the other hand most sentences are short and it was preferred to use only field-sizes that are multiples of half-bytes, the following method is used: the size of the w-field is chosen to be one byte (8 bits); any sentence of length $\ell > 256$ words, such that $\ell = 80k + r (0 \leq r < 80)$, is split into k units of 80 words, followed (if $r > 0$) by a sentence of r words. These sentences form only a negligible percentage of the database. While resolving the storage problem, the insertion of such "virtual points" in the middle of a sentence creates some problems for the retrieval process. When in a query one asks to retrieve occurrences of keywords A and B such that A and B are adjacent or that no more than some small number of words appear between them, one usually does not allow A and B to appear in different sentences. This is justified, since "adjacency" and "near vicinity" operators are generally used to retrieve expressions, and not the coincidental juxtaposition of A at the end of a sentence with B at the beginning of the following one. However in the presence of virtual points, the search should be extended also into neighboring "sentences," if necessary, since the virtual points are only artificial boundaries that might have split some interesting expression. Hence this solution further complicates the retrieval algorithms.

The methods presented in the next section not only yield improved compression, but also get rid of the virtual points.

A. Using Variable-Length Fields

The basic idea of all the new methods is to allow the p-, s-, and w-fields to have variable length. As in POM, each compressed coordinate will be prefixed by a header that will encode the information necessary to decompress the coordinate. The methods differ in their interpretation of the header. The choice of the length of every field is based on statistics gathered from the entire database on the distribution of the values in each field. Thus for dynamically changing databases, the compression method would need frequent updates, so that the methods are more suitable for retrieval systems with static databases. However, if the text changes only slowly, say, it is a large corpus to which from time to time some documents that have characteristics similar to the documents already in the corpus are adjoined, then the methods will still perform well, though not optimally.

The codes in the header can have various interpretations: they can stand for a length ℓ, indicating that the corresponding field is encoded in ℓ bits; they can stand for a certain value v, indicating that the corresponding field contains that value; and finally they can indicate that no value for the corresponding field is stored and that the value of the preceding coordinate should be used. This is more general than the prefix-omission technique, since one can decide for every field individually whether to omit it, while in POM, the p-field is only omitted if the d-field is, etc.

The d-field is treated somewhat differently. Since this is the highest level of the hierarchy in our model, this field may contain also very large numbers (there are rarely 500 words in a sentence or 500 sentences in a paragraph, but a corpus may contain tens of thousands of documents). Moreover, the d-fields of most coordinates will contain values, in the representation of which one can save at most one or two bits, if at all. On the other hand, the d-field is the one where the greatest savings are achieved by POM. Thus we shall assume in the sequel that for the d-field, we just keep one bit in the header, indicating whether the value of the preceding coordinate should be copied; if not, the d-field will appear in its entire length.

We now describe the specific methods in detail.

A. The simple method. The header contains codes for the size (in bits) of every field.

(i) Allocate two bits for each of the p-, s-, and w-fields, giving four possible choices for each.

We consider the following variations:

a. One of the possible codes indicates the omission of the field; thus we are left with only three possible choices for the length of each field.
b. The four choices are used to encode field lengths, thus not allowing the use of the preceding coordinate.
c. Use a for the p- and s-fields, and b for the w-field.

Method A(i)c is justified by the fact that consecutive coordinates having the same value in their w-field are rare (3.5% of the concordance at RRP). The reason is that this corresponds to a certain word appearing in the same relative location in different sentences, which is mostly a pure coincidence; on

the other hand, consecutive coordinates having the same value in one of their other fields correspond to a certain word appearing more than once in the same sentence, paragraph, or document, and this occurs frequently. For instance, at RRP, 23.4% of the coordinates have the same s-field as their predecessors, 41.7% have the same p-field, and 51.6% have the same d-field.

Note that the header does not contain the binary encoding of the lengths, since this would require a larger number of bits. By storing a *code* for the lengths the header is kept smaller, but at the expense of increasing decompression time, since a table that translates the codes into actual lengths is needed. This remark applies also to the subsequent methods.

(ii) Allocate three bits in the header for each of the p-, s-, and w-fields, giving 8 possible choices for each.

The idea of (ii) is that by increasing the number of possibilities (and hence the overhead for each coordinate), the range of possible values can be partitioned more efficiently, which should lead to savings in the remaining part of the coordinate. Again three methods corresponding to a, b, and c of (i) were checked.

B. Using some fields to encode frequent values.

For some very frequent values, the code in the header will be interpreted directly as one of the values, and not as the length of the field in which they are stored. Thus, the corresponding field can be omitted in all these cases. However, the savings for the frequent values come at the expense of reducing the number of possible choices for the lengths of the fields for the less frequent values. For instance, at RRP, the value 1 appears in the s-field of more than 9 million coordinates (about 24% of the concordance); thus, all these coordinates will have no s-field in their compressed form, and the code in the part of the header corresponding to the s-field, will be interpreted as "value 1 in the s-field."

(i) Allocate 2 bits in the header for each of the p-, s-, and w-fields; one of the codes points to the most frequent value.
(ii) Allocate 3 bits in the header for each of the p-, s-, and w-fields; three of the codes point to the three most frequent values.

There is no subdivision into methods a, b, and c as in A (in fact the method used corresponds to a), because we concluded from our experiments that it is worth keeping the possibility of using the previous coordinate in case of equal values in some field. Hence, one code was allocated for this purpose, which left only two codes to encode the field lengths in (i) and four codes in (ii). For (ii) we experimented also with allowing two or four of the eight possible choices to encode the two or four most frequent values; however, on our data, the optimum was always obtained for three. There is some redundancy in the case of consecutive coordinates having both the same value in some field, and this value being the most frequent one. There are then two possibilities to encode the second coordinate using the same number of bits. In such a case, the code for the frequent value should be preferred over that pointing to the previous coordinate, as decoding of the former is usually faster.

C. Combining methods A and B.

Choose individually for each of the p-, s-, and w-fields the best of the previous methods.

D. Encoding length combinations.

If we want to push the idea of A further, we should have a code for *every* possible length of a field, but the maxima of the values can be large. For example, at RRP, one needs 10 bits for the maximal value of the w-field, 9 bits for the s-field, and 10 bits for the p-field. This would imply a header length of 4 bits for each of these fields, which cannot be justified by the negligible improvement over method A(ii).

The size of the header can be reduced by replacing the three codes for the sizes of the p-, s-, and w-fields by a single code in the following way. Denote by l_p, l_s, and l_w the lengths of the p-, s-, and w-fields respectively, i.e., the sizes (in bits) of the binary representations without leading zeros of the values stored in them. In our model $1 \leq l_p, l_s, l_w \leq 10$, so there are up to 10^3 possible triplets (l_p, l_s, l_w). However, most of these length combinations occur only rarely, if at all. At RRP, the 255 most frequent (l_p, l_s, l_w) triplets account already for 98.05% of the concordance. Therefore:

(i) Allocate 9 bits as header, of which 1 bit is used for the d-field; 255 of the possible codes in the remaining 8 bits point to the 255 most frequent (l_p, l_s, l_w) triplets; the last code is used to indicate that the coordinate corresponds to a "rare" triplet, in which case the p-, s-, and w-fields appear already in their decompressed form.

Although the "compressed" form of the rare coordinates, including a 9-bit header, may in fact need more space than the original coordinate, we still save on the average.

Two refinements are now superimposed. We first note that one does not need to represent the integer 0 in any field. Therefore one can use a representation of the integer $n-1$ in order to encode the value n, so that only $\lfloor \log_2(n-1) \rfloor + 1$ bits are needed instead of $\lfloor \log_2 n \rfloor + 1$. This may seem negligible, because only one bit is saved and only when n is a power of 2, thus for very few values of n. However, the first few of these values, 1, 2, and 4, appear very frequently, so that in fact this yields a significant improvement. At RRP, the total size of the compressed p-, s-, and w-fields (using method D) was further reduced by 7.4%, just by shifting the stored values from n to $n-1$.

The second refinement is based on the observation that since we know from the header the exact length of each field, we know the position of the left-most 1 in it, so that this 1 is also redundant. The possible values in the fields are partitioned into classes C_i defined by $C_0 = \{0\}, C_i = \{\ell : 2^{i-1} \leq \ell < 2^i\}$, and the header gives for the values in each of the p-, s-, and w-fields the indices i of the corresponding classes. Therefore if $i \leq 1$, there is no need to store any additional information because C_0 and C_1 are singletons, and for $\ell \in C_i$ for $i > 1$, only the $i-1$ bits representing the number $\ell - 2^{i-1}$ are kept. For example, suppose the values in the p-, s-, and w-fields are 3, 1, and 28. Then the encoded values are 2, 0, and 27, which belong to C_2, C_0, and C_5 respectively. The header thus points to the triplet (2, 0, 5) (assuming that this is one of the 255 frequent ones), and the rest of the coordinate consists of the five bits 01011, which are parsed from left

to right as 1 bit for the p-field, 0 bits for the s-field, and 4 bits for the w-field. A similar idea was used in [15] for encoding run lengths in the compression of sparse bit vectors.

(ii) Allocate 8 bits as header, of which 1 bit is used for the d-field; the remaining 7 bits are used to encode the 127 most frequent (l_p, l_s, l_w) triplets.

The 127 most frequent triplets still correspond to 85.19% of the concordance at RRP. This is therefore an attempt to save one bit in the header of each coordinate at the expense of having more noncompressed coordinates.

Another possibility is to extend method D also to the d-field. Let b be a Boolean variable corresponding to the two possibilities for the d-field, namely T = the value is identical to that of the preceding coordinate; thus omit it, or F = different value, keep it. We therefore have up to 2000 quadruples (b, l_p, l_s, l_w), which are again sorted by decreasing frequency.

(iii) Allocate 8 bits as header; 255 of the codes point to the 255 most frequent quadruples.

At RRP, these 255 most frequent quadruples cover 87.08% of the concordance. For the last two methods, one could try to get better results by compressing also some of the coordinates with the nonfrequent length combinations, instead of storing them in their decompressed form. We did not, however, pursue this possibility.

1. Encoding

After choosing the appropriate compression method, the concordance is scanned sequentially and each coordinate is compressed with or without using the preceding one. For each of the above methods, the length of the header is constant; thus, the set of compressed coordinates forms a prefix code. Therefore, the compressed coordinates, which have variable lengths, can simply be concatenated. The compressed concordance consists of the resulting very long bit string. This string is partitioned into blocks of equal size, the size corresponding to the buffer size of a read/write operation. If the last coordinate in a block does not fit there in its entirety, it is moved to the beginning of the next block. The first coordinate of each block is considered as having no predecessor, so that if in the original encoding process a coordinate that is the first in a block referred to the previous coordinate, this needs to be corrected. This makes it possible now to access each block individually, while adding only a negligible number of bits to each block.

2. Decoding

Note that for a static information retrieval system, encoding is done only once (when building the database), whereas decoding directly affects the response time for on-line queries. In order to increase the decoding speed, we use a small precomputed table $\mathcal{T}$ that is stored in internal memory. For a method with header length k bits, this table has 2^k entries. In entry i of $\mathcal{T}$, $0 \leq i < 2^k$, we store the relevant information for the header consisting of the k-bit binary representation of the integer i.

For the methods in A, the relevant information simply consists of the lengths, P, S, and W, of the p-, s-, and w-fields (recall that we assume that only one bit is kept in the header for the d-field, so either the d-field appears in its entire length D, which is constant, or it is omitted), and of the sum of all these lengths (including D), which is the length of the remaining part of the coordinate. We shall use the following notations: for a given internal structure of a decompressed coordinate, let h_d, h_p, h_s, and h_w be the indices of the leftmost bit of the d-, p-, s-, and w-fields respectively, the index of the rightmost bit of a coordinate being 0. For example with a 4-byte coordinate and one byte for each field we would have $h_d = 31$, $h_p = 23$, $h_s = 15$, and $h_w = 7$; these values are constant for the entire database. COOR and LAST are both addresses of a contiguous space in memory in which a single decompressed coordinate can fit (hence of length $h_d + 1$ bits). The procedure SHIFT(X, y, z) shifts the substring of X, which is obtained by ignoring its y rightmost bits, by z bits to the left. Then the following loop could be used for the decoding of a coordinate:

1. **loop** while there is more input or until a certain coordinate is found
2. $H \leftarrow$ next k bits // read header
3. (TOT, P, S, W) $\leftarrow \mathcal{T}(H)$ // decode header using table
4. COOR $\leftarrow$ next TOT bits // right justified suffix of coordinate
5. SHIFT(COOR, $W, h_w - W$) // move d-, p-, and s-fields
6. SHIFT(COOR, $h_w + S, h_s - S$) // move d- and p-fields
7. SHIFT(COOR, $h_s + P, h_p - P$) // move d-field
8. **if** TOT $= P + S + W$ **then** copy d-field from LAST
9. **if** $P = 0$ **then** copy p-field from LAST
10. **if** $S = 0$ **then** copy s-field from LAST
11. **if** $W = 0$ **then** copy w-field from LAST
12. LAST $\leftarrow$ COOR
13. **end** of loop

There is no need to initialize LAST, since the first coordinate of a block never refers to the preceding coordinate.

For the methods in B and C, we store sometimes actual values, and not just the lengths of the fields. This can be implemented by using negative values in the table $\mathcal{T}$. For example, if $P = -2$, this could be interpreted as "value 2 in the p-field." Note that when the value stored in a field is given by the header, this field has length 0 in the remaining part of the coordinate. Thus we need the following updates to the above algorithm: line 3 is replaced by

(TOT, $P1$, $S1$, $W1$) $\leftarrow \mathcal{T}(H)$
if $P1 < 0$ **then** $P \leftarrow 0$ **else** $P \leftarrow P1$

and statements similar to the latter for the s- and w-fields. After statement 11 we should insert

if $P1 < 0$ **then** put $-P1$ in p-field of COOR

and similar statements for the s- and w-fields.

The decoding of the methods in D is equivalent to that of A. The only difference is in the preparation of the table $\mathcal{T}$ (which is done only once). While for A to each field correspond certain fixed bits of the header that determine the length of that field, for D the header is nondivisible and represents the lengths

TABLE 3 Distribution of Values Stored in p-, s-, and w-Fields

Value		1	2	3	4	5	79	83	87	93	119	120
Ignoring	p-field	14.1	35.2	46.5	54.2	60.2		99				
Preceding	s-field	24.2	40.2	51.1	58.8	64.5	99					
Coordinate	w-field	3.0	5.8	8.6	11.4	14.0					99	
Using	p-field	9.6	25.2	36.5	45.0	51.7				99		
Preceding	s-field	17.9	33.0	44.3	52.6	58.9			99			
Coordinate	w-field	1.9	4.4	7.1	9.7	12.4						99

of all the fields together. This does not affect the decoding process, since in both methods a table-lookup is used to interpret the header. An example of the encoding and decoding processes appears in the next section.

3. Parameter Setting

All the methods of the previous section were compared on the concordance of RRP. Each coordinate had a (d, p, s, w) structure and was of length 6 bytes (48 bits). Using POM, the average length of a compressed coordinate was 4.196 bytes, i.e., a compression gain of 30%.

Table 3 gives the frequencies of the first few values in each of the p-, s-, and w-fields, both with and without taking into account the previous coordinate. The frequencies are given in cumulative percentages; e.g., the row entitled s-field contains in the column headed i the percentage of coordinates having a value $\leq i$ in their s-field. We have also added the values for which the cumulative percentage first exceeds 99%.

As one can see, the first four values in the p- and s-fields account already for half of the concordance. This means that most of the paragraphs consist of only a few sentences and most of the documents consist of only a few paragraphs. The figures for the w-field are different, because short sentences are not preponderant. While the (noncumulative) frequency of the values i in the s-field is a clearly decreasing function of i, it is interesting to note the peek at value 2 for the p-field. This can be explained by the specific nature of the Responsa literature, in which most of the documents have a question–answer structure. Therefore the first paragraph of a document usually contains just a short question, whereas the answer, starting from the second paragraph, may be much longer.

When all the coordinates are considered (upper half of Table 3), the percentages are higher than the corresponding percentages for the case where identical fields in adjacent coordinates are omitted (lower half of Table 3). This means that the idea of copying certain fields from the preceding coordinate yields to savings, which are, for the small values, larger than could have been expected from knowing their distribution in the noncompressed concordance.

Using the information collected from the concordance, all the possible variants for each of the methods in A and B have been checked. Table 4 lists for each of the methods the variant for which maximal compression was achieved. The numbers in boldface are the frequent values used in methods B and C; the other numbers refer to the lengths of the fields. The value 0 indicates that the field of the preceding coordinate should be copied.

TABLE 4 Optimal Variants of the Methods

Method	p-field	s-field	w-field
A(i)a	0 2 5 10	0 2 5 9	0 4 6 10
A(i)b	1 3 5 10	1 3 5 9	3 5 6 10
A(ii)a	0 1 2 3 4 5 6 10	0 1 2 3 4 5 6 9	0 1 3 4 5 6 7 10
A(ii)b	1 2 3 4 5 6 7 10	1 2 3 4 5 6 7 9	1 2 3 4 5 6 7 10
B(i)	0 **2** 4 10	0 **1** 4 9	0 **4** 6 10
B(ii)	0 **1 2 3** 3 4 5 10	0 **1 2 3** 3 4 5 9	0 **3 4 5** 3 5 6 10
C	0 2 5 10	0 **1 2 3** 3 4 5 9	3 5 6 10

The optimal variants for the methods A(ii) are not surprising: since most of the stored values are small, one could expect the optimal partition to give priority to small field lengths. For method C, each field is compressed by the best of the other methods, which are A(i)a for the p-field, B(ii) for the s-field, and A(i)b for the w-field, thus requiring a header of $1 + 2 + 3 + 2 = 8$ bits (including one bit for the d-field).

The entries of Table 4 were computed using the first refinement mentioned in the description of method D, namely storing $n - 1$ instead of n. The second refinement (dropping the leftmost 1) could not be applied, because it is not true that the leftmost bit in every field is a 1. Thus for all the calculations with methods A and B, an integer n was supposed to require $\lfloor \log_2(n-1) \rfloor + 1$ bits for $n > 1$ and one bit for $n = 1$.

As an example for the encoding and decoding processes, consider method C, and a coordinate structure with $(h_d, h_p, h_s, h_w) = (8, 8, 8, 8)$, i.e., one byte for each field. The coordinate we wish to process is (159, 2, 2, 35). Suppose further that only the value in the d-field is the same as in the previous coordinate. Then the length D of the d-field is 0; in the p-field the value 1 is stored, using two bits; nothing is stored in the s-field, because 2 is one of the frequent values and directly referenced by the header; and in the w-field the value 34 is stored, using 6 bits. The possible options for the header are numbered from left to right as they appear in Table 4; hence the header of this coordinate is 0-10-011-11, where dashes separating the parts corresponding to different fields have been added for clarity; the remaining part of the coordinate is 01-100010. Table $\mathcal{T}$ has $2^8 = 256$ entries; at entry 79 (= 01001111 in binary) the values stored are (TOT, $P1$, $S1$, $W1$) = (8, 2, −2, 6). When decoding the compressed coordinate 0100111101100010, the leftmost 8 bits are considered as header and converted to the integer 79. Table $\mathcal{T}$ is then accessed with that index, retrieving the 4-tuple (8, 2, −2, 6), which yields the values (P, S, W) = (2, 0, 6). The next TOT = 8 bits are therefore loaded into COOR of size 4 bytes, and after the three shifts we get

$$\text{COOR} = 00000000 - 00000010 - 00000000 - 00100010.$$

Since TOT = $P + S + W$ the value of the d-field is copied from the last coordinate. Since $P1 < 0$, the value $-S1 = 2$ is put into the s-field.

On our data, the best method was D(i) with an average coordinate length of 3.082 bytes, corresponding to 49% compression relative to the full 6-byte coordinate, and giving a 27% improvement over POM. The next best method was C with 3.14 bytes. Nevertheless, the results depend heavily on the statistics

of the specific system at hand, so that for another database, other methods could be preferable.

The main target of the efforts was to try to eliminate or at least reduce the unused space in the coordinates. Note that this can easily be achieved by considering the entire database as a single long run of words, which we could index sequentially from 1 to N, N being the total number of words in the text. Thus $\lfloor \log_2 N \rfloor + 1$ bits would be necessary per coordinate. However, the hierarchical structure is lost, so that, for example, queries asking for the cooccurrence of several words in the same sentence or paragraph are much harder to process. Moreover, when a coordinate is represented by a single, usually large, number, we lose also the possibility of omitting certain fields that could be copied from preceding coordinates. A hierarchical structure of a coordinate is therefore preferable for the retrieval algorithms. Some of the new compression methods even outperform the simple method of sequentially numbering the words, since the latter would imply at the RRP database a coordinate length of 26 bits = 3.25 bytes.

B. Model-Based Concordance Compression

For our model of a textual database, we assume that the text is divided into documents and the documents are made up of words. We thus use only a two-level hierarchy to identify the location of a word, which makes the exposition here easier. The methods can, however, be readily adapted to more complex concordance structures, like the 4-level hierarchy mentioned above. In our present model, the conceptual concordance consists, for each word, of a series of (d, w) pairs, d standing for a document number, and w for the index, or offset, of a word within the given document:

$$\begin{aligned} \text{word}_1 : \quad & (d_1, w_1)(d_1, w_2) \cdots (d_1, w_{m_1}) \\ & (d_2, w_1)(d_2, w_2) \cdots (d_2, w_{m_2}) \\ & \cdots \\ & (d_N, w_1) \cdots (d_N, w_{m_N}) \\ \text{word}_2 : \quad & \cdots \end{aligned}$$

For a discussion of the problems of relating this conceptual location to a physical location on the disc, see [7].

It is sometimes convenient to translate our 4-level hierarchy to an equivalent one, in which we indicate the index of the next document containing the word, the number of times the word occurs in the document, followed by the list of word indices of the various occurrences:

$$\begin{aligned} \text{word}_1 : \quad & (d_1, m_1; w_1, w_2, \ldots, w_{m_1}) \\ & (d_2, m_2; w_1, \ldots, w_{m_2}) \\ & \cdots \\ & (d_N, m_N; w_1, \ldots, w_{m_N}) \\ \text{word}_2 : \quad & \cdots \end{aligned}$$

Our task is to model each of the components of the latter representation, and use standard compression methods to compress each entity. Below we assume that we know (from the dictionary) the total number of times a word occurs in the database, the number of different documents in which it occurs, and (from a separate table) the number of words in each document. The compression algorithm is then based on predicting the probability distribution of the various values in the coordinates, devising a code based on the predicted distributions, and using the codeword corresponding to the actual value given.

We thus need to generate a large number of codes. If so, the Shannon–Fano method (as defined in [41]) seems the most appropriate if we are concerned with processing speed. Thus an element, which according to the model at hand appears with probability p, will be encoded by $\lceil -\log_2 p \rceil$ bits. Once the length of the codeword is determined, the actual codeword is easily generated. However, Shannon–Fano codes are not optimal and might in fact be quite wasteful, especially for the very low probabilities.

While Shannon–Fano coding is fast, when high precision is required Huffman codes are a good alternative. Under the constraint that each codeword consists of an integral number of bits, they are optimal; however, their computation is much more involved than that of Shannon–Fano codes, because every codeword depends on the whole set of probabilities. Thus more processing time is needed, but compression is improved. On the other hand, Huffman codes are not effective in the presence of very high probabilities. Elements occurring with high probability have low information content; yet their Huffman codeword cannot be shorter than one bit. If this is a prominent feature, arithmetic coding must be considered.

Arithmetic coding more directly uses the probabilities derived from the model, and overcomes the problem of high-probability elements by encoding entire messages, not just codewords. Effectively, an element with probability p is encoded by exactly $-\log_2 p$ bits, which is the information theoretic minimum. While in many contexts arithmetic codes might not improve much on Huffman codes, their superiority here might be substantial, because the model may generate many high probabilities. There is of course a time/space tradeoff, as the computation of arithmetic codes is generally more expensive than that of Huffman codes.

Initially we are at the beginning of the document list and are trying to determine the probability that the next (when we start, this is the first) document containing a term is d documents away from our current location. We know the number of documents that contain the term, say N, and the number of documents, say D, from which these are chosen. (More generally, after we have located a number of documents that contain the term, D and N will respectively represent the total number of remaining documents and, of these, the number that contain the term. Our reasoning will then continue in parallel to that of the first occurrence.)

Our first question, then, is what is the probability distribution of the first/next document containing the term? Assuming that the events involved are independent and equally distributed, this is equivalent to asking, if N different objects are selected at random from an ordered set of D objects, what is the probability that d is the index of the object with minimum index?

Because of the uniformity assumption, each of the $\binom{D}{N}$ ways of picking N out of D objects have same probability, viz, $1/\binom{D}{N}$, but of these, only $\binom{D-d}{N-1}$ satisfy the condition that d is the minimum index. That is, certainly one document must be the dth one, so we only have freedom to choose $N-1$ additional documents. Since all of these must have index greater than d, we have only $D-d$ options for these $N-1$ selections. Thus the probability that the next document has (relative) position d is $\Pr(d) = \binom{D-d}{N-1}/\binom{D}{N}$.

We first note this is a true probability

$$\sum_d \Pr(d) = \sum_{d=1}^{D-N+1} \frac{\binom{D-d}{N-1}}{\binom{D}{N}} = \sum_{k=N-1}^{D-1} \frac{\binom{k}{N-1}}{\binom{D}{N}} = \frac{\binom{D}{N}}{\binom{D}{N}} = 1,$$

where the last equality uses the well-known combinatoric identity that permits summation over the upper value in the binomial coefficient [11]. Second, we note that we can rewrite the probability as

$$\left(\frac{N}{D-N+1}\right) \times \left(1 - \frac{d}{D}\right) \times \left(1 - \frac{d}{D-1}\right) \times \cdots \times \left(1 - \frac{d}{D-N+2}\right).$$

If $d \ll D$, this is approximately $(N/D) \times (1 - d/D)^{N-1}$, which is in turn approximately proportional to $e^{-d(N-1)/D}$ or γ^d, for $\gamma = e^{-(N-1)/D}$. This last form is that of the geometric distribution recommended by Witten *et al.* [42].

The encoding process is then as follows. We wish to encode the d-field of the next coordinates $(d, m; w_1, \ldots, w_m)$. Assuming that the probability distribution of d is given by $\Pr(d)$, we construct a code based on $\{\Pr(d)\}_{d=1}^{D-N+1}$. This assigns codewords to all the possible values of d, from which we use the codeword corresponding to the actual value d in our coordinate. If the estimate is good, the actual value d will be assigned a high probability by the model, and therefore be encoded with a small number of bits.

Next we encode the number of occurrences of the term in this document. Let us suppose that we have T occurrences of the term remaining (initially, this will be the total number of occurrences of the term in the database). The T occurrences are to be distributed into the N remaining documents the word occurs in. Thus we know that each document being considered must have at least a single term, that is, $m = 1 + x$, where $x \geq 0$. If $T = N$, then clearly $x = 0$ $(m = 1)$, and we need output no code—m conveys no information in this case. If $T > N$, then we must distribute the $T - N$ terms not accounted for over the remaining N documents that contain the term. We assume, for simplicity, that the additional amount, x, going to the currently considered document is Poisson distributed, with mean $\lambda = (T - N)/N$. The Poisson distribution is given by $\Pr(x) = e^{-\lambda}\frac{\lambda^x}{x!}$. This allows us to compute the probability of x for all possible values $(x = 0, 1, \ldots, T - N)$ and to then encode x using one of the encodings above.

We must finally encode all the m offsets, but this problem is formally identical to that of encoding the next document. The current document has W words, so the distribution of w, the first occurrence of the word, is given by the probabilities $\binom{W-w}{m-1}/\binom{W}{m}$. Once this is encoded, we have a problem identical to the initial one in form, except that we now have $m - 1$ positions left to encode and $W - w$ locations. This continues until the last term, which is uniformly distributed over the remaining word locations.

Then we encode the next document, but this is again a problem identical in form to the initial problem—only we now have one fewer document $(N-1)$ having the term, and d fewer target documents $(D-d)$ to consider.

The formal encoding algorithm is given in Fig. 19. We begin with a conceptual concordance, represented for the purpose of this algorithm as a list of

```
{
    for s ⟵ 1 to S     /* for each word in concordance */
    {
        D  ⟵  total number of documents
        T  ⟵  total number of occurrences of word s
        N  ⟵  total number of documents in which word s occurs
        d_0 ⟵ 0
        for i ⟵ 1 to N     /* for each document containing word s */
        {
            /* process document i */
            output d_code(d_i − d_{i−1}, N − i, D)
            if T > N
                output m_code(m_i − 1,  (T − N)/N,  T − N)
            /* process occurrences of word s in document i */
            W  ⟵  total number of words in document i
            w_0 ⟵ 0
            for j ⟵ 1 to m_i
            {
                output w_code(w_j − w_{j−1}, m_i − j, W)
                W  ⟵  W − (w_j − w_{j−1})
            }
            /* update parameters and continue */
            D  ⟵  D − (d_i − d_{i−1})
            T  ⟵  T − m_i
        }
    }
}
d_code(d, N, D)
{
    construct a code C_1 based on probabilities that d = k: { C(D−k, N) / C(D, N+1) }_{k=1}^{D−i+1}
    return C_1(d)
}
m_code(x, λ, max)
{
    F  ⟵  Σ_{k=0}^{max} e^{−λ} λ^k / k!     /* correction factor for truncated Poisson distribution */
    construct a code C_2 based on probabilities that x = k: { (1/F) e^{−λ} λ^k / k! }_{k=0}^{max}
    return C_2(x)
}
w_code(w, m, W)
{
    construct a code C_3 based on probabilities that w = k: { C(W−k, m) / C(W, m+1) }_{k=1}^{W−i+1}
    return C_3(w)
}
```

FIGURE 19 Concordance Compression Algorithm.

entries. Our concordance controls S different words. For each word, there is an entry for each document it occurs in, of the form $(d_i, m_i; w_1, \ldots, w_{m_i})$, where d_i, m_i, and w_j are given similarly to the second representation defined above.

Note that we do not encode the absolute values d_i and w_j, but the relative increases $d_i - d_{i-1}$ and $w_j - w_{j-1}$; this is necessary, because we redefine, in each iteration, the sizes D, W, and T to be the *remaining* number of documents, the number of words in the current document, and the number of occurrences of the current word, respectively.

In fact, one should also deal with the possibility where the independence assumptions of the previous section are not necessarily true. In particular, we consider the case where terms cluster not only within a document, but even at the between document level. Details of this model can be found in [43].

V. BITMAPS

For every distinct word W of the database, a bitmap $B(W)$ is constructed, which acts as an "occurrence" map at the document level. The length (in bits) of each map is the number of documents in the system. Thus, in the RRP for example, the length of each map is about 6K bytes. These maps are stored in compressed form on a secondary storage device. At RRP, the compression algorithm was taken from [44], reducing the size of a map to 350 bytes on the average. This compression method was used for only about 10% of the words, those which appear at least 70 times; for the remaining words, the *list* of document numbers is kept and transformed into bitmap form at processing time. The space needed for the bitmap file in its entirety is 33.5 MB, expanding the overall space requirement of the entire retrieval system by about 5%.

At the beginning of the process dealing with a query of the type given in Eq. (1), the maps $B(A_{ij})$ are retrieved, for $i = 1, \ldots, m$ and $j = 1, \ldots, n_i$. They are decompressed and a new map ANDVEC is constructed:

$$\text{ANDVEC} = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n_i} B(A_{ij}) \right).$$

The bitmap ANDVEC serves as a "filter," for only documents corresponding to 1 bits in ANDVEC can possibly contain a solution. Note that no more than three full-length maps are simultaneously needed for its construction.

For certain queries, in particular when keywords with a small number of occurrences in the text are used, ANDVEC will consist only of zeros, which indicates that nothing should be retrieved. In such cases the user gets the correct if somewhat meager results, without a single merge or collate action having been executed. However, even if ANDVEC is not null, it will usually be much sparser than its components. These maps can improve the performance of the retrieval process in many ways to be now described.

A. Usefulness of Bitmaps in IR

First, bitmaps can be helpful in reducing the number of I/O operations involved in the query-processing phase. Indeed, since the concordance file is usually too

large to be stored in the internal memory, it is kept in compressed form in secondary storage, and parts of it are fetched when needed and decompressed. The compressed concordance file is partitioned into equisized blocks such that one block can be read by a single I/O operation; it is accessed via the dictionary, which contains for each word a pointer to the corresponding (first) block. A block can contain coordinates of many "small" words (i.e., words with low frequency in the database), but on the other hand, the coordinate list of a single "large" (high-frequency) word may extend over several consecutive blocks. In the RRP, for example, about half of the words appear only once, but on the other hand there are some words that occur hundreds of thousands of times! It is for the large words that the bitmap ANDVEC may lead to significant savings in the number of I/O operations. Rather than reading *all* the blocks to collect the list of coordinates that will later be merged and/or collated, we access only blocks which contain coordinates in the documents specified by the 1 bits of ANDVEC. Hence if the map is sparse enough, only a small *subset* of the blocks need to be fetched and decompressed. To implement this idea, we need, in addition to the bitmap, also a small list $L(W)$ for each large word W, $L(W) = \{(f_j, \ell_j)\}$, where f_j and ℓ_j are respectively the document numbers of the first and last coordinate of W in block number j, and j runs over the indices of blocks that contain coordinates of W. The list $L(W)$ is scanned together with the bitmap, and if there is no 1 bit in ANDVEC in the bit range $[f_j, \ell_j]$, the block j is simply skipped.

There are, however, savings beyond I/O operations. Once a concordance block containing some coordinates that might be relevant is read, it is scanned in parallel with ANDVEC. Coordinates with document numbers corresponding to 0 bits are skipped. For the axis, which is the first keyword A_i to be handled, this means that only parts of the lists $C(A_{ij})$ will be transferred to a working area, where they are merged. In order to save internal memory space during the query processing, the lists of the keywords A_{kj}, for $k \neq i$, are not merged like the lists of the axis, but are directly collated with the axis. Such collations can be involved operations, as the distance constraints may cause each coordinate of the axis to be checked against several coordinates of every variant of other keywords, and conversely each such coordinate might collate with several coordinates of the axis. Therefore the use of ANDVEC may save time by reducing the number of collations. Moreover, after all the variants of the second keyword have been collated with the axis, the coordinates of the axis not matched can be rejected, so that the axis may shrink considerably. Now ANDVEC can be updated by deleting some of its 1 bits, which again tends to reduce the number of read operations and collations when handling the following keywords. The updates of the axis and ANDVEC are repeated after the processing of each keyword A_j of the query (1).

For conventional query-processing algorithms, the consequence of increasing the number m of keywords is an increased processing time, whereas the set of solutions can only shrink. When m is increased with the bitmap approach, however, the time needed to retrieve the maps and to perform some additional logical operations is usually largely compensated for by the savings in I/O operations caused by a sparser ANDVEC. The new approach seems thus to be particularly attractive for a large number of keywords. Users

are therefore encouraged to change their policy and to submit more complex queries!

Another possible application of the bitmaps is for getting a selective display of the results. A user is often not interested in finding *all* the occurrences of a certain phrase in the database, as specified by the query, but only in a small subset corresponding to a certain author or a certain period. The usual way to process such special requests consists in executing first the search ignoring the restrictions, and then filtering out the solutions not needed. This can be very wasteful and time-consuming, particularly if the required subrange (period or author(s)) is small. The bitmaps allow the problem to be dealt with in a natural way, requiring only minor changes to adapt the search program to this application. All we need is to prepare a small repertoire $\mathcal{R}$ of fixed bitmaps, say one for each author, where the 1 bits indicate the documents written by this author, and a map for the documents of each year or period, etc. The restrictions can now be formulated at the same time the query is submitted. In the construction algorithm, ANDVEC will not be initialized by a string containing only 1's, but by a logical combination of elements of $\mathcal{R}$, as induced by the additional restrictions. Thus user-imposed restrictions on required ranges to which solutions should belong on one hand, and query-imposed restrictions on the co-occurrence of keywords on the other, are processed in exactly the same way, resulting in a bit vector, the sparsity of which depends directly on the severity of the restrictions. As was pointed out earlier, this may lead to savings in processing time and I/O operations.

Finally, bitmaps can be also helpful in handling negative keywords. If a query including some negative keywords $\overline{D_i}$ is submitted at the document-level, one can use the binary complements $B(\overline{D_i})$ of the maps, since only documents with no occurrence of D_i (indicated by the 0 bits) can be relevant. However, for other levels, the processing is not so simple. In fact, if the query is not on the document level, the bitmaps of the negative keywords are useless, and ANDVEC is formed only by the maps of the positive keywords. This difference in the treatment of negative and positive keywords is due to the fact that a 0 bit in the bit vector of a positive keyword means that the corresponding document cannot possibly be relevant, whereas a 1 bit in the bit vector of a negative keyword D_j only implies that D_j appears in the corresponding document; however, this document can still be retrieved, if D_j is not in the specified neighborhood of the other keywords. Nevertheless, even though the negative keywords do not contribute in rendering ANDVEC sparser, ANDVEC will still be useful also for the negative words: only coordinates in the relevant documents must be checked *not* to fall in the vicinity of the axis, as imposed by the (l_i, u_i).

B. Compression of Bitmaps

It would be wasteful to store the bitmaps in their original form, since they are usually very sparse (the great majority of the words occur in very few documents). Schuegraf [45] proposes to use *run-length coding* for the compression of sparse bit vectors, in which a string of consecutive 0's terminated by a

1 (called a *run*) is replaced by the length of the run. A sophisticated run-length coding technique can be found in Teuhola [46]. Jakobsson [47] suggests to partition each vector into k-bit blocks, and to apply Huffman coding on the 2^k possible bit patterns. This method is referred to below as the method NORUN.

1. Hierarchical Compression

In this section we concentrate on *hierarchical bit-vector compression:* let us partition the original bit vector v_0 of length l_0 bits into k_0 equal blocks of r_0 bits, $r_0 \cdot k_0 = l_0$, and drop the blocks consisting only of 0's. The resulting sequence of nonzero blocks does not allow the reconstruction of v_0, unless we add a list of the indices of these blocks in the original vector. This list of up to k_0 indices is kept as a binary vector v_1 of $l_1 = k_0$ bits, where there is a 1 in position i if and only if the ith block of v_0 is not all zero. Now v_1 can further be compressed by the same method.

In other words, a sequence of bit vectors v_j is constructed, each bit in v_j being the result of ORing the bits in the corresponding block in v_{j-1}. The procedure is repeated recursively until a level t is reached where the vector length reduces to a few bytes, which will form a single block. The compressed form of v_0 is then obtained by concatenating all the nonzero blocks of the various v_i, while retaining the block-level information. Decompression is obtained simply by reversing these operations and their order. We start at level t, and pass from one level to the next by inserting blocks of zeros into level $j-1$ for every 0 bit in level j.

Figure 20 depicts an example of a small vector v_0 of 27 bits and its derived levels v_1 and v_2, with $r_i = 3$ for $i = 0, 1, 2$ and $t = 2$. The sizes r_j of the blocks are parameters and can change from level to level for a given vector, and even from one word of the database to another, although the latter is not practical

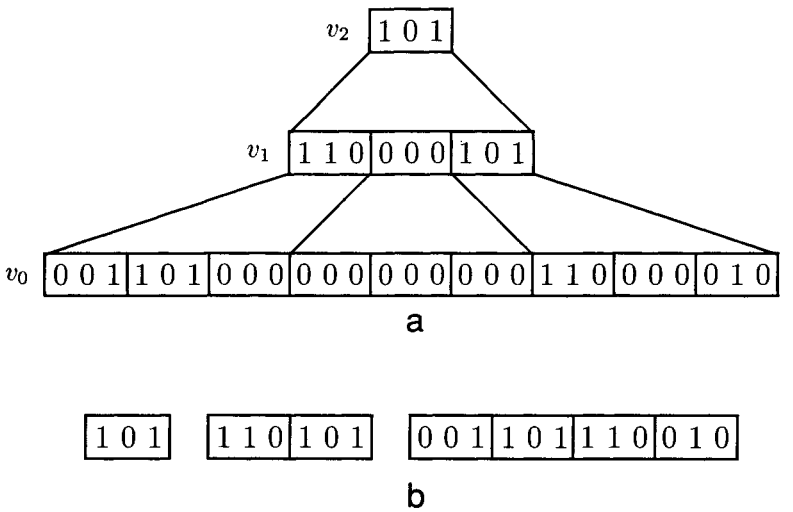


FIGURE 20 Hierarchical bit-vector compression. (a) Original vector and two derived levels and (b) compressed vector.

for our applications. Because of the structure of the compressed vector, we call this the TREE method, and shall use in our discussion the usual tree vocabulary: the *root* of the tree is the single block on the top level, and for a block x in v_{j+1}, which is obtained by ORing the blocks $y_1, \ldots, y_{r_j}$ of v_j, we say that x is the *parent* of the nonzero blocks among the y_i.

The TREE method was proposed by Wedekind and Härder [48]. It appears also in Vallarino [49], who used it for two-dimensional bitmaps, but only with one level of compression. In [50], the parameters (block size and height of the tree) are chosen assuming that the bit vectors are generated by a memoryless information source; i.e., each bit in v_0 has a constant probability p_0 for being 1, independently from each other. However, for bitmaps in information retrieval systems, this assumption is not very realistic a priori, as adjacent bits often represent documents written by the same author; there is a positive correlation for a word to appear in consecutive documents, because of the specific style of the author or simply because such documents often treat the same or related subjects.

We first remark that the hierarchical method does not always yield real compression. Consider, for example, a vector v_0 for which the indices of the 1 bits are of the form ir_0 for $i \leq l_0/r_0$. Then there are no zero-blocks (of size r_0) in v_0; moreover all the bits of v_i for $i > 0$ will be 1, so that the whole tree must be kept. Therefore the method should be used only for sparse vectors.

In the other extreme case, when v_0 is very sparse, the TREE method may again be wasteful: let $d = \lceil \log_2 l_0 \rceil$, so that a d-bit number suffices to identify any bit position in v_0. If the vector is extremely sparse, we could simply list the positions of all the 1 bits, using d bits for each. This is in fact the inverse of the transformation performed by the bit vectors: basically, for every different word W of the database, there is one entry in the inverted file containing the list of references of W, and this list is transformed into a bitmap; here we change the bitmap back into its original form of a list.

A small example will illustrate how the bijection of the previous paragraph between lists and bitmaps can be used to improve method TREE. Suppose that among the $r_0 \cdot r_1 \cdot r_2$ first bits of v_0 only position j contains a 1. The first bit in level 3, which corresponds to the ORing of these bits, will thus be set to 1 and will point to a subtree consisting of three blocks, one on each of the lower levels. Hence in this case a single 1 bit caused the addition of at least $r_0 + r_1 + r_2$ bits to the compressed map, since if it were zero, the whole subtree would have been omitted. We conclude that if $r_0 + r_1 + r_2 \geq d$, it is preferable to consider position j as containing zero, thus omitting the bits of the subtree, and to add the number j to an appended list L, using only d bits. This example is readily generalized so as to obtain an optimal partition between tree and list for every given vector, as will now be shown.

We define l_j and k_j respectively as the number of bits and the number of blocks in v_j, for $0 \leq j \leq t$. Note that $r_j \cdot k_j = l_j$. Denote by $T(i, j)$ the subtree rooted at the ith block of v_j, with $0 \leq j \leq t$ and $1 \leq i \leq k_j$. Let $S(i, j)$ be the size in bits of the compressed form of the subtree $T(i, j)$, i.e., the total number of bits in all the nonzero blocks in $T(i, j)$, and let $N(i, j)$ be the number of 1 bits in the part of the original vector v_0, which belongs to $T(i, j)$.

During the bottom-up construction of the tree these quantities are recursively evaluated for $0 \leq j \leq t$ and $1 \leq i \leq k_j$ by

$$N(i, j) = \begin{cases} \text{number of 1 bits in block } i \text{ of } v_0 & \text{if } j = 0, \\ \sum_{h=1}^{r_j} N((i-1)r_j + h, j-1) & \text{if } j > 0; \end{cases}$$

$$S(i, j) = \begin{cases} 0 & \text{if } j = 0 \text{ and } T(i, 0) \text{ contains only zeros,} \\ r_0 & \text{if } j = 0 \text{ and } T(i, 0) \text{ contains a 1 bit,} \\ r_j + \sum_{h=1}^{r_j} S((i-1)r_j + h, j-1) & \text{if } j > 0. \end{cases}$$

At each step, we check the condition

$$d \cdot N(i, j) \leq S(i, j). \tag{6}$$

If it holds, we prune the tree at the root of $T(i, j)$, adding the indices of the $N(i, j)$ 1 bits to the list L, and setting then $N(i, j)$ and $S(i, j)$ to 0. Hence the algorithm partitions the set of 1 bits into two disjoint subsets: those compressed by the TREE-method and those kept as a list. In particular, if the pruning action takes place at the only block of the top level, there will be no tree at all.

Note that by definition of $S(i, j)$, the line corresponding to the case $j > 0$ should in fact be slightly different: r_j should be added to the sum $X = \sum_{h=1}^{r_j} S((i-1)r_j + h, j-1)$ only if $X \neq 0$. However, no error will result from letting the definition in its present form. Indeed, if $X = 0$, then also $N(i, j) = 0$ so that the inequality in (6) is satisfied in this case, thus $S(i, j)$ will anyway be set to 0. Note also that in case of equality in (6), we execute a pruning action although a priori there is no gain. However, since the number of 1 bits in v_j is thereby reduced, this may enable further prunings in higher levels, which otherwise might not have been done.

We now further compress the list L (of indices of 1 bits which were "pruned" from the tree) using POM, which can be adapted to the compression of a list of d-bit numbers: we choose an integer $c < d - 1$ as parameter, and form a bitmap v of $k = \lceil l_0/2^c \rceil$ bits, where bit i, for $0 \leq i < k$, is set to 1 if and only if the integer i occurs in the $d - c$ leftmost bits of at least one number in L. Thus a 1 bit in position i of v indicates that there are one or more numbers in L in the range $[i2^c, (i+1)2^c - 1]$. For each 1 bit in v, the numbers of the corresponding range can now be stored as relative indices in that range, using only c bits for each, and an additional bit per index serving as flag, which identifies the last index of each range. Further compression of the list L is thus worthwhile only if

$$d \cdot |L| > k + (c+1)|L|. \tag{7}$$

The left-hand side of (7) corresponds to the number of bits needed to keep the list L uncompressed. Therefore this secondary compression is justified only when the number of elements in L exceeds $k/(d - c - 1)$.

For example, for $l_0 = 128$ and $c = 5$, there are 4 blocks of 2^5 bits each; suppose the numbers in L are 36, 50, 62, 105, and 116 (at least five elements are necessary to justify further compression). Then there are three elements in the second block, with relative indices 4, 18, and 30, and there are two elements in the fourth block, with relative indices 9 and 20, the two other blocks being empty. This is shown in Fig. 21.

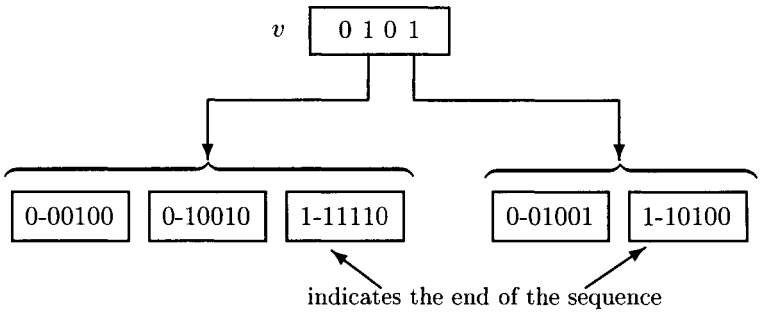


FIGURE 21 Further compression of index list.

Finally we get even better compression by adapting the cutoff condition (6) dynamically to the number of elements in L. During the construction of the tree, we keep track of this number and as soon as it exceeds $k/(d-c-1)$, i.e., it is worthwhile to further compress the list, we can relax the condition in (6) to

$$(c+1) \cdot N(i, j) \leq S(i, j), \tag{8}$$

since any index that will be added to L will use only $c+1$ bits for its encoding.

In fact, after recognizing that L will be compressed, we should check again the blocks already handled, since a subtree $T(i, j)$ may satisfy (8) without satisfying (6). Nevertheless, we have preferred to keep the simplicity of the algorithm and not to check again previously handled blocks, even at the price of losing some of the compression efficiency. Often, there will be no such loss, since if we are at the top level when $|L|$ becomes large enough to satisfy (7), this means that the vector v_0 will be kept in its entirety as a list. If we are not at the top level, say at the root of $T(i, j)$ for $j < t$, then all the previously handled trees will be reconsidered as part of larger trees, which are rooted on the next higher level. Hence it is possible that the subtree $T(i, j)$, which satisfies (8) but not (6) (and thus was not pruned at level j), will be removed as part of a larger subtree rooted at level $j+1$.

2. Combining Huffman and Run-Length Coding

As we are interested in sparse bit strings, we can assume that the probability p of a block of k consecutive bits being zero is high. If $p \geq 0.5$, method NORUN assigns to this 0 block a codeword of length one bit, so we can never expect a better compression factor than k. On the other hand, k cannot be too large since we must generate codewords for 2^k different blocks.

In order to get a better compression, we extend the idea of method NORUN in the following way: there will be codewords for the $2^k - 1$ nonzero blocks of length k, plus some additional codewords representing runs of zero-blocks of different lengths. In the sequel, we use the term "run" to designate a run of zero-blocks of k bits each.

The length (number of k-bit blocks) of a run can take any value up to l_0/k, so it is impractical to generate a codeword for each: as was just pointed out, k

cannot be very large, but l_0 is large for applications of practical importance. On the other hand, using a fixed-length code for the run length would be wasteful since this code must suffice for the maximal length, while most of the runs are short. The following methods attempt to overcome these difficulties.

Starting with a fixed-length code for the run lengths, we like to get rid of the leading zeros in the binary representation $B(\ell)$ of run length ℓ, but we clearly cannot simply omit them, since this would lead to ambiguities. We *can* omit the leading zeros if we have additional information such as the position of the leftmost 1 in $B(\ell)$. Hence, partition the possible lengths into classes C_i, containing run lengths ℓ that satisfy $2^{i-1} \leq \ell < 2^i, i = 1, \ldots, \lfloor log_2(l_0/k) \rfloor$. The $2^k - 1$ nonzero block-patterns and the classes C_i are assigned Huffman codewords corresponding to the frequency of their occurrence in the file; a run of length ℓ belonging to class C_i is encoded by the codeword for C_i, followed by $i - 1$ bits representing the number $\ell - 2^{i-1}$. For example, a run of 77 0-blocks is assigned the codeword for C_7 followed by the 6 bits 001101. Note that a run consisting of a single 0-block is encoded by the codeword for C_1, without being followed by any supplementary bits.

The Huffman decoding procedure must be modified in the following way: The table contains for every codeword the corresponding class C_i as well as $i - 1$. Then, when the codeword that corresponds to class C_i is identified, the next $i - 1$ bits are considered as the binary representation of an integer m. The codeword for C_i followed by those $i - 1$ bits represent together a run of length $m + 2^{i-1}$; the decoding according to Huffman's procedure resumes at the ith bit following the codeword for C_i. Summarizing, we in fact encode the length of the binary representation of the length of a run, and the method is henceforth called `LLRUN`.

Method `LLRUN` seems to be efficient since the number of bits in the binary representation of integers is reduced to a minimum, and the lengths of the codewords are optimized by Huffman's algorithm. However, encoding and decoding are admittedly complicated and thus time consuming. We therefore propose other methods for which the encoded file will consist only of codewords, each representing a certain string of bits. Even if their compression factor is lower than `LLRUN`'s, these methods are justified by their simpler processing.

To the $2^k - 1$ codewords for nonzero blocks, a set S of t codewords is adjoined representing $h_0, h_1, \ldots, h_{t-1}$ consecutive 0 blocks. Any run of zero-blocks will now be encoded by a suitable linear combination of some of these codes. The number t depends on the numeration system according to which we choose the h_i's and on the maximal run length M, but should be low compared to 2^k. Thus in comparison with method `NORUN`, the table used for compressing and decoding should only slightly increase in size, but long runs are handled more efficiently. The encoding algorithm now becomes:

Step 1. Collect statistics on the distribution of run lengths and on the set NZ of the $2^k - 1$ possible nonzero blocks. The total number of occurrences of these blocks is denoted by N_0 and is fixed for a given set of bitmaps.

Step 2. Decompose the integers representing the run lengths in the numeration system with set S of "basis" elements; denote by TNO(S) the total number of occurrences of the elements of S.

Step 3. Evaluate the relative frequency of the appearance of the $2^k - 1 + t$ elements of NZ $\cup$ S and assign a Huffman code accordingly.

For any $x \in (\mathrm{NZ} \cup S)$, let $p(x)$ be the probability of the occurrence of x and $\ell(x)$ the length (in bits) of the codeword assigned to x by the Huffman algorithm. The weighted average length of a codeword is then given by $\mathrm{AL}(S) = \sum_{x \in (\mathrm{NZ} \cup S)} p(x)\ell(x)$ and the size of the compressed file is

$$\mathrm{AL}(S) \times (N_0 + \mathrm{TNO}(S)).$$

After fixing k so as to allow easy processing of k-bit blocks, the only parameter in the algorithm is the set S. In what follows, we propose several possible choices for the set $S = \{1 = h_0 < h_1 < \ldots < h_{t-1}\}$. To overcome coding problems, the h_i and the bounds on the associated digits a_i should be so that there is a unique representation of the form $L = \sum_i a_i h_i$ for every natural number L.

Given such a set S, the representation of an integer L is obtained by the simple procedure

for $i \leftarrow t - 1$ **to** 0 **by** -1
  $a_i \leftarrow \lfloor L/h_i \rfloor$
  $L \leftarrow L - a_i \times h_i$
end

The digit a_i is the number of times the codeword for h_i is repeated. This algorithm produces a representation $L = \sum_{i=0}^{t-1} a_i h_i$, which satisfies

$$\sum_{i=0}^{j} a_i h_i < h_{j+1} \qquad \text{for } j = 0, \ldots, t-1. \tag{9}$$

Condition (9) guarantees uniqueness of representation (see [51]).

A natural choice for S is the standard binary system (method POW2), $h_i = 2^i, i \geq 0$, or higher base numeration systems such as $h_i = m^i, i \geq 0$ for some $m > 2$. If the run length is L, it will be expressed as $L = \sum_i a_i m^i$, with $0 \leq a_i < m$ and if $a_i > 0$, the codeword for m^i will be repeated a_i times. Higher base systems can be motivated by the following reason.

If p is the probability that a k-bit block consists only of zeros, then the probability of a run of r blocks is roughly $p^r(1 - p)$; i.e., the run lengths have approximately geometric distribution. The distribution is not exactly geometric since the involved events (some adjacent blocks contain only zeros; i.e., a certain word does not appear in some consecutive documents) are not independent. Nevertheless the experiments showed that the number of runs of a given length is an exponentially decreasing function of run length. Hence with increasing base of the numeration systems, the relative weight of the h_i for small i will rise, which yields a less uniform distribution for the elements of NZ $\cup$ S calculated in Step 3. This has a tendency to improve the compression obtained by the Huffman codes. Therefore passing to higher-order numeration systems will reduce the value of $\mathrm{AL}(S)$.

On the other hand, when numeration systems to base m are used, $\mathrm{TNO}(S)$ is an increasing function of m. Define r by $m^r \leq M < m^{r+1}$ so that at most r m-ary digits are required to express a run length. If the lengths are uniformly distributed, the average number of basis elements needed (counting multiplicities)

is proportional to $(m-1)r = (m-1)\log_m M$, which is increasing for $m > 1$, and this was also the case for our nearly geometric distribution. Thus from this point of view, lower base numeration systems are preferable.

As an attempt to reduce TNO(S), we pass to numeration systems with special properties, such as systems based on Fibonacci numbers

$$F_0 = 0, \quad F_1 = 1, \qquad F_i = F_{i-1} + F_{i-2} \quad \text{for} \quad i \geq 2.$$

(a) The binary Fibonacci numeration system (method FIB2): $h_i = F_{i+2}$. Any integer L can be expressed as $L = \sum_{i \geq 0} b_i F_{i+2}$ with $b_i = 0$ or 1, such that this binary representation of L consisting of the string of b_i's contains no adjacent 1's. This fact for a binary Fibonacci system is equivalent to condition (9), and reduces the number of codewords we need to represent a specific run length, even though the number of added codewords is larger than for POW2 (instead of $t(\text{POW2}) = \lfloor \log_2 M \rfloor$ we have $t(\text{FIB2}) = \lfloor \log_\phi(\sqrt{5}M) \rfloor - 1$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio). For example, when all the run lengths are equally probable, the average number of codewords per run is asymptotically (as $k \to \infty$) $\frac{1}{2}(1 - 1/\sqrt{5})t(\text{FIB2})$ instead of $\frac{1}{2}t(\text{POW2})$.

(b) A ternary Fibonacci numeration system: $h_i = F_{2(i+1)}$, i.e., we use only Fibonacci numbers with even indices. This system has the property that there is at least one 0 between any two 2's. This fact for a ternary Fibonacci system is again equivalent to (9).

VI. FINAL REMARKS

Modern information retrieval systems are generally based on inverted files and require large amounts of storage space and powerful machines for the processing of sophisticated queries. Data compression techniques that are specifically adapted to the various files in an IR environment can improve the performance, both by reducing the space needed to store the numerous auxiliary files and by reducing the necessary data transfer and thereby achieving a speedup.

We have presented a selected choice of techniques pertaining to the different files involved in a full-text IR system; some are given with considerable detail, others are only roughly described. We hope that, nevertheless, the reader will get a useful overall picture, which can be completed by means of the appended literature.

Our main focus has been on IR systems using inverted files. With the development of ever more powerful computers, it may well be that brute force methods, like searching large files using some pattern matching techniques (see, e.g., [52]) or probabilistic approaches using signature files (see, e.g., [53]), will again be considered a feasible alternative, even for very large files.

REFERENCES

1. Pennebaker, W. B., and Mitchell, J. L. *JPEG: Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1993.
2. Bratley, P., and Choueka, Y. *Inform. Process. Manage.* 18:257–266, 1982.

3. Attar, R., Choueka, Y., Dershowitz, N., and Fraenkel, A. S. *J. Assoc. Comput. Mach.* **25**: 52–66, 1978.
4. Bookstein, A., and Klein, S. T. *Inform. Process. Manage.* **26**:525–533, 1990.
5. Davis, D. R., and Lin, A. D. *Commun. ACM* **8**:243–246, 1965.
6. Choueka, Y., Fraenkel, A. S., Klein, S. T., and Segal, E. In *Proc. 10th ACM-SIGIR Conf.*, New Orleans, pp. 306–315, 1987.
7. Bookstein, A., Klein, S. T., and Ziff, D. A. *Inform. Process. Manage.* **28**:795–806, 1992.
8. Fraenkel, A. S. *Jurimetrics J.* **16**:149–156, 1976.
9. Even, S. *IEEE Trans. Inform. Theory* **9**:109–112, 1963.
10. McMillan, B. *IRE Trans. Inform. Theory* **2**:115–116, 1956.
11. Even, S. *Graph Algorithms.* Comput. Sci. Press, New York, 1979.
12. Shannon, C. E. *Bell System Tech. J.* **27**:379–423, 623–656, 1948.
13. Huffman, D. *Proc. of the IRE* **40**:1098–1101, 1952.
14. Van Leeuwen, J. In *Proc. 3rd ICALP Conference*, pp. 382–410. Edinburgh Univ. Press, Edinburgh, 1976.
15. Fraenkel, A. S., and Klein, S. T. *Combinatorial Algorithms on Words*, NATO ASI Series Vol. F12, pp. 169–183. Springer-Verlag, Berlin, 1985.
16. Knuth, D. E. *The Art of Computer Programming*, Vol. III, *Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.
17. Knuth, D. E. *The Art of Computer Programming*, Vol. I, *Fundamental Algorithms*, Addison-Wesley, Reading, MA, 1973.
18. Larmore, L. L., and Hirschberg, D. S. *J. Assoc. Comput. Mach.* **37**:464–473, 1990.
19. Fraenkel, A. S., and Klein, S. T. *Computer J.* **36**:668–678, 1993.
20. Reghbati, H. K. *Computer* **14**(4):71–76, 1981.
21. Bookstein, A., and Klein, S. T. *ACM Trans. Inform. Systems* **8**:27–49, 1990.
22. Moffat, A., Turpin, A., and Katajainen, J. *Proc. Data Compression Conference DCC-95*, Snowbird, UT, pp. 192–201, 1995.
23. Schwartz, E. S., and Kallick, B. *Commun. ACM* **7**:166–169, 1964.
24. Hirschberg, D. S., and Lelewer, D. A. *Commun. ACM* **33**:449–459, 1990.
25. Witten, I. H., Moffat, A., and Bell, T. C. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, New York, 1994.
26. Moffat, A., Turpin, A. In *Proc. Data Compression Conference DCC-96*, Snowbird, UT, pp. 182–191, 1996.
27. Gilbert, E. N., and Moore, E. F. *Bell System Tech. J.* **38**:933–968, 1959.
28. Bookstein, A., and Klein, S. T. *Computing* **50**:279–296, 1993.
29. Zipf, G. K. *The Psycho-Biology of Language.* Houghton, Boston, 1935.
30. Katona, G. H. O., and Nemetz, T. O. H. *IEEE Trans. Inform. Theory* **11**:284–292, 1965.
31. Moffat, A., Zobel, J., and Sharman, N. *IEEE Trans. Knowledge Data Engrg.* **9**:302–313, 1997.
32. Rissanen, J. *IBM J. Res. Dev.* **20**:198–203, 1976.
33. Rissanen, J., and Langdon, G. G. *IBM J. Res. Dev.* **23**:149–162, 1979.
34. Witten, I. H., Neal, R. M., and Cleary, J. G. *Commun. ACM* **30**:520–540, 1987.
35. Fraenkel, A. S., Mor, M., Perl, Y. *Acta Inform.* **20**:371–389, 1983.
36. Fraenkel, A. S., Mor, M., and Perl, Y. In *Proc. 19th Allerton Conf. on Communication, Control and Computing*, pp. 762–768, 1981.
37. Ziv, J., and Lempel, A. *IEEE Trans. Inform. Theory* **23**:337–343, 1977.
38. Ziv, J., and Lempel, A. *IEEE Trans. Inform. Theory* **24**:530–536, 1978.
39. Whiting, D. L., George, G. A., and Ivey, G. E. U. S. Patent 5,126,739 (1992).
40. Fraenkel, A. S., and Mor, M. *Computer J.* **26**:336–343, 1983.
41. Hamming, R. W. *Coding and Information Theory.* Prentice-Hall, Englewood Cliffs, NJ, 1980.
42. Witten, I. H., Bell, T. C., and Nevill, C. G. In *Proc. Data Compression Conference*, Snowbird, UT, pp. 23–32, 1991.
43. Bookstein, A., Klein, S. T., and Raita, T. *ACM Trans. Inform. Systems* **15**:254–290, 1997.
44. Choueka, Y., Fraenkel, A. S., Klein, S. T., and Segal, E. In *Proc. 9th ACM-SIGIR Conf.*, Pisa, pp. 88–96. ACM, Baltimore, MD, 1986.
45. Schuegraf, E. J. *Inform. Process. Manage.* **12**:377–384, 1976.
46. Teuhola, J. *Inform. Process. Lett.* **7**:308–311, 1978.
47. Jakobsson, M. *Inform. Process. Lett.* **7**:304–307, 1978.

48. Wedekind, H., and Härder, T. *Datenbanksysteme II*. Wissenschaftsverlag, Mannheim, 1976.
49. Vallarino, O. Special Issue, *SIGPLAN Notices*, 2:108–114, 1976.
50. Jakobsson, M. *Inform. Process. Lett.* 14:147–149, 1982.
51. Fraenkel, A. S. *Amer. Math. Monthly* 92:105–114, 1985.
52. Boyer, R. S., and Moore, J. S. *Commun. ACM* 20:762–772, 1977.
53. Faloutsos, C., and Christodulakis, S. *ACM Trans. Office Inform. Systems* 2:267–288, 1984.

This Page Intentionally Left Blank

DATABASE AND DATA COMMUNICATION NETWORK SYSTEMS

Techniques and Applications

VOLUME 3

This Page Intentionally Left Blank

DATABASE AND DATA COMMUNICATION NETWORK SYSTEMS

Techniques and Applications

VOLUME 3

Edited by

Cornelius T. Leondes

Professor Emeritus
University of California
Los Angeles, California

ACADEMIC PRESS

An imprint of Elsevier Science

Amsterdam Boston London New York Oxford Paris
San Diego San Francisco Singapore Sydney Tokyo

Front cover: Digital Image © 2002 PhotoDisc.

This book is printed on acid-free paper. ♾

Academic Press
An imprint of Elsevier Science
525 B Street, Suite 1900, San Diego, California 92101-4495, USA
http://www.academicpress.com

Academic Press
84 Theobalds Road, London WC1X 8RR, UK
http://www.academicpress.com

Library of Congress Catalog Card Number: 20016576

International Standard Book Number: 0-12-443895-4 (set)
International Standard Book Number: 0-12-443896-2 (volume 1)
International Standard Book Number: 0-12-443897-0 (volume 2)
International Standard Book Number: 0-12-443898-9 (volume 3)

PRINTED IN THE UNITED STATES OF AMERICA
02 03 04 05 06 07 MM 9 8 7 6 5 4 3 2 1

CONTENTS

CONTENTS OF VOLUME 3

CONTENTS OF VOLUME I

CONTENTS OF VOLUME 2

CONTRIBUTORS

Numbers in parentheses indicate the pages on which the author's contributions begin.

C. Apostolas (859) Network Communications Laboratory, Department of Informatics, University of Athens, Panepistimioupolis, Athens 15784, Greece

Bernardino Arcay (825) Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Gill Barequet (277) The Technion—Israel Institute of Technology, Haifa 32000, Israel

József Békési (233) Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

Elisa Bertino (175) Dipartimento di Scienze dell'Informazione, Università di Milano, 20135 Milano, Italy

Hsin-Horng Chen (203) Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China

Shyi-Ming Chen (203) Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan, Republic of China

Jian Chen (663) Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Carlos Dafonte (825) Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain

Timon C. Du (1) Department of Industrial Engineering, Chung Yuan Christian University, Chung Li, Taiwan 32023; and Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Shatin, NT Hong Kong

Elena Ferrari (175) Dipartimento di Chimica, Fisica e Matematica, Università dell'Insubria - Como, Italy

Gábor Galambos (233) Department of Informatics, Teacher's Training College, University of Szeged, Szeged H-6701, Hungary

Jacob Gan (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

Mario Gerla (693) Computer Science Department, University of California—Los Angeles, Los Angeles, California 90095

Stathes Hadjiefthymiades (635) Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

M. Ahsan Akhtar Hasin[1] (417) Industrial Systems Engineering, Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

Li Hong-Yi (901) Advanced Wireless Networks, Nortel Research, Nepean, Ontario, Canada K2G 6J8

Hiroshi Ishikawa (77) Department of Electronics and Information Engineering, Tokyo Metropolitan University, Tokyo 192-0397, Japan

Hiroo Iwata (549) Institute of Engineering Mechanics and Systems, University of Tsukuba, Tsukuba 305-8573, Japan

Rong-Hong Jan (719) Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

Chua Chee Kai (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

Myoung Ho Kim (41) Department of Computer Science, Korea Advanced Institute of Science and Technology, Taejon 305-701, Korea

Wataru Kishimoto (783) Department of Information and Image Sciences, Chiba University, Chiba 263-8522, Japan

Shmuel Tomi Klein (573) Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel

Marco Lazzari[2] (751) ISMES, Via Pastrengo 9, 24068 Seriate BG, Italy

Doheon Lee (41) Department of BioSystems, Korea Advanced Institute of Science and Technology, Daejon, Republic of Korea

Chiang Lee (123) Institute of Information Engineering, National Cheng-Kung University, Tainan, Taiwan, Republic of China

[1] Current address: Industrial and Production Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh

[2] Current address: Dipartimento de Scienze della Formazione, Università di Bergamo, Bergamo 24029, Italy

Mong Li Lee (497) School of Computing, National University of Singapore, Singapore 117543

Kunwoo Lee (517) School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea

Ngoh Lek-Heng (901) SingAREN, Kent Ridge Digital Labs, Singapore 119613

Jason Lowder (663) School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

Drakoulis Martakos (635) Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

Saverio Mascolo (693) Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 70125 Bari, Italy

Tong Mei (367) Gintic Institute of Manufacturing Technology, Singapore 638075

Troy Milner (663) School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

P. C. Pandey (417) Asian Institute of Technology, Klong Luang, Pathumthani 12120, Thailand

Sangkun Park (517) Institute of Advanced Machinery and Design, Seoul National University, Seoul 151-742, Korea

Samuel Pierre (289) Mobile Computing and Networking Research Laboratory (LARIM); and Department of Computer Engineering, École Polytechnique de Montréal, Montréal, Quebec, Canada H3C 3A7

Sameer Pradhan (663) Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Paolo Salvaneschi (751) ISMES, Via Pastrengo 9, 24068 Seriate BG, Italy

G. Sfikas[3] (859) Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

Timothy K. Shih (327) Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 25137, Republic of China

José A. Taboada (825) Department of Electronics and Computer Science, Universidade de Santiago de Compostela, 15782, Santiago de Compostela (A Coruña), Spain

R Tafazolli (859) Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

I-Fei Tsai (719) Wistron Corporation, Taipei 221, Taiwan

[3]Current address: Lucent Technologies, Optimus, Windmill Hill Business Park, Swindon, Wiltshire SN5 6PP, England

Irene M. Y. Woon (497) School of Computing, National University of Singapore, Singapore 117543

Xindong Wu (663) Department of Computer Science, University of Vermont, Burlington, Vermont 05405

Du Zhaohui (367) School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

17

INFORMATION DATA ACQUISITION ON THE WORLD WIDE WEB DURING HEAVY CLIENT/SERVER TRAFFIC PERIODS

STATHES HADJIEFTHYMIADES
DRAKOULIS MARTAKOS

Department of Informatics and Telecommunications, University of Athens, Panepistimioupolis, Ilisia, Athens 15784, Greece

I. INTRODUCTION

In the second half of the 1990s, the WWW evolved into dominant software technology and broke several barriers. Despite the fact that it was initially intended for a wide (universal) area network, the WWW today enjoys a tremendous

Database and Data Communication Network Systems, Vol. 3

success and penetration even into corporate environments (intranets). Key players in the business software arena, like Oracle, Informix, and Microsoft, recognize the success of this open platform and constantly adapt their products to it. Originally conceived as a tool for the cooperation between high-energy physicists, this network service is nowadays becoming synonymous with the Internet as it is used vastly by the majority of Internet users even for tasks like e-mail communication.

The universal acceptance of the WWW stimulated the need to provide access to the vast legacy of existing heterogeneous information.[1] Such information ranged from proprietary representation formats to engineering and financial databases, which were, since the introduction of WWW technology, accessed through specialized tools and individually developed applications. The vast majority of the various information sources were stored in Relational DBMS (RDBMS), a technology that enjoys wide acceptance in all kinds of applications and environments. The advent of the Web provided a unique opportunity for accessing such data repositories, through a common front-end interface, in an easy and inexpensive manner. The importance of the synergy of WWW and database technologies is also broadened by the constantly increasing management requirements for Web content ("database systems are often used as high-end Web servers, as Webmasters with a million of pages of content invariably switch to a web site managed by database technology rather than using file system technology," extract from the Asilomar Report on Database Research [4]).

Initial research on the WWW database framework did not address performance issues, but since this area is becoming more and more important, relevant concern is begining to grow. The main topic of this chapter, namely the study of the behavior exhibited by WWW-enabled information systems under a heavy workload, spans a number of very important issues, directly associated with the efficiency of service provision. Gateway specifications (e.g., CGI, ISAPI[2]) are very important due to their use for porting existing systems and applications to the WWW environment. Apart from the gateway specifications used, the architecture of the interface toward the information repository (i.e., RDBMS) is also extremely important (even with the same specifications quite different architectures may exhibit quite different behaviour in terms of performance). The internal architecture of HTTP server software is also an important issue in such considerations. Older single-process architectures are surely slower in dispatching clients' request than newer multithreaded schemes. Prespawned server instances (processes or threads) also play an important role.

Lastly, as part of our study, we are considering how existing WWW-based systems are being measured. We provide a brief description of some of the most important performance evaluation tools.

[1]The significance of this issue triggered the organization, by W3C, of the Workshop on Web Access to Legacy Data, in parallel with the Fourth International WWW Conference, held in Boston, MA in December 1995.

[2]In this chapter, server APIs like ISAPI and NSAPI, will also be referred to as Gateway specifications despite the flexibility they offer for the modification of a basic server functionality, Since we are mostly concerned with interfaces to legacy systems, such features of server APIs will not be considered.

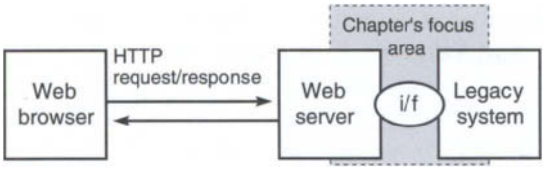


FIGURE 1 Basic architectural assumption: a three-tier scheme.

It should be noted that the entire chapter is focused on the three-tier scenario where access to a legacy information system is always realized through a browser–server combined architecture (i.e., HTTP is always used), as shown in Fig. 1 (as opposed to an applet-based architecture where access to the legacy system can be realized directly, without the intervention of HTTP: a 2-tier scheme).

As illustrated in Fig. 1, we do not consider network-side aspects (e.g., network latency, HTTP). The network infrastructure underneath the Internet is continuously becoming more and more sophisticated with technologies like ATM and xDSL. The best-effort model for Internet communications is being supplemented with the introduction of QoS frameworks like Integrated Services/Differentiated Services. Lastly, on the application layer, HTTP/1.1, Cascading Style Sheets (CSS), and Portable Network Graphics (PNG) show W3C's intention to "alleviate Web slowdown" [24].

II. GATEWAY SPECIFICATIONS

In terms of gateway specifications considerable work has been done in the area of the Common Gateway Specification (CGI) and its comparison with proprietary interfaces like ISAPI and NSAPI. Other specifications like Fast CGI, the Servlet API, and Netscape's, CORBA-based, WAI (Web Application Interface) have also emerged and are extensively used nowadays. In the following paragraphs we will try to set the scene for studying accesses to legacy systems by providing very brief introductions to major gateway specifications.

A. Common Gateway Interface

The CGI is a relatively simple interface mechanism for running external programs (general purpose software, gateways to existing systems) in the context of a WWW information server in a platform independent way. The mechanism has been in use in the WWW since 1993. By that time, CGI appeared in the server developed by the National Center for Supercomputing Applications (NCSA http demon, httpd). The CGI specification is currently in the Internet Draft status [10]. Practically, CGI specifies a protocol for information exchange between the information server and the aforementioned external programs as well as a method for their invocation by the WWW server (clause 5 of the Internet Draft). Data are supplied to the external program by the information

server through environment variables or the "standard input" file descriptor. CGI is a language independent specification. Therefore, CGI programs may be developed in a wide variety of languages like C, C++, Tcl/Tk, Perl, Bourne shell, or even Java. Owing to its simplicity, support for CGI is provided in almost all WWW servers (WWW server independence). A very important issue associated with the deployment of CGI is the execution strategy followed by the controlling WWW server. CGI-based programs (also referred to as scripts) run as short-lived processes separately to the httpd. As such, they are not into position to modify basic server functionality (i.e., logging) or share resources with each other and the httpd. Additionally, they impose considerable resource waste and time overhead due to their one process/request scheme. A more detailed discussion on the pros and cons of the CGI design can be found in Section II. G.

During the evolution of WWW software, key commercial players like Netscape and Microsoft recognized the deficiency of CGI and introduced their own proprietary interfaces for extending basic server functionality. Such mechanisms were named NSAPI (Netscape Server API) and ISAPI (Internet Server API), respectively. NSAPI was supported in the Communications/Commerce generation of servers and, now, in the Fast-Track/Enterprise series. ISAPI first appeared in the initial version of the Internet Information Server (IIS) and has been a standard feature of Microsoft server software ever since then. Other efforts toward the same direction are the FastCGI specification from Open Market Inc., the Java Servlet API from Sun, and the WAI from Netscape Communications.

B. Netscape Server API

The NSAPI specification enables the development of server plug-ins (also called Service Application Functions or SAFs) in C or C++ that are loaded and invoked during different stages of the HTTP request processing. Since server plug-ins run within the Netscape server's process (such plug-ins are initialized and loaded when the server starts up), the plug-in functions can be invoked with little cost (no separate process needs to be spawned as in the case of CGI). Also, NSAPI exposes the server's internal procedure for processing a request. It is, therefore, feasible to develop a server plug-in that is invoked during any of the steps in this procedure. These steps are briefly presented below:

- ***AuthTrans*** (authorization translation): verification of authorization information included in the request.
- ***NameTrans*** (name translation): mapping of the request URI into a local file system path.
- ***PathCheck*** (path checking): verification that the local file system path is valid and that the requesting user is authorized to access it.
- ***ObjectType*** (object typing): determination of the MIME type of the requested resource (e.g., text/html, image/gif).
- ***Service*** (service): the response is returned to the client.
- ***AddLog*** (adding log entries): server's log file is updated.

The server executes SAFs in response to incoming requests through a MIME mapping approach. Should the incoming request pertain to a hypothetical MIME type (also called internal server type), the associated SAF code is invoked to handle the request. Such mapping is maintained in a basic configuration file. Each SAF has its own settings and has access to the request information and any other server variables created or modified by previous SAFs. The SAF performs its operation based on all this information. It may examine, modify, or create server variables based on the current request and its purpose within its step.

C. Web Application Interface

WAI [32] is one of the programming interfaces, provided in the latest Netscape servers, that allows the extension of their functionality. WAI is a CORBA-based programming interface that defines object interfaces to the HTTP request/response data as well as server information. WAI compliant applications can be developed in C, C++, or Java (JDK 1.1). WAI applications accept HTTP requests from clients, process them, and, lastly generate the appropriate responses. Server plug-ins may also be developed using WAI. Netscape claims that WAI accomplishes considerable performance improvements when compared to CGI. Since WAI-compliant modules (or WAI services) are persistent (in contrast to CGI programs), response times are reduced thus, improving performance. Additionally, WAI modules are multithreaded so the creation of additional processes is unnecessary.

Specifically, WAI allows accessing the HTTP request headers, accessing server internal information, reading data transmitted from the browser (i.e., the content of a POST request triggered by the submission of an HTML form), setting the headers and the status in the HTTP response, redirecting the interacting client to another location, or composing the response that will be sent to the client (e.g., an HTML page). WAI modules may either be applications that run outside the server process in the same or different machines (the CORBA ORB deals with all the details) or be in the form of server plug-ins that run within the server's process. Server plug-ins can be shared libraries or dynamic link libraries.

D. Internet Server API

ISAPI [29,40] is an interface to WWW servers that allows the extension of their basic functionality in an efficient way. ISAPI compliant modules run in Windows 9x/NT environments and have the form of dynamic link libraries (DLL). ISAPI DLLs can be loaded and called by a WWW server and provide a functionality similar to that of CGI applications (such ISAPI DLLs are called extensions). The competitive advantage of ISAPI over CGI is that ISAPI code runs in the same address space as the WWW server (i.e., in-process) and has access to all the resources available to the server (thus, the danger of a server crash due to error-prone extension code is not negligible). Additionally, ISAPI extensions, due to their multithreaded orientation, have lower overhead than CGI applications because they do not require the creation of additional processes upon reception of new requests (the creation of a new thread by the server is much faster) and do not perform time-consuming communications

across process boundaries, ISAPI DLLs may be unloaded if the memory is needed by another process. Interaction between the ISAPI extension code and the WWW server is performed through a memory structure called Extension Control Block (ECB). Each request addressed to the ISAPI extension causes a new ECB structure to be allocated and filled with information such as the QUERY_STRING parameter [10] encountered in conventional CGI applications. Filters are another type of ISAPI extensions. Filters are loaded once, upon server's start-up, and invoked for each incoming request. Practically, ISAPI filters allow the customization of the flow of data within the server process. ISAPI is now used by several Web servers including Microsoft, Process Software, and Spyglass.

E. FastCGI

FastCGI [6] is basically an effort to provide a "new implementation of CGI" [7] that would enjoy the portability of its predecessor while overcoming its performance handicap. FastCGI processes (gateway instances according to the terminology in this chapter) are persistent in the sense that after dispatching some request remain memory resident (and do not exit, as conventional CGI dictates), awaiting for another request to arrive. Instead of using environment variables, standard input and output. FastCGI communicates with the server process (similarly to CGI it runs as a separate process) through a full duplex Socket connection. Basing the interface on Sockets allows the execution of the gateway instance on a machine different from that of the WWW server (a distributed approach). The information transferred over this full-duplex connection is identical to that exchanged in the CGI case. Thus, migration from classical CGI programs to FastCGI is a fairly simple process. FastCGI supports the language independence of its predecessor (languages like Tcl/Tk and Perl can be used). FastCGI applications can be programmed either in a single-threaded or in a multithreaded way. The FastCGI framework supports a performance enhancement technique called "session affinity." Session affinity allows the server to route incoming requests to specific memory resident copies of FastCGI processes based on information conveyed within requests (e.g., username/password of the authentication scheme). The performance benefit comes from the caching that FastCGI applications are allowed to perform on user-related data.

In a database access scenario presented in [8], the performance of API-based applications in a single-threaded server architecture is slightly better than that accomplished by FastCGI processes (with internal caching/session affinity disabled). In those tests, the connections established toward the database system are not torn down after the execution of individual queries (database connection caching).

F. Servlet Java API

Servlets are protocol- and platform-independent server side components written in Java. Servlets execute within Java-based Web servers (e.g., Java Web Server) or within external "Servlet engines" interfaces to other types of web servers like

Netscape and Apache (such techniques will be discussed below). They are to the server side what applets are to the client side. Servlets can be used in many ways but generally regarded as a replacement to of CGI for the dynamic creation of HTML content. Servlets first appeared with the Java Web Server launched in 1997 [9]. Servlets allow the realization of three-tier schemes where access to a legacy database or another system is accomplished using some specification like Java Database Connectivity (JDBC) or Internet Inter-ORB Protocol (IIOP). Unlike CGI, the Servlet code stays resident after the dispatch of an incoming request. To handle simultaneous requests new threads are spawned instead of processes. A Web server uses the Servlet API, a generic call-back interface to control the behavior of a Servlet (i.e., initialization—termination through the invocation of specific methods). Servlets may be loaded from the local file system or invoked from a remote host much like in the applet case.

As reported in [9], FastCGI and Java Servlets accomplish comparable performances much higher than that of conventional CGI. Specifically, Organic Online, a San Francisco-based Web development company, measured a 3–4 requests/s throughput for FastCGI or Servlets while CGI managed to handle only 0.25 requests/s.

Some concerns about the performance of Servlets in Web servers other than the Java Web Server are raised in [27]. Servlets execute on top of the Java Virtual Machine (JVM). Spawning the JVM is a time- and resource-consuming task (a time of 5 s and a memory requirement of 4 MB is reported for a standard PC configuration). Such a consumption could even compromise the advantage of Servlets over conventional CGI if a different instance of JVM was needed for each individual request reaching the Web server. In the Java Web Server case there is no need to invoke a new instance of JVM since the Servlet can be executed by the JVM running the server module (a "two-tier" scheme). In the Apache case, the Servlet engine is a stand-alone application (called JServ), running independently to the server process. The Web server passes requests to this prespawned engine, which then undertakes their execution (a proxy-like scheme). A quite similar scheme is followed in every other Web server whose implementation is not based in Java. Indicative examples are the latest versions of Microsoft's IIS and Netscape's FastTrack. Popular external Servlet engines include JRun from Live Software, Inc. (http://www.livesoftware.com/) and WAICoolRunner from Gefion Software (http://www.gefionsoftware.com/). WAICoolRunner has been used in some of the experiments documented in subsequent paragraphs.

G. Comparisons between CGI and Server APIs

Gateway specifications are compared in terms of characteristics or the performance they can achieve. A detailed comparative analysis of the characteristics of CGI and server APIs can be found in [12,37].

Specifically, CGI is the most widely deployed mechanism for integrating WWW servers with legacy information systems. However, its design does not match the performance requirements of contemporary applications: CGI applications do not run within the server process. In addition to the performance overhead (new process per request), this implies that CGI applications cannot

modify the behavior of the server's internal operations, such as logging and authorization (see Section B above). Finally, CGI is viewed as a security issue, due to its connection to a user-level shell.

Server APIs can be considered as an efficient alternative to CGI. This is mainly attributed to the fact that server APIs entail a considerable performance increase and load decrease as gateway applications run in or as part of the server processes (parctically the invocation of the gateway module is equivalent to a regular function call[3]) instead of starting a completely new process for each new request, as the CGI specification dictates. Furthermore, through the APIs, the operation of the server process can be customized to the individual needs of each site. The disadvantages of the API solution include the limited portability of the gateway code, which is attributed to the absence of standardization (completely different syntax and command sets) and strong dependence on internal server architecture.

The choice for the programming language in API configurations is extremely restricted if compared to CGI (C or C++ Vs C, C++, Perl, Tcl/Tk, Rexx, Python, and a wide range of other languages). As API-based programs are allowed to modify the basic functionality offered by the Web server, there is always the concern of a buggy code that may lead to crashes.

The two scenarios involve quite different resource requirements (e.g., memory) as discussed in [37]. In the CGI case, the resource needs are proportional to the number of clients simultaneously served. In the API case, resource needs are substantially lower due to the function-call-like implementation of gateways and multithreaded server architecture.

In terms of performance, many evaluation reports have been published during the past years. Such reports clearly show the advantages of server APIs over CGI or other similar mechanisms but also discuss performance differences between various commercial products.

In [19], three UNIX-based Web servers (NCSA, Netscape, and OpenMarket) have been compared using the WebStone benchmark (see Section V. B). Under an extensive load, the NSAPI configuration achieved 119% better throughput[4] than the CGI configuration running in NCSA. In terms of connections per second, the Netscape/NSAPI configuration outperformed NCSA/CGI by 73%. OpenMarket/CGI achieved slightly better performance than the NCSA/CGI combination. Under any other load class, NSAPI outperformed the other configurations under test but the gap was not as extensive as in the high-load case.

In [45], the comparison between server API implementations and CGI shows the ISAPI is around five times faster than CGI (in terms of throughput). NSAPI, on the other hand, is reported only two times faster than CGI. A quite similar ratio is achieved for the connections per second metric. In terms of average response times, ISAPI's performance is reported to be 1/3 that of NSAPI's.

In [44], published by Mindcraft Inc. (the company that purchased WebStone from Silicon Graphics), quite different results are reported for the

[3]Differences arise from the start-up strategies followed: some schemes involve that gateway instances/modules are preloaded to the server while others follow the on-demand invocation.

[4]Throughput measured in MBps.

two APIs. Specifically, NSAPI is shown to outperform, in certain cases, ISAPI by 40%. In other cases, NSAPI achieves an 82% of the best performance exhibited by ISAPI. Such differences, in the measurements included in the two aforementioned reports, have been attributed to certain modifications incorporated into the NSAPI code provided with the WebStone benchmark. No clear results can be obtained by the two performance evaluation tests. Microsoft's IIS (and ISAPI) seem to be better harmonized with the Windows environments while NSAPI is a portable specification (it can be applied to Netscape servers irrespective of the underlying OS).

III. ARCHITECTURES OF RDBMS GATEWAYS

One of the most important application of the WWW platform refers to the integration of database management systems and RDBMS in particular. Many issues are associated with this particular type of WWW applications, namely the generic architecture the stateful operation and performance, which is of prime concern in this chapter.

The first WWW interfaces for relational management systems appeared during the period 1993–1994. The first products began to come into view in 1995 with WebDBC [33] from Nomad (later StormCloud), Cold Fusion [1] from Allaire, and dbWeb [26] from Aspect Software Engineering (later acquired by Microsoft). The importance of such middleware triggered a very rapid pace in the development of this kind of software. Such interfacing tools have become crucial and indispensable components to the Internet as well as enterprise intranets. The first generation of WWW-RDBMS (e.g., WebDBC) interfacing products was mainly intended for the Windows NT platform and capitalized on the ODBC specification to gain access to databases.

Throughout the development of the generic RDBMS interface presented in [14], the authors observed that, during users' queries for information, a considerable amount of time was spent on the establishment of connections toward the relational management system (Informix in our prototype, which was based on the CGI specification). Irrespectively of the efficiency of the adopted gateway mechanism (i.e., the "slow" CGI vs a "fast" server API), establishing a new connection (per request) to the relational management system is a time- and resource-consuming task that should be limited and, if possible, avoided. Of course, the CGI specification is not efficient in terms of execution speed (and, in any case, aggravates the performance of the system) but if portability and compliance to standards do necessitate its adoption, a "clever" solution must be worked out.

In [15] we adopted the scheme of permanent connections toward the database management system [37]. Permanent connections are established by one or more demon processes that reside within the serving host and execute queries on behalf of specific clients (and gateway instances, i.e., CGI scripts). Demons prevent the inefficient establishment of a large number of database connections and the relevant resource waste; the associated cost is incurred by the demon processes (prior to actual hit dispatch–query execution) and not by the gateway instances (e.g., CGI script, ISAPI thread) upon hit dispatch. Thus,

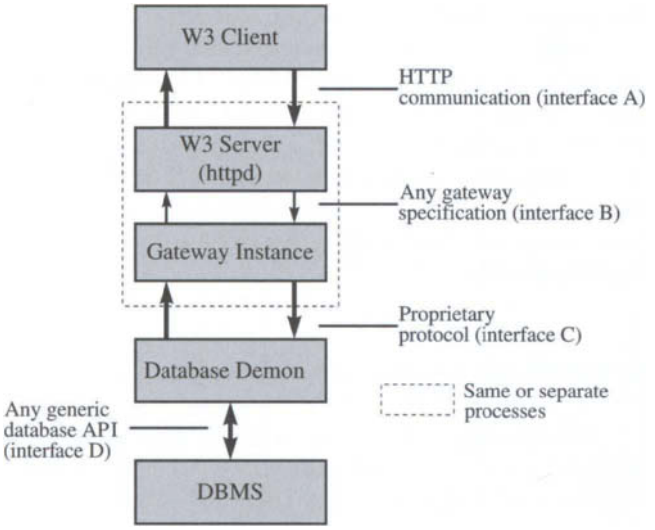


FIGURE 2 Optimized WWW-DBMS interface.

no additional time overhead is perceived by the interacting user in his queries for information. The discussed scheme is shown in Fig. 2.

A. Generic DBMS Interfaces

A very important issue in the demon-based architecture shown in Fig. 2 is the interface toward the DBMS (interface D) as well as the communication mechanism with the gateway instance (interface C in Fig. 2). The demon process should be able to dispatch any kind of queries—SQL statements irrespective of database, table, and field structures. This requirement discourages the adoption of a static interface technique like the embedding of SQL statements in typical 3GLs (i.e., the Embedded SQL API–ISO SQL-92 standard). Instead, a generic interface toward the database should be used.

Contemporary RDBMS offer, as part of the Embedded SQL framework, the Dynamic SQL capability [20,30,35], which allows the execution of any type of statement without prior knowledge of the relevant database schema, table, and field structures (unlike Static SQL, where such information is hard-coded into the respective programs). Dynamic SQL statements can be built at runtime and placed in a string host variable. Subsequently, they are posted to the RDBMS for processing. As the RDBMS needs to generate an access plan at runtime for dynamic SQL statements, dynamic SQL is slower than its static counterpart. This last statement deserves some more discussion: if a generic interface towards a DBMS is pursued, then the execution efficiency of the system could be undermined. Embedded SQL scripts with hard-coded statements execute faster than CGI scripts with dynamic SQL capabilities. The real benefit, in this architecture, comes from the decoupling of the dynamic SQL part (database demon) from the actual gateway instance (Fig. 2).

A second option for generic access to a DBMS is the SQL Call-Level Interface (CLI). The SQL CLI was originally defined by the SQL Access Group (SAG) to provide a unified standard for remote data access. The CLI requires the use of intelligent database drivers that accept a call and translate it into the native database server's access language. The CLI is used by front-end tools to gain access to the RDBMS; the latter should incorporate the appropriate driver. The CLI requires a driver for every database to which it connects. Each driver must be written for a specific server using the server's access methods and network transport stack. The SAG API is based on Dynamic SQL (statements still need to be prepared[5] and then executed). During the fall of 1994, the SAG CLI became an X/Open specification (currently, it is also referred to as X/Open CLI [42]) and later on an ISO international standard (ISO 9075-3 [21]). Practically, the X/Open CLI is a SQL wrapper (a procedural interface to the language): a library of DBMS functions—based on SQL—that can be invoked by an application. As stated in [30], "a CLI interface is similar to Dynamic SQL, in that SQL statements are passed to the RDBMS for processing at runtime, but it differs from Embedded SQL as a whole in that there are no embedded SQL statements and no precompiler is needed."

Microsoft's Open DataBase Connectivity (ODBC) API is based on the X/Open CLI. The 1.0 version of the specification and the relevant Software Development Kit (SDK), launched by Microsoft in 1992, have been extensively criticized for poor performance and limited documentation. Initially, ODBC was confined to the MS Windows platform, but later was ported to other platforms like Sun's Solaris. The ODBC 2.0, which was announced in 1994, has been considerably improved over its predecessor. A 32-bit support contributed to the efficiency of the new generation of ODBC drivers. In 1996, Microsoft announced ODBC 3.0. Nowadays, most database vendors (e.g., Oracle, Informix, Sybase) support the ODBC API in addition to their native SQL APIs. However, a number of problems and hypotheses undermine the future of ODBC technology. ODBC introduces substantial overhead (especially in SQL updates and inserts) due to the extra layers of its architecture (usually, ODBC sits on top of some other vendor-specific middleware like Oracle's SQL*Net). It is a specification entirely controlled by Microsoft. The introduction of the OLE/DB framework gave grounds to the suspicion that Microsoft does not seem committed to progress ODBC any further.

Since the advent of the Java programming language, a new SQL CLI has emerged. It is named JDBC (Java DataBase Connectivity) and was the result of a joint development effort by Javasoft, Sybase, Informix, IBM, and other vendors. JDBC is a portable, object-oriented CLI, written entirely in Java but very similar to ODBC [36]. It allows the development of DBMS-independent Java code that is, at the same time, independent of the executing platform. JDBC's architecture, similar to ODBC's, introduces a driver manager (JDBC driver manager) for controlling individual DBMS drivers. Applications share a common interface with the driver manager. A classification of JDBC drivers

[5]It is requested by the RDBMS to parse, validate, and optimize the involved statement and, subsequently, generate an execution plan for it.

suggests that they are either direct or ODBC-bridged. Specifically, there are four types of JDBC drivers [23]:

- Type 1 refers to the ODBC-bridged architecture and involves the introduction of a translation interface between the JDBC and the ODBC driver. ODBC binary code and the required database client code must be present in the communicating party. Thus, it is not appropriate for applets running in a browser environment.
- Type 2 drivers are based on the native protocols of individual DBMS (i.e., vendor-specific) and were developed using both Java and native code (Java methods invoke C or C++ functions provided by the database vendor).
- Type 3 drivers are exclusively Java based. They use a vendor-neutral protocol to transmit (over TCP/IP sockets) SQL statements to the DBMS, thus necessitating the presence of a conversion interface (middleware) on the side of the DBMS.
- Type 4 drivers are also exclusively based on Java (pure Java driver) but, in contrast to Type 3, use a DBMS-specific protocol (native) to deliver SQL statements to the DBMS.

As discussed in [31], Type 1 drivers are the slowest, owing to the bridging technique. Type 2 drivers are on the extreme other end. Type 3 drivers load quickly (due to their limited size) but do not execute requests are fast as Type 2. Similarly, Type 4 execute quite efficiently but are not comparable to Type 2. Type 1 is generally two times slower than Type 4. In the same article, it is argued that the highest consumption of CPU power in JDBC drivers comes from conversions between different data types and the needed translation between interfaces. JDBC is included, as a standard set of core functions, in the Java Development Kit (JDK) ver. 1.1.

B. Protocols and Interprocess Communication Mechanisms

Another very important issue in the architecture shown in Fig. 2 is the C interface (i.e., the interface between gateway instances and the database demon). As discussed in [15], the definition of interface C involves the adoption of a protocol between the two cooperating entities (i.e., the gateway instance and the database demon) as well as the selection of the proper IPC (Interprocess Communication) mechanism for its implementation.

In the same paper we proposed a simplistic, request/response, client/server protocol for the realization of the interface. The gateway instance (ISAPI/NSAPI thread, CGI process, Servlet, etc.) transmits to the database demon a ClientRequest message, and the database demon responds with a ServerResponse. The ClientRequest message indicates the database to be accessed, the SQL statement to be executed, an identifier of the transmitting entity/instance, and the layout of the anticipated results (results are returned merged with HTML tags). The Backus–Naur Form (BNF) of ClientRequest is:

ClientRequest = DatabaseName SQLStatement [ClientIdentifier] ResultsLayout
DatabaseName = *OCTET
SQLStatement = *OCTET

ClientIdentifier = *DIGIT ; UNIX PID
ResultsLayout = "TABLE" | "PRE" | "OPTION"

ClientIdentifier is the process identifier (integer, long integer) of a CGI script (or thread identifier in the case of a server API) that generated the request. This field is optional, depending on the IPC mechanism used, and could be avoided in a connection-oriented communication (e.g., Sockets). *OCTET denotes a sequence of printable characters and thus, represents a text field. Results are communicated back to the client processes by means of the ServerResponse message. The BNF of ServerResponse is:

ServerResponse = ResponseFragment ContinueFlow
ResponseFragmet = *OCTET
ContinueFlow = "YES" | "NO"

The ResponseFragment (text) field contains the actual information retrieved by the demon process from the designated database. As mentioned, such information is returned to the client, embedded within HTML code. The type of tags used is the one specified in the ResultsLayout field of ClientRequest. The ContinueFlow field is used for optimizing, in certain IPC scenarios (e.g., Message Queues), the transmission of results back to the gateway instance.

The ClientRequest-ServerResponse protocol is very simplistic, and shifts processing from the gateway instance (client) to the database demon (a "thin" client scheme). More advanced protocols (i.e., RDA/DRDA) may be used over the C interface to allow more complicated processing at the side of the gateway instance.

Protocols pertaining to the C interface are deployed using either some IPC mechanism like Message Queues [15,17] or Sockets [18] (low-level middleware) or some kind of middleware like CORBA/IIOP. Message Queues [39] are a quite efficient, message-oriented, IPC mechanism that allows the simultaneous realization of more than one dialog (i.e., various messages, addressed to different processes or threads, can be multiplexed in the same queue). BSD Sockets are ideal for implementation scenarios where the Web server (and, consequently, the gateway instances) and the database demon execute on different hosts (a distributed approach). The advent of the WinSock library for the Microsoft Windows environments rendered Sockets a universal, platform-independent IPC scheme. Message Queues are faster than Sockets but restrict the communication in a single host since they are maintained at the kernel of the operating system. In some recent tests, which are also presented in this chapter, for the realization of a scheme similar to that shown in Fig. 2, we have employed Named Pipes under the Windows NT environment. A Named Pipe is a one or two-way communication mechanism between a server process and one or more client processes executing on the same or different nodes (networked IPC). Under Windows NT, Named Pipes are implemented as a file system and share many characteristics associated with files with respect to use and manipulation.

A type of middleware, used extensively nowadays in many application domains, is the CORBA Object Request Broker (CORBA ORB) [34]. CORBA simplifies the development of distributed applications with components that collaborate reliably, transparently, and in a scaleable way.

The efficiency of CORBA ORB for building interfaces between Java applications is discussed in [36]. It is reported that CORBA performs similarly (and, in some cases, better) to Socket-based implementations while only buffering entails a substantial improvement in Socket communications. Another comparison between several implementations of CORBA ORB (e.g., Orbix, ORBeline) and other types of middleware like Sockets can be found in [13]. In particular, low-level implementations such as Socket-based C modules and C++ wrappers for Sockets significantly outperformed their CORBA or RPC higher-level competitors (tests were performed over high-speed ATM networks—traffic was generated by the TTCP protocol benchmark tool). Differences in performance ranged from 20 to 70% depending on the data types transferred through the middleware (transmission of structures with binary fields has proved considerably "heavier" than scalar data types). A very important competitor of CORBA is DCOM developed by Microsoft. DCOM ships with Microsoft operating systems, thus increasing the impact of this emerging specification.

C. Experiments and Measurements

In this section, we present two series of experiments that allow the quantification of the time overheads imposed by conventional gateway architectures and the benefits that can be obtained by evolved schemes (such as that shown in Fig. 2).

Firstly, we examined the behavior of a Web server setup encompassing a Netscape FastTrack server and Informix Online Dynamic Server (ver. 7.2), both running on a SUN Ultra 30 workstation (processor, SUN Ultra 250 MHz; OS, Solaris 2.6) with 256 MB of RAM. In this testing platform we have evaluated the demon-based architecture of Fig. 2 and the ClientRequest/ServerResponse protocol of the previous Section, using, on the B interface, the CGI and NSAPI specifications (all tested scenarios are shown in Fig. 3) [17].

The IPC mechanism that we have adopted was Message Queues, member of the System V IPC family. Quite similar experiments were also performed with BSD Sockets [18] but are not reported in this section. Both types of gateway instances (i.e., CGI scripts and NSAPI SAFs) as well as the server demon were programmed in the C language. The server demon, for the D interface, used the Dynamic SQL option of Embedded SQL (Informix E/SQL). Only one database demon existed in the setup. Its internal operation is shown in Fig. 4, by means of a flowchart. It is obvious; from Fig. 4, that the database demon operates in a

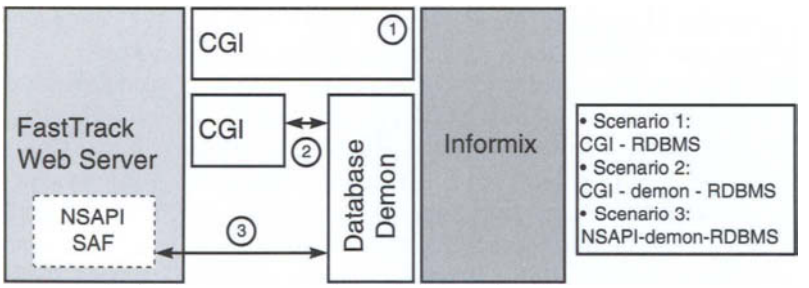


FIGURE 3 Database access scenarios (I).

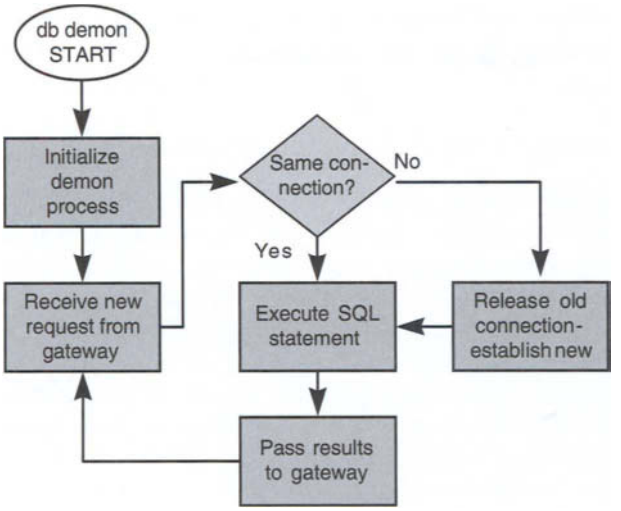


FIGURE 4 Flowchart for database demon operation.

generic way, accessing any of the databases handled by the DBMS and executing any kind of SQL statement. If the, until recently, used DBMS connection (e.g., to a database or specific user account) is the same as the connection needed by the current gateway request, then that connection is being used. Lastly, we should note that incoming requests are dispatched by the demon process in an iterative way.

As shown in Fig. 3, we compared the conventional (monolithic), CGI-based Informix gateway (C and Embedded SQL—Static SQL option) against the CGI ↔ Database Demon ↔ Informix scheme (Scenario 2) and the NSAPI SAF ↔ Database Demon ↔ Informix combined architecture (Scenario 3). In all three cases, the designated database access involved the execution of a complicated query over a sufficiently populated Informix table (around 50,000 rows). The table contained the access log accumulated in a Web server over a period of six months. The layout of the table followed the Common Log Format found in all Web servers, and the row size was 196 bytes. The executed query was the following: Select the IP address and total size of transmitted bytes (grouping by the IP address) from the access log table where the HTTP status code equals 200 (i.e., Document follows). The size of the HTML page produced was 5.605 KB in all scenarios (a realistic page size, considering the published WWW statistics [3,5]). The tuples extracted by the database were embedded in an HTML table (ResultsLayout = "TABLE").

The above described experiments were realized by means of an HTTP pinger, a simple form of the benchmark software to be discussed in Section V. The pinger program was configured to emulate the traffic caused by up to 16 HTTP clients. In each workload level (i.e., number of simultaneous HTTP clients), 100 repetitions of the same request were directed, by the same thread of the benchmark software, to the WWW server. The recorded statistics included:

- *Connect time* (ms): the time required for establishing a connection to the server.

- *Response time* (ms): the time required to complete the data transfer once the connection has been established.
- *Connect rate* (connections/s): the average sustained throughput of the server.
- *Total duration* (s): total duration of the experiment.

The pinger program executed on an MS Windows NT Server (Version 4) hosted by a Pentium II 300-MHz machine with 256 MB of RAM and a PCI Ethernet adapter. Both machines (i.e., the pinger workstation and the web/database server) were interconnected by a 10-MBps LAN and were isolated from any other computer to avoid additional traffic, which could endanger the reliability of the experiments.

From the gathered statistics, we plot, in Fig. 5, the average response time per request. The scatter plot of Fig. 5 is also enriched with polynomial fits. Figure 5 shows that the CGI ↔ demon architecture (Scenario 2) performs systematically better than the monolithic CGI gateway (Scenario 1) and worst than the NSAPI ↔ demon configuration (Scenario 3), irrespective of the number of

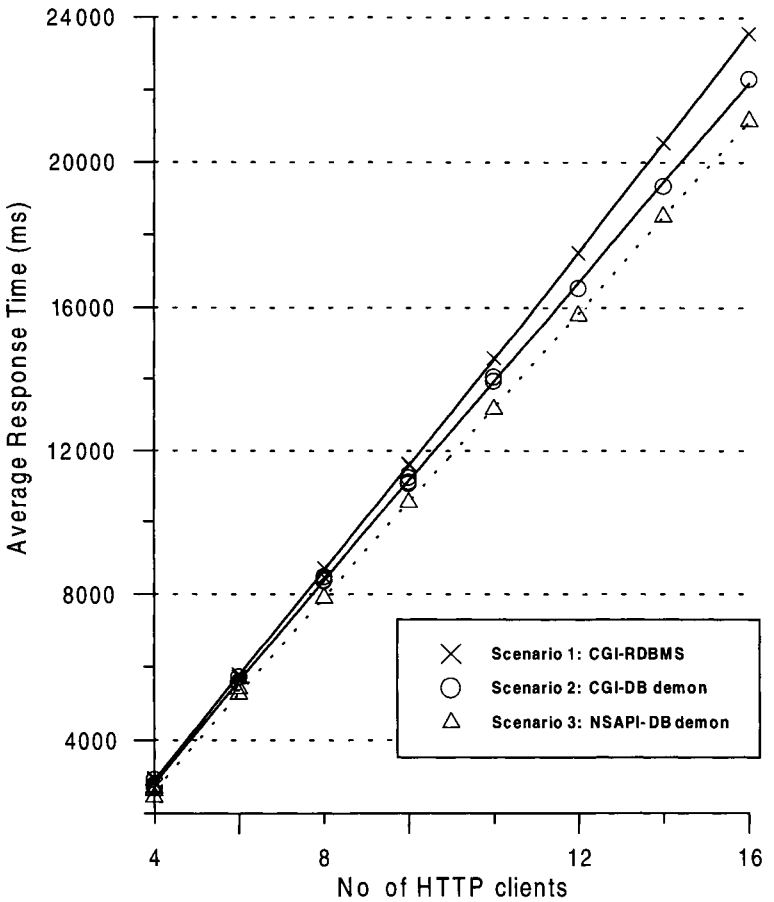


FIGURE 5 Response time vs. number of clients.

HTTP clients (i.e., threads of the pinger program). The performance gap of the three solutions increases proportionally to the number of clients. Figure 5 also suggests that for a relatively small/medium workload (i.e., 16 simultaneous users), the serialization of ClientRequests in Scenarios 2 and 3 (i.e., each ClientRequest incurs a queuing delay due to the iterative nature of the single database demon) does not undermine the performance of the technical option.

A number of additional tests were performed in order to cover even more specifications and gateway architectures. Such tests were performed in the Oracle RDBMS (Ver. 7.3.4) running on a Windows NT Server operating system (Ver. 4). The Web server setup included Microsoft's Internet Information Server (IIS) as well as Netscape's FastTrack Server (not operating simultaneously). Both the Web servers and RDBMS were hosted by a Pentium 133 HP machine with 64 MB of RAM. In this setup we employed, on the B interface (Fig. 2), the CGI, NSAPI, ISAPI, and Servlet specifications, already discussed in previous paragraphs. On the D interface we made use of Embedded SQL (both Static and Dynamic options), ODBC, and JDBC. Apart from conventional, monolithic solutions, we have also evaluated the enhanced, demon-based architecture of Fig. 2. All the access scenarios subject to evaluation are shown in Fig. 6.

The IPC mechanism that we adopted for the demon-based architecture (Scenario 7, Fig. 6) was Named Pipes. All modules were programmed in the C language and compiled using Microsoft's Visual C. Similarly to the previous set of experiments, only one database demon (Dynamic SQL) existed in this setup. The flowchart of its internal operation is identical to that provided in Fig. 4. The database schema and the executed query were also identical to the previous series of experiments.

In this second family of experiments we employed the same HTTP pinger application with the previously discussed trials. It was executed on the same workstation hosting the two Web servers as well as the RDBMS. The pinger was configured to emulate the traffic caused by a single HTTP user. Each experiment consisted of 10 repetitions of the same request transmitted toward the Web server over the TCP loop-back interface. As in the previous case, connect time, response time, connect rate, and total duration were the statistics recorded. Apart from those statistics, the breakdown of the execution time of each gateway instance was also logged. This was accomplished by enriching the

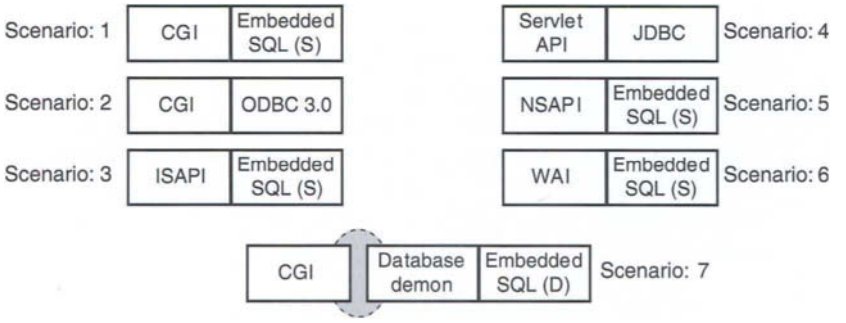


FIGURE 6 Database access scenarios (2).

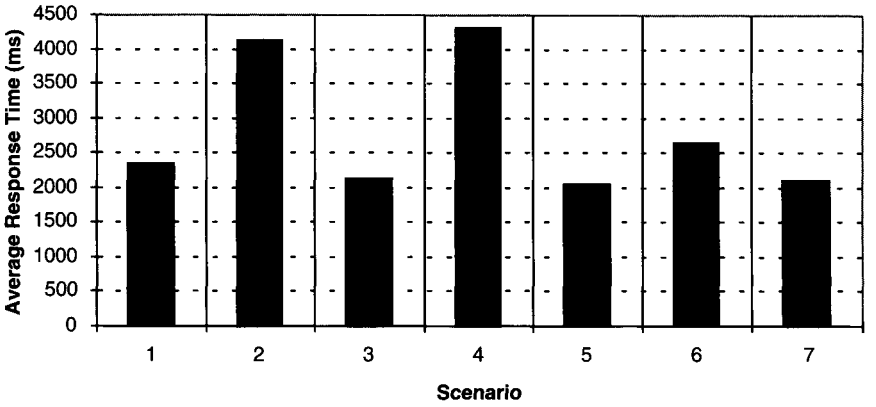


FIGURE 7 Response times for scenarios 1–7.

code of gateway instances with invocations of the C `clock ()` function, which returns the CPU time consumed by the calling process. Thus, we were able to quantify the time needed for establishing a connection to the RDBMS (to be referred to as Tcon) as well as the time needed to retrieve the query results (to be referred to as Tret). Such logging of CPU times was performed in the following scenarios:

- 1 (CGI/Embedded SQL),
- 2 (CGI/ODBC),
- 4 (Servlet/JDBC),[6] and
- 7 (CGI ↔ Database Demon/Dynamic SQL).

We restricted the time breakdown logging in those scenarios since they involve different database access technologies.

Figure 7 plots the average response time for each scenario, and shows that the CGI ↔ Demon architecture accomplishes quite similar times to those of the NSAPI and ISAPI gateways. In the multithreaded gateway specifications (i.e., NSAPI, ISAPI, and WAI), connections toward the RDBMS are established by the executing thread (hence, the associated cost is taken into account). Connections by multithreaded applications are only possible if mechanisms like Oracle's Runtime Contexts [35] or Informix's dormant connections [20] are used. Other multithreaded configurations are also feasible: a database connection (and the associated runtime context) could be preestablished (e.g., by the DLL or the WAI application) and shared among the various threads, but such configurations necessitate the use of synchronization objects like mutexes. In such scenarios a better performance is achieved at the expense of the generality of the gateway instance (i.e., only the initially opened connection may be reused).

In Scenario 2, ODBC Connection Pooling was performed by the ODBC Driver Manager (ODBC 3.0), thus reducing the time overhead associated with connection establishments after the initial request. In Scenario 4, the JDBC

[6] Gefion's WAICoolRunner was used as a Servlet engine in this scenario.

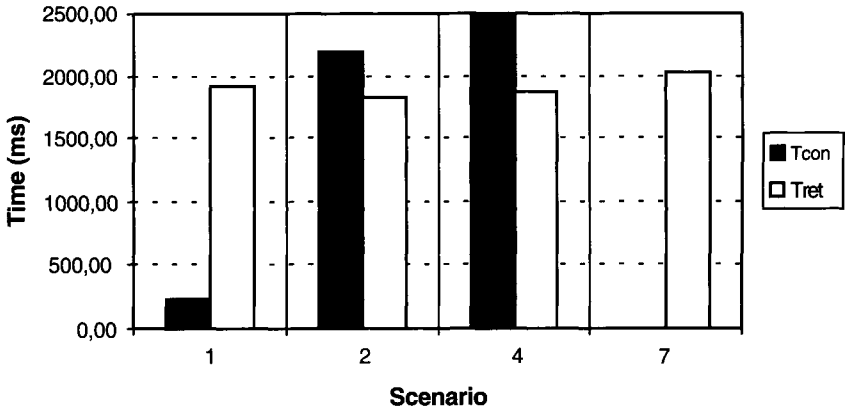


FIGURE 8 Time breakdown for Scenarios 1, 2, 4, and 7.

drivers shipped with Oracle 7.3.4 were used. Specifically, we employed Type 2 drivers (see Section III. A).

In Fig. 8 we show where the factors Tcon and Tret range, and how in the ODBC and JDBC scenarios (i.e., Scenarios 2 and 4) a very important percentage of the execution time of the gateway is consumed for the establishment of connections toward the DBMS. The Embedded SQL scenario (i.e., Scenario 1) achieves the lowest Tcon. The highest Tret is incurred in Scenario 7, where query execution is fully dynamic (see Section III. A for the Dynamic SQL option of Embedded SQL). Tcon is not plotted in Fig. 8 for Scenario 7, as this cost is only incurred once by the database demon.

D. State Management Issues

In general, stateful Web applications impose considerable additional overhead. Extra information is carried across the network to help identify individual sessions and logically correlate user interactions. The IETF RFC for the Cookies mechanism [25] specifies additional fields in the HTTP headers to accommodate state information. Complicated applications, though, may require a large volume of Cookie information to be transmitted to and from the browser. At the server's side, additional processing–handling is required in order to perform state management. In [16] a framework for the adaptation of stateful applications to the stateless Web has been proposed. This framework involved the placement of an extra software layer in the browser-database chain. Such additional, server-side, tiers would associate incoming Cookie values with the appropriate threads of a multithreaded database application, pass execution information, relay results, and assign new Cookie values to the browser in order to preserve a correct sequence of operations. An additional entity responsible for handling state information is also required in the architecture presented in [22]. Considerations concerning the overheads introduced by state management techniques are also discussed in [37].

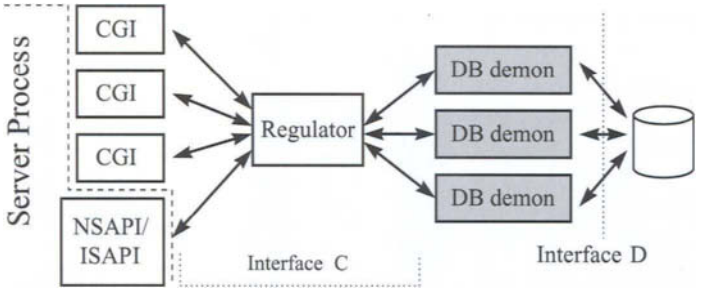


FIGURE 9 Generalized demon-based architecture.

E. Persistent or Nonpersistent Connections to Databases

The database connection persistence problem is mentioned in [37]. A conventional scheme, with monolithic CGIs establishing connections directly toward the RDBMS, could exhaust the available licenses due to the sporadic nature of WWW requests [3,11]. Additionally, as discussed in previous paragraphs, a substantial time overhead is to be incurred in this scheme. Maintaining a database connection per active session (e.g., through a server API or FastCGI) may not be a sound strategy as the maximum licenses limit may also be easily reached while extensive periods of client inactivity cause a terrible waste in resources. In these two scenarios, uncontrolled connection establishment with the RDBMS is, surely, a problem. Persistent database connections may, however, be beneficial for response times as state management is simplified [16].

A demon-based architecture, on the other hand, drastically restricts the number of connections simultaneously established toward the RDBMS (controlled rate of connection establishment). Multiple demons for serving incoming requests in a more efficient way may be established (so as to reduce potential queuing delays caused by the availability of a single dispatching entity [17]). In such scenario a regulating entity will be needed to route requests to idle demons and handle results in the reverse direction (Fig. 9).

F. Template-Based Middleware

Software tools intended for building database-powered WWW sites usually follow a template approach. A generic engine processes preprogrammed templates (in an interpreter-like way) and performs database interactions as needed (run-time binding of reusable code, e.g., some ODBC front-end interface, with a specializing template). Templates practically are the combination of HTML with a proprietary tag set (dealing with query specification, transactions, result set handling, flow control, etc.). An indicative example of a template-based middleware is Allaire's Cold Fusion with its Cold Fusion Markup Language (CFML).[7] Surely, the interpreter-like approach, dictated by the adoption of templates, causes a slowdown in the dispatch of queries. On the other hand,

[7]Formerly known as DBML, DataBase Markup Language.

such tools are extremely efficient programming tools that drastically reduced the time required to develop database gateways in the old-fashioned way using techniques such as Embedded SQL.

IV. WEB SERVER ARCHITECTURES

The architecture of HTTP demons (or WWW servers) have been a very important issue since the advent of the WWW. The concern of the members of the WWW community about the implications of WWW server architecture on performance is clear in works like [28,43]. The very first servers (e.g., NCSA httpd 1.3 and CERN 3.0) were designed to fork out a new server process for each incoming request (i.e., the server clones itself upon arrival of a new request). Newer server architectures adopt the so-called preforking (or pool of processes) approach. The preforking approach involves the provisional generation, by the master server process, of a set of slave processes (the number is customizable). Such processes are sitting idle until an HTTP request reaches the master process. Then the master process accepts the connection and passes the file descriptor to one of the prespawned clients. If the number of simultaneous requests at the server exceeds the number of prespawned clients, then the master httpd process starts forking new instances (similarly to the older techniques). Measurements reported in [28] show that this technique practically doubles the performance of the WWW server. Such measurements were based on one of the very first benchmarking tools for WWW servers, the NCSA pinger.

Newer server architectures were designed in the more efficient, multithreaded paradigm. Notable examples are Netscape's FastTrack/Enterprise servers. In those architectures, upon WWW subsystems' boot, only one server process is spawned. Within this server process the pool of available instances principle is still followed but on the thread level. In other words, a number of threads are prespawned and left idle, awaiting incoming requests to server. If the threads in the thread pool are all in use, the server can create additional threads within the same process/address space to handle pending connections. It is suggested that the number of prespawned server processes should be greater than 1 only in those cases where multiprocessor hardware is used. Servers are optimized on the thread model supported by the underlying OS (e.g., in the HP-UX case where non native threading is supported, a user-level package is provided while in the Solaris case the many-to-many model is adopted).

As discussed in Section II. B, gateways can be built in Netscape servers using the NSAPI specification. Taking into account the server's multithreaded architecture, the performance benefit is twofold: (a) SAF/plug-in code is preloaded in the memory space of the server process, and (b) the server spawns autonomous threads instead of forking processes for each incoming request (Fig. 10). Quite similar behavior is exhibited by Microsoft servers.

The Java Web Server, a product until recently offered by SUN, also adopts the multithreaded architecture discussed above, but relies heavily on the Servlet technology (see Section II. F). A pool of service handler threads (e.g., HTTP handler threads) is provided and bound to a specific ServerSocket. Such handler

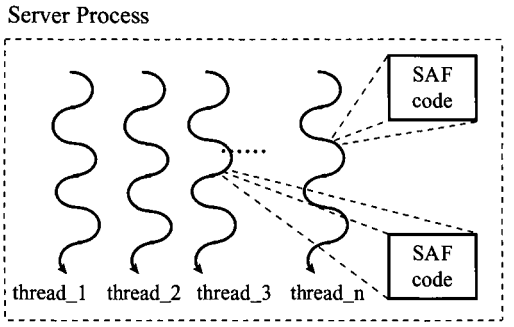


FIGURE 10 Multithreaded Netscape Server architecture.

threads, when associated with an incoming request, know how to dispatch it to HTTP Servlets (after authorization and name translation). Built-in HTTP Servlets are core Servlets that the HTTP service handler threads use to provide standard functionality. Indicative examples are the FileServlet, which is used to respond to simple file requests, and the CGIServlet to provide basic CGI functionality (Fig. 11). A meta-Servlet, called InvokerServlet, is used for loading, invoking, and shutting down Servlets that were explicitly invoked by the client (i.e., using the URL scheme).

V. PERFORMANCE EVALUATION TOOLS

During the past years a number of tools have emerged for the performance evaluation of Web server systems. Some of these tools take the reported Web traffic models into account (analytic workload generation). The vast majority of tools, however, follows the so-called trace-driven approach for workload generation. The trace-driven approach, practically, tries to imitate prerecorded traffic logs either by sampling or by reexecuting traces.

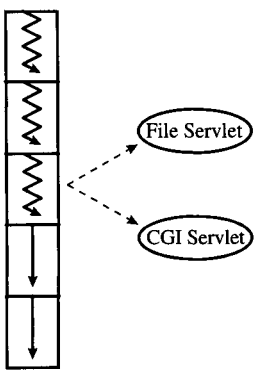


FIGURE 11 HTTP connection handling in JavaServer.

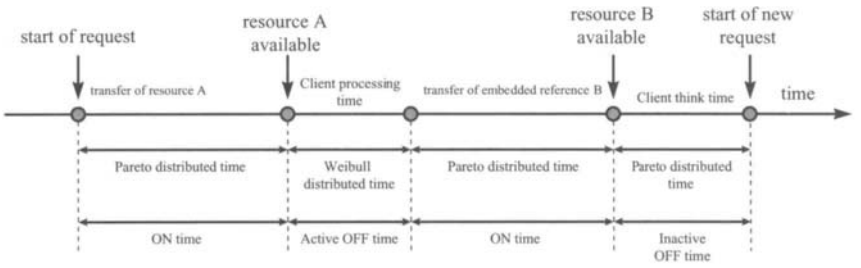


FIGURE 12 Time distributions for WWW traffic.

A. Analytic Workload Generation

Traffic modeling for the WWW has been studied in considerable detail in [11] where a self-similar nature is suggested. Self-similarity is attributed to the multiplexing of a large number of ON/OFF sources (which in the case of the WWW are interacting users). The period spent by the sources in the ON or the OFF states is following heavy-tailed distributions. In particular, ON times (times corresponding to resource transmissions) are best modeled through the Pareto distribution. OFF times are classified either as Active (attributed to the processing time of the Web client, rendering time, etc.) or as Inactive (attributed to user think time). The former are much more extended than the latter. Different distributions seem suitable for those two time components. The Weibull distribution is best suited for Active times [3] while the Pareto distribution best approximates the period of Inactivity. Such time distributions are shown in Fig. 12.

The SURGE (Scaleable URL Reference Generator) tool, presented in [3], is a workload generator compliant with the reported traffic models for the WWW. Specifically, SURGE follows the Web models for file sizes, request sizes, popularity, embedded references, temporal locality, and OFF times. SURGE manages to put much more stress on the CPU of the measured system than conventional benchmarks (e.g., SPECWeb96—see Section V.B.2) do. Additionally, under conditions of high workload, SURGE achieves the generation of self-similar network traffic, which is not the case with other tools.

B. Trace-Driven Workload Generation

An impressively high number of simpler tools for the performance evaluation of WWW server systems have appeared during the past years (http://www.softwareqatest.com/qatweb1.html). In the following paragraphs we briefly describe the most popular benchmark tools currently used by the WWW community.

1. WebStone

WebStone [41] is a WWW server benchmark–performance evaluation tool originally developed by Silicon Graphics Inc. (SGI). Currently, the tool is being progressed by Mindcraft Inc. (http://www.mindcraft.com), which acquired the WebStone rights from SGI.

WebStone allows the measurement of a series of variables, namely the average and maximum connect time, the average and maximum response time, the data throughput rate, and lastly, the number of pages and files retrieved. The benchmark software is executed simultaneously in one or more clients positioned in the same network as the measured server. Each client is capable of spawning a number of processes, named "WebChildren," depending on how the system load has been configured. Each WebChild requests information from the server based on a given configuration file. WebStone treats the server as a black box (black box testing). The execution of WebChildren is coordinated by a process called WebMaster. The WebMaster distributes the WebChildren software and test configuration files to the clients. Then it starts a benchmark run and waits for the WebChildren to report back the performance they measured. Such information is combined in a single report by the WebMaster. WebStone has been extended to allow the performance evaluation of CGI programs and server API modules (in WebStone 2.5, both ISAPI and NSAPI are supported). The performance measured is largely dependent on the configuration files used by WebChildren. Although such files can be modified, using the standard WebStone file enables the comparison between different benchmark reports. This standard file, practically, focuses on the performance of the Web server, operating system, network, and CPU speed.

WebStone configuration files determine basic parameters for the testing procedure (e.g., duration, number of clients, number of WebChildren) as well as the workload mix that should be used. Each workload mix corresponds to a specific set of resources that should be requested, in a random sequence, from the measured server. Mixes may contain different sizes of graphic files or other multimedia resources (e.g., audio, movies). As discussed in the previous paragraph, WebStone mixes were recently enriched to invoke CGI scripts for ISAPI/NSAPI compliant modules.

2. SPECWeb96

SPECWeb96 [38] was released by the Standard Performance Evaluation Committee (SPEC) in 1996. The benchmark tool is quite similar to WebStone: it implements black box server testing (also termed SUT, System Under Test). Workload is generated by one or more client processes ("drivers") running on one or more workstations. Similarly to WebStone, SPECWeb96 can be configured to generate different workloads. To provide a common indications of server efficiency, SPECWeb96 introduced the SPECWebOps/sec metric, calculated as the number of successfully completed requests divided by the duration of the test.

SPECWeb's advantage over WebStone is that the former is a standardized benchmark, agreed among a number of manufacturers, while the latter represents an individual effort that has, however, attracted the interest of the WWW community. In general, the abundance of Web benchmarks renders the comparison of measurements quite difficult and unreliable. This problem is addressed by SPECweb96, which, though, is not as sophisticated as other tools are.

Although both the WebStone and SPECWeb96 report the number of HTTP operations per second, their measurements are not comparable since their generated workload is quite different. Due to its larger file-set sizes and its

pattern of access, SPECweb96 gives emphasis to a system's memory and I/O subsystem.

The workload mix in SPECWeb comprises files organized in four classes: files less than 1 KB (35% of all requests), files in the range 1–10 KB (50% of requests), files between 10 and 100 KB (14%), and finally, larger files (only 1% of total requests). Within each class there are nine discrete sizes. Accesses within a class are not uniformly distributed. Instead, a Poisson distribution is being used within the limits of the file class. Files are retrieved from a series of directories with a specific structure; this scheme tries to emulate real-world conditions. The contents of the directories are compatible with the file size distributions discussed above while the number of directories varies according to the maximum load to be reached.

SPECWeb is confined only to the GET HTTP method. SPECWeb cannot handle other types of requests like POST. Additionally, CGI issues are not addressed by the benchmark.

According to [2], both the WebStone and SPECWeb96 tools are not capable of generating client request rates that exceed the capacity of the server being tested or even emulating conditions of bursty traffic. This is mainly due to certain peculiarities of the Transport Control Protocol (TCP), which the HTTP uses as a reliable network service. To tackle the aforementioned problem, the same paper proposes a quite different Socket connection handling by the benchmark software (called S-Client for scaleable client). Measurements presented in the same paper show that S-Client manage to really put the server software under stress.

Other tools in the Web benchmarking area include Microsoft's InetLoad (http://www.downloadsafari.com/Files/inetapps/I/InetLoad.html), and eTesting Labs' WebBench (http://www.etestinglabs.com/benchmarks/webbench/webbench.asp).

VI. EPILOGUE

Database connectivity is surely one of the most important issues in the constantly progressing area of WWW software. More and more companies and organizations are using the WWW platform for exposing their legacy data to the Internet. On the other hand, the amazing growth of WWW content forces the adoption of technologies like RDBMS for the systematic storage and retrieval of such information. Efficiency in the mechanisms intended for bridging the WWW and RDBMS is a very crucial topic. Its importance stems from the stateless character of the WWW computing paradigm, which necessitates a high frequency of short-lived connections toward the database management systems. In this chapter, we have addressed a series of issues associated with the considered area of WWW technology. We have evaluated different schemes for database gateways (involving different gateway specifications and different types of database middleware). Although aspects like generality, compliance to standards, state management, and portability are extremely important their pursuit may compromise the performance of the database gateway. The accumulated experience from the use and development of database gateways over

the past 5–6 years suggests the use of architectures like the database demon scheme, which try to meet all the above-mentioned requirements to a certain extent but also exhibit a performance close to server APIs.

ACKNOWLEDGMENTS

We are indebted to Mr. J. Varouxis for extremely useful discussions and technical assistance in the preparation of the experiments presented in this chapter. We also express our thanks to Mrs. F. Hadjiefthymiades for reading this draft and commenting on its technical content.

REFERENCES

1. *COLD FUSION User's Guide Ver. 1.5,* Allaire Corp. 1996.
2. Banga, G., and Druschel, P. Measuring the capacity of a Web server. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems,* Monterey, Dec.1997.
3. Barford, P., and Crovella, M. Generating representative Web workloads for network and server performance evaluation. In *proceedings of ACM SIGMETRICS, International Conference on Measurement and Modeling of Computer Systems,* July 1998.
4. Bernstein, P. *et al.* The asilomar report on database research. *ACM SIGMOD Record* **27**(4): 1998.
5. Bray, T. Measuring the Web. *Comput. Networks ISDN Systems* **28**(7–11): 1996.
6. Brown, M. *Fast CGI specification.* Open Market Inc., available at http://www.fastcgi.idle.com/devkit/doc/fcgi-spec.html, Apr. 1996.
7. Brown, M. FastCGI: A high performance Gateway interface. Position paper for the workshop *Programming the Web—A search for APIs, 5th International WWW Conference,* Paris, France, 1996.
8. Brown, M. *Understanding FastCGI Application Performance.* Open Market Inc., available at http://www.fastcgi.idle.com/devkit/doc/fcgi-perf.htm, June 1996.
9. Chang, P. I. Inside the Java Web Server: An overview of Java Web server 1.0, Java Servlets, and the JavaServer architecture. Available at http://java.sun.com/features/1997/aug/jws1.html, 1997.
10. Coar, K., and Robinson, D. The WWW Common Gateway Interface—Version 1.2. Internet Draft, Feb. 1998.
11. Crovella, M., Taqqu, M., and Bestavros, A. Heavy-tailed probability distributions in the World Wide Web. In *A Practical Guide to Heavy Tails—Statistical Techniques and Applications* (Adler R., Feldman R., and Taqqu M., Eds.). BirkHauser, Basel, 1998.
12. Everitt, P. The ILU requested: Object services in HTTP servers. W3C Informational Draft, Mar. 1996.
13. Gokhale, A., and Schmidt, D. C. Measuring the performance of communication middleware on high speed networks. In *Proc. ACM SIGCOMM Conference,* 1996.
14. Hadjiefthymiades, S., and Martakos, D. A generic framework for the deployment of structured databases on the World Wide Web. *Comput. Networks ISDN Systems* **28**(7–11): 1996.
15. Hadjiefthymiades, S., and Martakos, D. Improving the performance of CGI compliant database gateways. *Comput. Networks ISDN Systems* **29**(8–13): 1997.
16. Hadjiefthymiades, S., Martakos, D., and Petrou, C. State management in WWW database applications. In *Proceedings of IEEE Compsac '98,* Vienna, Aug. 1998.
17. Hadjiefthymiades, S., Martakos, D., and Varouxis, I. Bridging the gap between CGI and server APIs in WWW database gateways. Technical Report TR99-0003, University of Athens, 1999.
18. Hadjiefthymiades, S., Papayiannis, S., Martakos, D., and Metaxaki-Kossionides, Ch. Linking the WWW and relational databases through Server APIs: A distributed approach. Technical Report TR99-0002, University of Athens, 1999.

19. Performance benchmark tests of Unix Web servers using APIs and CGIs. Haynes & Company, Shiloh Consulting, available at http://home.netscape.com/comprod/server_central/performance_whitepaper.html, Nov. 1995.
20. *Informix-ESQL/C Programmer's Manual.* Informix Software Inc., 1996.
21. IS 9075-3, International standard for database language SQL—Part 3: Call level interface. ISO/IEC 9075-3, 1995.
22. Iyengar, A. Dynamic argument embedding: Preserving state on the World Wide Web. *IEEE Internet Comput.* Mar–Apr. 1997.
23. *JDBC Guide: Getting Started.* Sun Microsystems Inc., 1997.
24. Khare, R., and Jacobs, I. W3C recommendations reduce "world wide wait". World Wide Web Consortium, available at http://www.w3.org/Protocols/NL-PerfNote.html, 1997.
25. Kristol, D., and Montuli, L. HTTP State Management Mechanism. RFC 2109, Network Working Group, 1997.
26. Laurel, J. dbWeb white paper. Aspect Software Engineering Inc., Aug. 1995.
27. Mazzocchi, S., and Fumagalli, P. Advanced Apache Jserv techniques. In *Proceedings of ApacheCon '98,* San Francisco, CA, Oct. 1998.
28. McGrath, R. Performance of several HTTP demons on an HP 735 workstation. Available at http://www.archive.ncsa.uiuc.edu/InformationServers/Performance/V1.4/report.html, Apr. 1995.
29. *Internet Server API (ISAPI) Extensions.* MSDN Library, MS-Visual Studio '97, Microsoft Corporation, 1997.
30. *ODBC 3.0 Programmer's Reference.* Microsoft Corporation, 1997.
31. Nance, B. Examining the network performance of JDBC. *Network Comput. Online* May 1997.
32. *Writing Web Applications with WAI—Netscape Enterprise Server/FastTrack Server.* Netscape Communications Co., 1997.
33. *User's Guide, WebDBC Version 1.0 for Windows NT.* Nomad Development Co., 1995.
34. *CORBA: Architecture and Specification.* Object Management Group, 1997.
35. *Programmer's Guide to the Oracle Pro*C/C++ Precompiler.* Oracle Co., Feb. 1996.
36. Orfali, R., and Harkey, D. *Client/Server Programming with JAVA and CORBA.* Wiley, New York, 1998.
37. Ju, P., and Pencom Web Works, *Databases on the Web—Designing and Programming for Network Access.* M&T Books, New York, 1997.
38. An explanation of the SPECweb96 benchmark. Standard Performance Evaluation Corporation, available at http://www.spec.org/osg/web96/webpaper.html, 1996.
39. Stevens, W. R. *UNIX Network Programming.* Prentice–Hall, Englewood Cliffs, NJ, 1990.
40. Tracy, M. *Professional Visual C++ ISAPI Programming.* Wrox Press, Chicago, 1996.
41. Trent, G., and Sake, M. WebSTONE: The first generation in HTTP server benchmarking. Silicon Graphics Inc., Feb. 1995.
42. Data management: SQL call-level interface (CLI). X/Open CAE Specification, 1994.
43. Yeager, N., and McGrath, R. *Web Server Technology—The Advanced Guide for World Wide Web Information Provides.* Morgan Kaufmann, San Mateo, CA, 1996.
44. Karish, C., and Blakeley, M. Performance benchmark test of the Netscape fast-track Beta 3 Web Server. Mindcraft Inc., available at http://www.mindcraft.com/services/web/ns01-fasttrack-nt.html, 1996.
45. Performance benchmark tests of Microsoft and NetScape Web Servers. Haynes & Company, Shiloh Consulting, Feb. 1996.

This Page Intentionally Left Blank

18

INFORMATION EXPLORATION ON THE WORLD WIDE WEB

XINDONG WU

Department of Computer Science, University of Vermont, Burlington, Vermont 05405

SAMEER PRADHAN

JIAN CHEN

Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

TROY MILNER

JASON LOWDER

School of Computer Science and Software Engineering, Monash University, Melbourne, Victoria 3145, Australia

Database and Data Communication Network Systems, Vol. 3

I. INTRODUCTION

Over the past ten years, the World Wide Web (or the Web, for short) has grown exponentially [9,39,51,64]. According to the Internet Domain Survey by Zakon [70], the Internet has grown from only 617,000 hosts in October 1991 to over 43 million hosts in January 1999 and in excess of 4 million Web servers in April 1999. The amount of information on the Web is immense. Commercial sites like Lycos [37], Alta Vista [3], and Web Crawler [61] and many others are search engines that help Web users find information on the Web. These commercial sites use indexing software agents to index as much of the Web as possible. For example, Lycos [37] claims that it has indexed more than 90% of the Web [34].

Currently, the Web is indexed mainly on its visible information: headings, subheadings, titles, images, metadata, and text. Information retrieval from indexes is usually via one of the search engines, where submission of keywords as a query returns a list of Web resources related to the query.

The chapter is organized as follows. Section II provides an introduction to the most popular Web browsers, Netscape Communicator and Internet Explorer, from which you can start your search engines. Section III describes how search engines work on the Web. Section IV analyzes some leading search engines. In Section V, we outline the limitations of existing search engines, and review some research efforts on advanced Web information exploration using data-mining facilities.

II. GETTING STARTED WITH NETSCAPE COMMUNICATOR AND INTERNET EXPLORER

A Web browser is a software program that enables you to find and view information published on the Web. The first browser, called NCSA Mosaic, was developed at the National Center for Supercomputing Applications in the early 1990s. The easy-to-use point-and-click interface helped popularize the Web, although few at that time could imagine the explosive growth that would soon occur.

Although many different browsers are available, Microsoft Internet Explorer (referred to as Explorer, for short) and Netscape Navigator/Communicator (referred to as Netscape, for short) are the two most popular and probably the only two browsers you should consider for general Web surfing. Netscape Communications Inc. and Microsoft Inc. have put so much money into their browsers that the competition cannot keep up. The battle between the two companies to dominate the market has led to continual improvements to their software. Version 3.0 and later releases of either browser are excellent choices. Both Explorer and Netscape are based on NCSA Mosaic. You can download Explorer and Netscape for free from each company's website. If you have one browser already, you can test out the other. Also note that there are slight differences between the Windows and Macintosh versions.

Browsers come loaded with all sorts of features. Fortunately, you can learn the basics in just a few minutes, then take time to explore the more advanced functions. Both Explorer and Netscape have more similarities than differences,

FIGURE 1 Netscape icon.

so we will primarily cover their similarities. For the most up-to-date information about the browsers and a complete tutorial, check out the online handbook under the Help menu (on either Explorer or Netscape) or go to the Websites of the respective software companies.

A. Starting with Your Web Browser

When you first launch your Web browser, usually by double-clicking on the icon (Figs. 1 and 2) on your desktop, a predefined Web page will appear. With Netscape for instance, you will be taken to Netscape's NetCenter.

You can change the home page that loads when you launch your browser. For example, with Netscape Version 4.6, go to the Edit menu, then select Preferences. In the Home page section, type in the new Web address in the text box. Anytime you want to return to your home page from any other Website, just click the Home button on Netscape's toolbar (Fig. 3).

If you are using Explorer, and you want to set the home page to your favorite Website, just go to that site, then click the View menu, then select Options. Click the "Navigation" tab, then "Use Current" button. Finally, click "OK." And it is done!

Both Netscape and Explorer have a small picture in the upper right hand corner of the browser. When this image is animated, it indicates that your browser software, known as a client, is accessing data from a remote computer, called a server. The server can be located anywhere. Your browser downloads remote files to your computer, then displays them on your screen. The speed of this process depends on a number of factors: your modem speed, your Internet service provider's modem speed, the size of the files you are downloading, how busy the server is, and the traffic on the Internet.

At the bottom of the Web browser you will find a window known as a status bar. You can watch the progress of Web page transactions, such as the address of the site you are contacting, whether the host computer has been contacted, and the size of the files to be downloaded.

Once the files are downloaded and displayed in your browser, you can click on any of the links to jump to other Web pages. However, it is easy to get lost on this electronic web. That is where your browser can really help you.

FIGURE 2 Explorer icon.

FIGURE 3 Netscape toolbar.

The row of buttons at the top of your web browser, known as the toolbar (Figs. 3 and 4), helps you travel through the Web of possibilities, even keeping track of where you have been. Since the toolbars for Netscape and Explorer differ slightly, we will first describe what the buttons in common do:

- "Back" returns you to the previous page you have visited.
- "Forward" returns you to that page again.
- "Home" takes you to whichever home page you have chosen. (If you haven't selected one, it will return you to the default home page, usually the Microsoft or Netscape Website.)
- "Reload" or "Refresh" does just that, loads the current web page again. Why would you want to do this? Sometimes all of the elements of a Web page do not get loaded at the first time, because the file transfer is interrupted. Also, when you download a Web page, the data are cached, meaning they are stored temporarily on your computer. The next time you want that page, instead of requesting the file from the Web server, your Web browser just accesses it from the cache. However, if a Web page is updated frequently, as may be the case with news, sports scores, or financial data, you would not get the most current information. By reloading the page, these timely data are updated.
- "Print" lets you make a hard copy of the current document loaded in your browser.
- Finally, "Stop" stops the browser from loading the current page.

1. Buttons Unique to Netscape

- You can turn off the graphic images (see Section II.C) that load when you access a Web page. Because graphic files are large, the page will load much faster if it is has only text. If you then decide you to view the graphics, click the "Images" button.
- "Open" lets you load a Web page you may have stored on your computer's hard drive. (With Explorer, you can access this feature under the File menu.)
- "Find" lets you search for specific words in a document.

FIGURE 4 Explorer toolbar.

2. Buttons Unique to Explorer

- "Search" connects to a page on Microsoft's Website that lists a number of Internet directories and resources.
- "Favorites" is where you can record the addresses of Websites that you want to revisit. (In Netscape, this feature is called Bookmarks and located on the Menu bar.)
- "Font" lets you change the font size of the text on the web page. Each click on the button increases the size through the four settings.

As you browse from page to page and Website to Website, your browser remembers where you have been. Take a look under the Go menu in Netscape and Explorer. There you will find a history of your visits. To return to a previous place, just click on the name. Once you close your browser, your history is lost. You can also use the Go menu to move forward or backward.

B. Searching the Web Using Explorer and Netscape

1. General Tips for Web Search

When you use a search engine to find information on the Web, it helps to find ways to narrow your search so it is faster and more efficient. Most of the search engines use the same set of operators and commands in their search vocabulary.

The following are the most commonly used operators and a brief description of each. These would be used when typing a keyword or phrase into a search engine.

- *Quotes* (" "): Putting quotes around a set of words will only find results that match the words in that exact sequence.
- *Wildcard Use* (*): Attaching an * to the right-hand side of a word will return partial matches to that word.
- *Using Plus* (+): Attaching a + in front of a word requires that the word be found in every one of the search results.
- *Using Minus* (−): Attaching a − in front of a word requires that the word not be found in any of the search results.

2. Netscape's Web Search Features

Netscape's search engine is called Net Search (Fig. 5). Net Search in 1999 let you choose between six different search tools: Netscape Search, Alta Vista [3], Excite [18], Infoseek [23], LookSmart [35], and Lycos [37]. All these tools contain slightly different information, but each works in much the same way.

FIGURE 5 Netscape Search.

FIGURE 6 Netscape Web directory.

How to Search the Web Using Net Search

- Click the Search button on Netscape's toolbar. Netscape will load the Net Search page.
- Type the keywords into the text box.
- Press Enter (Return) or click the Search button. In a moment, Netscape will load a page that contains links to pages related to the keywords you have typed. When a link looks interesting, just click the link.

If you do not find what you are looking for, you can try your search again with a different search tool. Just go back to the Search page and click the name of a different search engine. The keywords you are searching for will appear in the new search tool automatically.

Browsing Directories

A directory is similar to a search engine in that it also stores information about the Web in a database. While search engines catalog every page they can find, a directory includes only the information about pages that human beings have selected. These pages are sorted into categories. Most directories let you choose between typing words in a search box and clicking the names of categories to browse the information like a library. Netscape's directory is called Web Directory. Here is how to browse the Web directory (Fig. 6):

- In the Location field, type the following URL: `http://home.netscape.com/directory`.
- Press Enter (Return). Netscape will load the Netscape Web Directory.
- Look at the list of topics and click on something that looks interesting to you. Netscape will load a page related to the topic you have chosen. On the loaded page, you will see a list of related topics.

Finding It Faster with Netscape Smart Browsing

Search engines are a good help for finding the information you are looking for, but you may not want to conduct a search every time you want to find a Web site that you just know is out there.

New versions of Netscape offer powerful Smart Browsing features that help you quickly and easily find the information that you need on the Web. Instead of remembering long URLs or typing an entire address to go to a site, Smart Browsing allows you to use a plain language to find exactly the information you want.

Two important features of Smart Browsing are Internet Keywords and What's Related.

Internet Keywords. Internet Keywords let you type words or questions into the Location field. When you press Enter (Return), Netscape sends the query to the Internet Keywords database, which then decides what to do with your words.

- If your word is a domain name, such as Netscape, Netscape will send you to search the home page associated with that domain.
- If your term is a popular company name, such as IBM, Netscape will look up the address and send you to that company's home page.
- If your search is a popular topic, such as computer, Netscape will direct you to a page of links related to that topic.
- If all failed, Netscape will automatically use Net Search to search the Web for your query.

What's Related. After you load a web page, just click the What's Related button on the Location field to load a list of pages that are related to the page you are viewing. It may take a minute for the results to load.

3. Explorer's Web Search Features

With Explorer Version 4.0, Microsoft introduced AutoSearch, a feature that allows users to type in a phrase in the Address Bar and Explorer will automatically conduct a search.

Explorer Version 5.0 adds to this feature by adding the ability to immediately direct users to the site that best matches the words they typed into the Address Bar. Along with the redirection to the best matched site Explorer Version 5.0 also opens the Search Pane, providing the user with other possible sites to match what they typed in.

Microsoft Network (MSN) is the default provider of AutoSearch; however, users can select which search provider they would like to use by going into the Customize button in the Search Assistant.

The Search Assistant is a tool that helps users to quickly and easily obtain more accurate results in their Internet searches.

When users click the Search button on the Explorer toolbar, the Search Assistant will open in the Search Pane. While continuing to provide users with the same method for searching the Web using the default search engine of their choice, the Search Assistant helps users designate the type of information they are looking for. This allows users to conduct a targeted search more easily.

C. Speeding up Web Browsing

There are many ways to make the browsing process much quicker and easier. One of the main ways to improve the speed of the computer when loading a document is to disable the automatic loading of images. Both Netscape and Explorer provide this facility.

1. With Netscape

- From the Netscape menu bar, select Edit, then Preferences. The Preferences dialog box will appear.
- Click Advanced on the Category list. That panel will move to the front.
- Uncheck Automatically Load Images.
- Click OK to close the Preferences dialog box and return to the Netscape window.

2. With Explorer

- From the Explorer menu bar, select View, then Options. The Options dialog box will appear.
- Select General on the Category list. That panel will move to the front.
- In the multimedia section, uncheck Show Pictures.
- Click OK to close the Options dialog box and return to the Explorer window.

The next time you access a web page, you will see little placeholders instead of the images that would normally load with the page, and your toolbar will have a new button called Images for you to click when you want to see the images on a page.

III. HOW SEARCH ENGINES WORK

Westera [63] indicates that search engines comprise three core components, a *gatherer* (Collector) that automatically collects Web document information, an *indexing and search* subsystem that indexes and allows queries on the collected information, and a *query interface* that provides a method for querying the index. Collection, categorization, and indexing of documents by a search engine is mostly automatic, and hence, search engines are notorious for returning numerous results from a query that are not very relevant. This is due to the lack of human intervention in the classification of the documents retrieved.

Document retrieval involving search engines is generally executed through a form where the user enters keywords or document attributes to be located. Two examples of search engines are HotBot [26] and AltaVista [3]. The search engines consult their databases, returning a list of documents considered relevant to the query. Some search engines permit query refinement, allowing the user to broaden or narrow the scope of the search using a query syntax, describing the sources of information (e.g., USENET, WWW), using a date to indicate the age of information, or restricting the query to a certain domain or country. This extra level of detail can also confuse the user rather than aid them in their search [63]. The syntax of notation used serves to further increase cognitive overhead that the user already experiences in learning to interact with a search engine [36].

Concept-based search engines try to determine the context and meaning of the user's query [8], e.g., a document relating to *Java* often contains the words *coffee, beverage, drink,* and *aroma,* giving the indexed document a context based on *Java coffee.* If the word *Java* is located in another document with words such as *programming, Internet, GUI,* and *virtual machine,* this document is likely to be about the *Java programming language.* Concept-based searching employs word associations to find words similar in meaning, even if the retrieved document does not contain precise keywords used in the query. An example of a concept-based search engine is Excite [18].

Meta-indexes, or *multiengine search tools* query multiple search engines and repositories (see Section C below) concurrently [63], and therefore can be a combination of the *repository* and *search engine* document retrieval techniques.

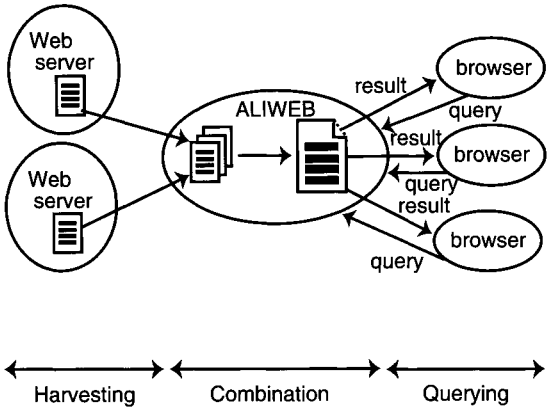


FIGURE 7 ALIWEB [29].

Technically, meta-indexes are not a search engine, but rather a query interface to multiple search engines. Meta-indexes mostly return a limited number of results from each search engine or subject directory they query, and hence, can be noncomprehensive [62]. Examples of meta-indexes are Dogpile [16], SavvySearch [56], and the *customizable multiengine search tool* [14].

There are two broad techniques that search engines employ to create and maintain their indexes, namely *Archie* indexing and *Harvest* indexing. We address them in Sections A and D, respectively.

A. Archie Indexing

Archie indexing is a term used to describe manual indexing of Web resources, where the index file is located on each server where the resource is located. One site, generally the site that the search engine is located, retrieves the index file from each Web server, allowing the combined index to be searched [29]. *ALIWEB* (Archie-like indexing on the Web) is an implementation of Archie indexing. Figure 7 describes the ALIWEB technique of combining its distributed indexes [29].

Search engines use specialized programs called *gatherers* (also known as "robots," "spiders," "collectors," or "worms") to traverse the Web and collect data. Archie indexing tries to remove some Web server load presented by these gatherers (see Section III. D. 1). Archie indexing does not require gatherers for resource discovery, minimizing the potentially serious dangers that gatherers have of overloading a server,[1] especially when several gatherers operate simultaneously. Such an approach also lowers the load on the computer hosting the gatherer by allowing inclusion of index data, with a minimum of overhead, into the existing index. Traditional gatherers are required to parse the document, extract the indexable information, and add this information to the index. This however, creates a substantially higher workload for the host.

[1] A local test by [29] shows the number of files a gatherer requested from a lightly loaded server was an average of 3 per second.

Archie indexing reduces another danger that gatherers present, security (see Section III.D.1). With Archie indexing, authors have a choice of what information is indexed, along with the descriptions for each indexed file. This security problem, however, is replaced by several other problems. The first is that an index must be located on each server. This index is maintained manually by *Webmasters,* and can become hard to keep up-to-date as documents may be out of the Webmasters control. Indexed documents can be moved or revoked, and new documents may be added by authors, and hence, the discovery of new information may be unsuccessful [29]. Another problem is that because an index is maintained on every server, these lists must be, in most cases, manually maintained to ensure correct indexing of each document. This effort can be time consuming and tedious. Many people believe that information providers are not willing, or able, to provide manually maintained indexes [29]. Scripts can be written to collect a list of files for the index; however, indexes created in this way must be manually reviewed to ensure the correct description of each indexed document.

B. Relevancy Ranking

Currently, Web documents are indexed mainly on their visible information: headings, subheadings, titles, images, meta-tags, and text. After a query is sent, search engines return a list of documents that are judged to be relevant to the search, in order of most relevant to least relevant. Some authors want their pages to be ranked highly [2] in a query, resulting in authors attempting to distort the search engines indexing and ranking algorithms. This is a type of *spamming,* with businesses opening solely to offer a guarantee that they can make a page frequently appear in the top ten results of a search [50].

It is not the size of the index, in terms of document numbers, that is important to the user, but the precision of the results. There has been recent optimism about textual content being the best way to rank results of a query. Spertus [59], Marchiori [38], Pringle *et al.* [50], and Barlow [8] argue that keyword searches have a lot of trouble distinguishing between words that are spelt the same but have different meanings: e.g., *Java* in the example of *concept-based* search techniques (see Section III). Barlow argues that this results in pages being returned that have little to do with the query. Many different approaches have been employed to combat this problem, like the ranking of pages depending on their hyperlinks (both outward and inward to the document) [38], indexing of picture and textual data [45], and ranking of text depending on its keyword location: e.g., words around keywords, and whether a word is part of the title of a document. These different approaches, to be accurate and effective, must be restricted to particular domains.

Ranking procedures of commercial search engines are not published because the ranking procedure is what gives the search engine the competitive "edge" and prevents spamming. However if we extrapolate commonly used

[2] Different weights can be assigned to words found in prominent areas; for example, keywords in titles and meta-information are ranked higher than keywords in heading tags and the document body.

text retrieval techniques such as those discussed by Salton [55], we can see some of the approaches in common use.

Keyword frequency is one of the more popular methods of index ranking. An inverted list is built of each file, and from there repeating words can be extracted to form an index. Words occurring in the title of the document and any headings can have a higher weight placed on them. Meta-data retrieval is one of the simple techniques for query indexing and ranking, where keywords in the meta-data are compared to the keywords in the body of the text to determine document relevancy. Meta-data can also be used to increase the word-weighting factor. Both of these options are easily abused by spamming. Anti-spam measures for such relevancy judgments include looking for duplicate words that repeat frequently within a short span of text, as well as words whose color is the same as the background color of the document. An anti-anti-spam measure employed for a short period of time was "hit-farms," where documents were duplicated under similar *uniform resource locators* (URLs) and sent to a search engine. Each of these documents had a redirect to the "real" page. Search engine owners took only a few weeks to include duplicate-document and near match URL tests to their indexing algorithms.

Ranking by inward links is one strategy employed to ensure that popular pages are always presented first. Pages with the greatest number of other pages pointing to them can be brought back first. Such an approach has the problem of spamming by collusion, where groups of people team together to link to each others sites to increase their number of inward links and thus ranking. "Voting" approaches are a different style of relevancy ranking. When users click on a site to visit from the results page, they have their choice counted by the search engine and the rankings adjusted for the query terms used.

The most accurate way for an author to ensure their page appears at the correct place in a hierarchy, or displayed when a certain query is entered is to be honest about the content of their document. Adding the keywords that are prominent in the text to the title of the document is useful in allowing users to identify the document quickly in a list of search results. Omitted titles in documents will quite often give the user "`No Title`" as the title of the document, or just the URL. Most Web-editing tools have a `Description` that you can fill in for the document. Many search engines look for this in order to get text to return as the search result. If you have to code this in by hand, it is worth entering. Inside the <HTML> tag, the author can place the following HTML:

```
<META NAME = "Description" CONTENT = "Joe's Motor Parts and Workshop">
```

Documents that do not have the "`Description`" `META` tag will often have the first 200 words extracted as the description of the document instead. Authors must, therefore, be careful about the text that makes up the first 200 characters of the document. Frequently the results from search engines appear with

```
Joe's Motor Parts and Workshop last updated 2/4/98
[Home] [Paints] [Mags] [Engines] [Exhausts] [Bookings] [Hours]
Welcome to Joe's! You will find the best in automotive...
```

C. Repositories

Repositories, sometimes called *subject directories,* are a hierarchical collection of references that allow a user to locate documents restricted by topic domain. Locating a document is performed by narrowing a selected topic from a broad area to a specialized area. To find a document on *hyper-text markup language* for example, a user would begin searching at the broad topic of *WWW,* then navigate through the document hierarchy, narrowing the topics until *hyper-text markup language* was located. An excellent example of a repository is Yahoo [68].

Humans manually maintain repositories. URLs are submitted to the repository by document owners, and then are indexed and categorized by the maintainers of the repository. The person submitting the document for indexing is asked to provide potential categories into which that page is to be placed. Maintainers may also be employed to search the Web for new pages, and hence Web pages found in this way, and through submission, lend themselves to being discarded. Repositories are one of the most basic search tools, and as an example, they resemble the table of contents of a paper, presenting users with little cognitive overhead in learning to interact with the index.

Repositories, being maintained and categorized by humans, are usually the most accurate at retrieving a particular document that is sought by the user. Such collections can be faster than search engines if trying to find an exact document when the topic domain is known. Westera [63] concludes that repositories can never index the entire Web due to human controls, but repositories are now themselves resorting to implementing search engines to keep their databases up-to-date [38].

D. How a Harvest Search Engine Works

The *Harvest* architecture was pioneered by Bowman *et al.* [11], and since has been partially replicated by many other advanced search engines. Harvest indexing was named as such to describe its "reaping" of Internet information. The following subsections will explain the central architecture of a Harvest search engine.

The architecture for Harvest can be broken down into three core subsystems: *gatherer, index and search,* and *query interface.* The gatherer subsystem (see Section 1 below) collects and extracts indexing information from one or more documents hosted by one or more Web servers. The index and search subsystem (see Section 2 below) provides indexing and searching of the gathered information, and the query interface subsystem (see Section 3 below) provides the interface to the index and search subsystem.

1. Gatherer Subsystem

Information retrieval of most search engines is done by the use of a gatherer (robot). A gatherer is an automatic program that traverses the Web collecting information [29]. The general algorithm for gatherer traversal is as follows [15]:

1. A gatherer starts with a certain URL for an initial Web page, *P.*
2. It retrieves *P,* extracts any URLs from *P,* and adds them to a list, *L.*

3. The gatherer then indexes *P,* or hands *P* to another subsystem to be indexed.
4. It then retrieves another URL from *L* (in some order[3]), which becomes the new *P,* repeating Steps 2 to 4 until it is stopped, or it empties *L.*

The largest danger in gatherer traversal is server security and overload [10, 11, 29]. Gatherers can potentially request files from any location on a server that a Web browser can, indexing any page which is linked, regardless of whether the author intended this link to be public. The overloading of servers is the other potential problem. Some gatherers have been known to query the same server 3 times per second [29], where retrieval of each URL requires a TCP connection, forking a UNIX process, changing directories, transmitting a file, and finally terminating the connection. These dangers have resulted in a surrogate standard being written, namely the "Standard for Robot Exclusion" [32].

The standard for robot exclusion describes a method for preventing a gatherer from retrieving particular directories and files from a Web server, through specification of gatherer access policies. This is achieved by creating a file accessible via HTTP on the server, located at the URL/`robots.txt`. The format of this file is governed by the standard, and is composed of a list of directories and files that the gatherer is not allowed to retrieve. Server administrators can, via this file, stop a gatherer from retrieving certain files on a server, thus preventing a gatherer from overloading a server and causing security breaches. This standard also enables an administrator to disallow certain gatherers from indexing a server, but still allow other gatherers to index a server. To implement this, the standard for robot exclusion states that each gatherer should identify itself to the server, state its purpose for indexing, and provide contact details of the gatherer's owner to the server. However, `/robots.txt` is in itself a security hole, as it publishes the location of directories that may contain sensitive data.

Harvest combats the problems that gatherers can cause in the following ways. The Harvest gatherer executes indexing software that is run on the site being indexed. The software scans a site, maintaining a "cache" of indexed information, so that separate connections from a gatherer to the index and search subsystem are not required. The gatherer then sends this cache to the indexing and search subsystem, allowing indexing of a site to be done in one process, minimizing server load [11].

2. Index and Search Subsystem

Harvest's index allows for diverse indexing and searching techniques that are required by different search engines. The principle requirements of a client search engine are that it supports incremental updates of its index and Boolean combinations of attribute-based queries. Harvest offers two different index and search subsystems: *Glimpse* and *Nebula.*

Glimpse

Glimpse supports space-efficient indexes, as low as 2–4% of the size of the source data; yet it still allows for Boolean queries, regular expressions queries,

[3]There have been studies on the order that a gatherer selects URLs out of its queue. Cho et al. [15] conclude that if a gatherer is intending to perform a single scan of the Web, then any URL order will suffice. This chapter is not investigating domain-based or topic-specific indexing, so the order that the gatherer will traverse the Web will not be considered.

and approximate matching (e.g., spelling mistakes). The index also allows for incremental updates and easy modification due to its relatively small size.

Glimpse is *based* on an inverted index (see Section III.E.1), with the main part of the index consisting of a list of all the words appearing in the index. For each word there is a pointer to a list of each occurrence of the word. The pointers can point to the same file, another file, or even positions in files, like paragraphs. Glimpse uses *agrep* [67] to search its main index and to search the areas found by the pointers in the main index. This also allows for approximate matching and other agrep services.

Bowman *et al.* [11] claim that using agrep allows for much more selective queries than keyword searching when the gatherer has extracted attributes on the gathered page, like titles, headings, or links. For example, searching for all pages with the keywords "incremental searching" with "unknown title" with one spelling mistake is possible.

Nebula

The purpose of *Nebula* is to allow fast queries, but as a tradeoff, the index size is larger than that of Glimpse. It provides support for *views,* which are defined by standing existing queries against the database. A view can scope a search to a particular subset of the database, for example, computing or homepages, allowing domain-based search engines to be designed.

Nebula can be extended to include domain-specific query resolution functions. A server that contains indexes of Masters theses can be extended with a query function to rank words found in "abstracts" and "introductions" higher than if they are located in the body of the thesis, for example. The database is set up with each term having a corresponding index that maps values to the objects, thus speeding up queries but increasing index size [11].

3. Query Interface Subsystem

Bowman *et al.* [11] indicate that there are two different goals to keep in mind when designing a query interface for use in an environment as diverse as the Internet. They believe search engine designers need to provide:

- A great degree of flexibility and the ability to customize for different communities and for different users.
- A simple, uniform interface so that users can move between domains without being overwhelmed.

The Harvest query interface handles all queries to the underlying database, allowing Boolean combinations of keywords and attribute-based expressions. Query results are given in a uniform format that includes index-specific information and the location of each object.

The query interface is a simple HTML form (see Section II. B for examples), which allows the query to be entered by the user, then spawns a process that queries the underlying database. The results of the query are presented in the browser as an HTML page, with hyperlinks to the location of returned pages. The current implementation of the Harvest query interface maintains no state across queries. The illusion of continuity and refinement of queries is created by embedding the query state information within those URLs that form the links

of the returned document. Query interface customization is supported through direct access to the index and search subsystem. Support of additional switches for approximate matching is provided by Glimpse, where Nebula can be used for alternate indexing and search support, with features like query refinement.

E. Anatomy of an Information Retrieval System

According to Gordon [21], there are two primary concerns for the representation of a document in an information retrieval system: devising a method of storing a representation of the document, and representing the needs of the user in the form of a query that can be processed by a computer. Campbell [13] adds that when these concerns are met, the solution improves the power and capability of the information retrieval system to retrieve documents that are of direct interest to the user.

Information retrieval systems have characteristics that make them different from traditional database management systems. Tenopir and Lundeen [60] and Campbell [13] suggested four primary differences between database management system data and information retrieval data:

1. Information retrieval systems store loose textual data, where little use is made of structural units such as fields. When fields are used they still only consist of text. Text processing is limited to search and retrieval.
2. The size of information retrieval systems is larger than traditional database management systems, owing to the way data are stored and also to the nature of the system—documents are rarely deleted.
3. Fields are generally not used in full-text databases, but when they are used (i.e., bibliographic databases), field length varies, causing set size fields to waste a lot of space.
4. Information retrieval systems that use fields require, in most cases, multiple values for each field, for example, an author field that contains a name that consists of three strings. Database management systems cannot handle multiple fields, and are therefore inappropriate for full-text databases.

Information retrieval systems consist of two core components [13]. The first, *file structure,* provides the system with a structure that allows documents to be stored. The second, *query language,* allows these stored documents to be queried and retrieved.

1. File Structure

Traditional file processing structures like B^+-*trees, hashing,* and *indexed sequential* rely on data having set attributes, but text-based information retrieval systems mostly do not contain set attributes. Faloutsos [19] argues that these methods can be employed if rearrangement of the document takes place before indexing, forcing a document's format to represent the more traditional database systems. The following describes two document rearrangement techniques, with the exception of the first, which discusses advantages and disadvantages of not rearranging the document structure.

Full-Text Scanning

With *full-text scanning* document structure is not rearranged at all. Retrieval consists of sequentially searching the stored documents and locating all documents that match the query. This method requires no extra storage space for indexes, and processing requirements are small when new documents are added, as the documents do not need rearrangement.

The largest advantage that full-text scanning has is that as documents change, reindexing is not required. Documents can be deleted and updated without effecting the system as a whole [13]. The major disadvantage to full-text scanning is that retrieval of documents may be slow, as sequentially scanning each document does take more time than using indexes to locate keywords in documents. Faloutsos [19] and Salton [55] demonstrate that speed can be increased by using parallel processors and faster string matching algorithms.

Document Inversion

Document inversion (file inversion or *inverted indexes)* is one of the oldest and most popular models for presenting full-text retrieval systems. Many authors, cited in Campbell [13], comment on the usefulness of this technique in full-text retrieval systems.

The file inversion method works by creating an index of all the unique keywords that exist in a document. As documents generally contain more than one word, the index consists of at least two levels. At the first level, the document's keywords are stored in a file called the *inverted file* and are generally arranged alphabetically. The inverted file can be organized in other traditional ways, like B^+-trees or hash-tables. Each keyword in the inverted file contains pointers to a *postings* file. The postings file contains the pointer to each keyword in the original document, such as the exact location, or more globally, a pointer to a sentence or a paragraph.

To retrieve a document from an inverted index, the following process can be used: The word that forms a query is located in the inverted file using an appropriate access method that matches the structure of the file. If the query word is located in the file, the pointer is followed to the postings file, where each posting is read to determine what documents to retrieve. The main advantage to inverted indexes is that they allow fast access to a document via a query [19], and further to that, via Boolean queries.

The large problem with inverted file indexing is the size of the index file it requires, which Faloutsos [19] demonstrates can be between 50 and 300% of the original size of the document. The removal of *stop words* from the index as described by Jones [28] can, however, reduce the size of the index and increase the accuracy of a query. Stop words are words that occur in text but are not expected to be search terms. When indexing documents, comparisons of keywords to be indexed with the list of stop words is performed, then if they correspond, the word itself is not indexed. This can greatly reduce the size of the index. Examples of stop words are *the, a, is,* and *I.*

Signature Files

Signature files, like inverted indexes, require at least one extra file to be created. Most authors, cited in Campbell [13], agree that this method is only

economical in *attribute indexing,* not full-text indexing. As such this method, which is similar to *hashing,* will not be discussed in detail. This method is centered around the creation of a signature file, which contains a calculated signature that maps each record in a file. Retrieval of a document is performed by calculating the location of the record from its corresponding signature.

There are many other techniques for text-retrieval systems, like *clustering* (grouping files with common attributes) and *multiattribute hashing* (hashing on multiple attributes), but these are not suitable for Web-based indexing, owing to the size of the index when indexing massive collections of Web documents.

2. Query Language

A *query language* specifies a set of rules for retrieving records (documents) from a database. In information retrieval, a query usually consists of a set of keywords linked by logical Boolean expressions "AND," "OR," and "NOT" [13]. Many systems today also allow *proximity searching:* queries using "close to" and "same sentence as" that can be used in conjunction with logical expressions.

Three vocabularies of querying are presented by Campbell [13]: *controlled, open,* and *natural.* The controlled vocabulary governs what keywords a query can consist of, where the words used in a search can only consist of the words that appear in the index. This form of retrieval is fast as all words in the query have a corresponding document. The second approach is using an open vocabulary, where any word can be used in a query. This approach means that sequential searching of documents must be performed, or some type of file organization where indexes are made available. The third approach is the use of a natural vocabulary. This query can be achieved in two ways. One uses pattern matching to locate documents that have a similar grammar to the query, and the other extracts the query sentence, compares terms, and allocates weights to each word in the term. It then weights each document with these words, returning documents that have the highest weight.

IV. TYPICAL SEARCH ENGINES

A. Search Engine Types

The Internet is divided into several different information spaces on the types of servers that accommodate information, which are the Gopher servers, the FTP (File Transfer Protocol) servers, the Telnet servers, and the HTTP (Hyper-text Transmission Protocol) servers (or Web servers). To facilitate quick searching through these servers, numerous services and applications have come into being. The Gopher space is searched by a program called "Veronica" (Very easy rodent-oriented net-wide index to computerized archives); the FTP space is searched with the program called "Archie" (from the word "archives"); the Telnet space is searched with the program called "Hytelnet", and the Web space, which is a superset of all the preceding information subsets, is searched with the help of "search engines" [25]. This last category of searching applications can search the whole Internet and is the most widely competed ones in the industrial arena. A *search engine* is a software program that takes a query from the user and finds information on numerous servers on the Internet using

that query [25]. It then filters the huge amount of information retrieved and provides the user with a list of Internet sites. The arrival of graphical browsers and therefore graphical search engines can be traced back to 1993.

There are currently six major search engines and many others that help a user to search through the maze of documents available on the Web. The majority of users that search the Internet use these search engines, and there have been strong efforts in academia to make the search simpler and more viable. In search applications, we have two distinct subcategories. They are *directories* or *repositories* (see Section III. C, for example, Yahoo [68] and Megallan [41]) and "query-based engines" (for example, Altavista [3] and Excite [18]). The latter search engines are further subdivided into "search engine directories" (for example. All-In-One Search [2]) "metasearch engines" (for example, SavvySearch [56] and AskJeeves [5]) and "subject-specific search engines" (for example, Medical World Search [40]).

Similar to the Web, the world of search engines has become complex, rich, volatile, and frequently frustrating [57]. Moreover the designers of these engines must cater for the user needs that comprise the entry of a small number of keywords to get relevant results. Not many users use the advanced facilities of search engines and hardly bother to decipher the mechanism behind the search engine [22]. As the search engine domain becomes more and more focused on particular subjects, the likelihood of a user getting relevant information increases, but the breadth of information is affected significantly. At the same time, as the search engines become more and more comprehensive, there can be various documents that are seemingly relevant, but belong to a completely different domain that uses the same semantics, but in a different sense (like *Java* is used to denote coffee as well as the programming language). Therefore the size of a search engine itself does not dominate the quality or relevancy of results. There are some search engines that try to, address this issue with the help of "concept search" (see Section III), which tries to find more than just the semantic meaning of the keyword, by considering other words that might accompany that keyword in the query or from the earlier queries of the same user (in case there is a record of the user available for analysis). Another way of knowing what the user is looking for is to get feedback from the initial query, for example, asking the user to identify the result that most closely matches the one that is expected. One of the search engines that exhibits these two features is Excite [18].

Each search engine can differ considerably in terms of the results generated, as it has its own perspective of the Web, which may or may not overlap with that of the other engines. It is observed sometimes that the optimum result set of hypertext links for a query is a list of selected results from several search engines. The plethora of search engines have made it difficult to decide which search engine to use and when. To make the most out of any search engine the user needs to know how it functions, *what* is it designed to retrieve, *when* it gives the best response time, and even the fact that it exists. Empirical results have indicated that no single search engine is likely to return more than 45% of relevant results [17]. Taking this clue, recently (from early 1994) the Web has witnessed an emergence of a new family of search engines that make use of several conventional search engines to bring relevant information to the

users. These are called "metasearch engines" (see Section III). In the same way as robot-based search engines were developed in response to the rapid Web growth, metasearch engines came into being owing to the increasing availability of conventional search engines. The search domain of these search engines is not the Web anymore, but the result set obtained through the use of interfacing conventional search engines. These metasearch engines, by means of some predetermined mechanism, decide which of the search engines they are going to avail and trigger a parallel query to each of them. They then analyze the outputs of these engines and display it in a uniform format. There are several metasearch engines on the Web today, including SavvySearch [56], AskJeeves [5], and MetaCrawler [42]. The query dissemination mechanism of metasearch engines must bring about a balance between two conflicting factors, viz. the *time taken to get the results* and the *relevancy of the results*. In some cases it is wise to avoid certain specialized engines as their response would hardly be relevant and sometimes to avoid certain engines just because they incorporate redundancy in the system—owing to a significant overlap between the areas of the Web that they cover. Some of these search engines like SavvySearch [56] allow the user to override the mechanism of selecting the search engines, by specifying whether a particular search engine is to be used for a query, and if so, then what *weightage* is to be given to the results generated by that search engine in the final compilation [17].

The mechanism of metasearch is complicated by four issues:

1. The corpus of the Web is not directly available and is indexed by the other search engines.
2. The search engine set comprises both the general and specific search engines and thus it must account for the varying expertise.
3. The capabilities of search engines are not a static factor and change continuously, depending on the update of their indexes.
4. The consumption of resources must be balanced against result quality.

These four issues are resolved with the help of the following mechanisms

1. *Generation of a metaindex* (see Section III) that tracks experiences in dispatching queries to the search engines.
2. *Ranking of each engine* based on the information in the metaindex as well as recent data on search engine performance.
3. *Varying the degree of parallelism* depending on the current machine and network conditions.

1. Metaindex of Prior Experience

A *metaindex,* which is essentially a matrix of the *number of terms used in a query* and the *name of the search engine used,* is constructed from passively accumulated user feedback over a prolonged period of time. A cell in the metaindex indicates the past performance of submitting the query to that search engine. Values are signed numbers: a *positive* representing good performance and a *negative* bad performance. The effectiveness values are rated based on the following two events: *No Results* and *Visits*. A *No Result* occurs when the search engine finds no document to the specified query; and a *Visit* occurs when

the user follows one of the links suggested by the search engine. Positive values are used to represent *Visit* events and negative values for the *No Visit* events. The intuition behind this is that higher positive values indicate the tendency of the search engine to return interesting results. Thus the likeliness of a search engine providing relevant results is considered while selecting it for a particular query or avoiding it as being a waste of resources.

2. Search Engine Ranking for a Query

Each search engine is *ranked* based on the following two factors: whether the search engine has done well over a period of time, and whether it has returned relevant results quickly in the recent past. Higher ranking engines are preferred while passing the query.

3. Degree of Parallelism

The *concurrency* value of an engine is calculated and used to determine the amount of resources needed for a particular query. It is inversely proportional to the estimated query cost, and is calculated using the following three factors: *expected network load, local CPU (central processing unit) load,* and *discrimination value.* The first two terms speak for themselves, the third term means that the search engine checks how general the query is, and in case it is very general, then it selects a lesser number of search engines, since there would be a significant overlap between the individual results, thus reducing the amount of duplicate elimination when the unique list is generated.

There is also another family of metasearch engines that bring about a fusion between the "search engine directories," which can also be referred to as the "manual" version of metasearch engines, and "fully automatic" metasearch engines. With search engine directories, a user selects the search engines that they want to employ for the required search and dictates some more parameters that decide the pressure on the system resources, in terms of time and network congestion, that they want to allot for the search. Automatic metasearch engines automatically decide all the above-mentioned factors. A hybrid search engines is ProFusion [52]. In a recent controlled experiment where a number of users were asked to rate the relevance of results obtained from 10 search engines, among which were MetaCrawler [41], SavvySearch [56], and ProFusion [52] (both manual and automatic) along with six other engines, ProFusion [52] returned the highest number of links judged to be relevant by the users [17].

B. Analysis of Some Leading Search Engines

Now let's look at some of the salient features that can be incorporated into the search engines for the purpose of enhancements and checking which of the search engines in the market address them. In our discussion, we will consider the following search engines: Altavista [3], Excite [18], Inktomi [24], which powers HotBot [26] and MSN [49], Infoseek [23], Lycos [37], and Northern Light [48]. Excite [18] also covers the Excite powered searches of AOL NetFind [4], Netscape Search [71], and WebCrawler [61]. Some data for Google [20] is also listed. These data are as of April 5, 1999 [58].

1. Deep Crawling

Search engines exhibiting deep crawling list many pages from a Website even if the site is not explicitly submitted to them. So the engines that support this feature will have, for searching, many more pages per site than those that do not implement it.

Alta Vista, Inktomi, and Northern Light implement this feature, whereas Excite, Infoseek, Lycos, and WebCrawler do not. In the case of WebCrawler, only home pages are listed.

2. Instant Indexing

An instant indexing search engine usually indexes the site submitted to it in a couple of days.

AltaVista, Infoseek, and MSN have this facility, whereas Excite, Inktomi, Lycos, Northern Light, and Google do not.

3. Frames Support

A search engine that demonstrates frame support follows all the links on a Web page that are in several different frames. So in case you have a lot of information that is represented in different frames, then you should make sure that the targeted search engines have this facility.

AltaVista and Northern Light are the only search engines that fully support this facility, whereas Lycos provides limited support and Excite, Inktomi, and Infoseek do not support it at all.

4. Image Maps

Image maps are images on Web pages that include hot regions (i.e., regions on the image that point to a particular Web page). This facility shows that a search engine can follow the client-side image maps (for more information see `http://www.cris.com/~automata/maps.htm`).

AltaVista, Infoseek, and Northern Light support this facility; however, Excite, Inktomi, and Lycos do not.

5. Password Protected Sites

This is the feature with which a search engine has access to data encapsulated at a password protected site. This means that in case the user is looking for something that is addressed at these sites, then the search engine indicates it, although the user needs to acquire the password in order to get the detailed information. With such search engines, there is a username and password for each password protected site.

AltaVista and Inktomi can access password-protected sites. However, Excite, Infoseek, Lycos, and Northern Light do not support it.

6. Robot Exclusion Standard

When Webmasters want to keep their Websites out of reach of some search engines, for privacy and other purposes, they must maintain a "`robot.txt`" file at some servers in the root directory of their Web servers (see Section III.D.1). The search engines that abide by this protocol will not index these Web sites,

whereas others will go ahead and index them. In case of servers where it is difficult or impossible to maintain this "robot.txt" file, there is another solution in the form of an HTML tag, called the Meta Robots tag. When this tag is inserted and set for an HTML file, the search engine ignores the HTML file for indexing purposes.

The following tag, if inserted in an HTML file, prevents it being indexed by search engines that follow this standard:

```
<META NAME="ROBOTS" CONTENT="NOINDEX">
```

All leading search engines support this protocol.

7. Link Popularity

Search engines implementing this feature determine the popularity of a page by counting the number of pages referring to the page.

Excite, Inktomi, and Lycos demonstrate this feature, whereas Altavista, Infoseek, and Northern Light do not.

8. Learns Frequency

This is the property of a search engine to record how frequently a Web page is updated.

Altavista and Infoseek have this ability, whereas Excite, Inktomi, Northern Light, and Lycos do not.

9. Full Body Text Indexing

All major search engines index the full visible body of Web text, although some exclude stopwords or the words that are deemed to be spam.

10. Stop Words

Some search engines do not consider stopwords (*to, at, and,* etc.) while indexing web pages.

AltaVista, Inktomi, Excite, Lycos, and Google ignore the stop words, whereas Infoseek and Northern Light do not.

11. ALT Text

Some search engines also index the ALT text (i.e., the text that is used as a substitute for images, in case the browser does not support their display or in case the user chooses to view a "text only" copy of the Web page) associated with some images in Web files.

AltaVista, Infoseek, and Lycos have this feature, whereas Excite, Inktomi, and Northern Light do not.

12. Stemming

Some search engines search for the variations of the keywords submitted in queries; for example, in the case of the keyword "grant," the search engines will also find pages for "grants," "granting," etc.

Infoseek, Lycos, and Northern Light have the stemming feature turned on by default, whereas in HotBot the user must manually turn it on. Altavista on the other hand searches for related terms, and in Excite you can modify the

search by telling it to find Web pages similar to one of the previous result sets that is, the closest to the favorable response.

13. Meta Tags Boost Ranking

Some search engines rank a page higher when one of the keywords falls in the META tag of the HTML page.

Infoseek and Inktomi support this feature, while AltaVista, Excite, Northern Light, and Lycos do not.

14. Links Popularity Boosts Ranking

The number of links to the page under consideration is used by some search engines while ranking the Web page.

AltaVista, Excite, Google, and Infoseek respect this type of ranking, while Inktomi, Lycos, and Northern Light do not.

15. Spamming through Meta Index Refresh

Some site owners create target pages that automatically take visitors to different pages within a Web site. This is done using the meta index refresh tag.

For example, the following tag, when inserted in an HTML file, will display the page in which it is placed, and then after 15 s will display `http://mywebpage.com`

```
<META http-equiv="refresh" content="15;URL=http://mywebpage.com">
```

AltaVista and Infoseek do not index pages with any redirection whatsoever, whereas Excite, Inktomi, Lycos, and Northern Light go ahead and index them.

16. Invisible Text Spamming

Some Web designers insert a lot of irrelevant keywords in their HTML file with the foreground color same as the background color. For the person visiting that pages these keywords are invisible and the listing of these pages in the search result set would not make any sense. Some search engines detect this type of spamming and penalize these pages.

AltaVista, Infoseek, Inktomi, Lycos, and Northern Light detect invisible text, whereas Excite does not.

17. Tiny Text Spamming

It is similar to the invisible text spamming case, and the only difference is that the invisible text is too small and hardly readable. There are some search engines that refuse to index the Web pages that have text below some font size.

AltaVista, Inktomi, and Lycos, detect this type of spamming, whereas Excite, Infoseek, and Northern Light do not.

C. Search Engine Sizes

The larger a search engine's index is, the more is the probability of finding requested information, at the same time more are the chances of getting back irrelevant results.

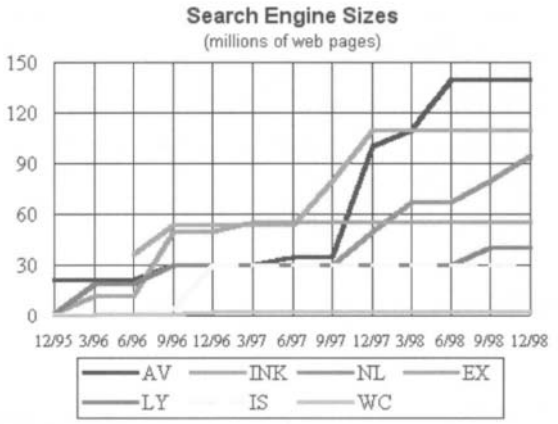


FIGURE 8 Search engine sizes. AV, AltaVista; INK, Inktomi; NL, Northern Light; EX, Excite; LY, Lycos; IS, Infoseek; WC, WebCrawler.

Serious searchers prefer to use search engines with a larger search index and more number of keywords. Figure 8 shows how the index of each search engine has increased since 1995 [58].

D. Summing It Up

Many efforts have been made to evaluate the performance of different search engines; however, they lack the required systematic efforts and consistency between the parameters chosen to measure them. The absence of user involvements has also been a problem for many such studies [57].

V. ADVANCED INFORMATION EXPLORATION WITH DATA MINING

Along with the enormous growth of the Web, search engines can return a large number of Web pages for a single search. Thus, it becomes time consuming for the user to go through the list of pages just to find the information. Information retrieval on the Web becomes more and more difficult. To remedy this problem, many researchers are currently investigating the use of their own robots (or "Web wanderers" [30]) that are more efficient than general search engines. These robots are software programs that are also known as agents, like CIFI [34], Web Learner [49], and many others. Some of theses agents are called intelligent software agents [54] because they have integrated machine learning techniques. The Web page entitled "Database of Web Robots Overview" at `http://www.robotstxt.org/wc/active/html/index.html` lists 272 of these robots or agents as of January 10, 2002.

The advantages of these robots are that they can perform useful tasks like statistical analysis, maintenance, mirroring, and most important of all resource discovery. However, there are a number of drawbacks, such as the following: They normally require considerable bandwidth to operate, thus resulting in network overload, bandwidth shortages, and increases in maintenance costs. Due

to the high demand of robots, network facilities are required to be upgraded—consequently resulting in budget increases. Robots generally operate by accessing external servers or networks to retrieve information. This raises ethical issues as to whether people should improve their system just because too many robots are accessing their sites. Koster mentioned in his paper "Robots in the Web: Threat or treat ?" that a robot visited his site using rapid fire requests and after 170 retrievals from the server, the server crashed [30].

A. SiteHelper: A Localized Web Helper

SiteHelper [46,47] is a software agent with incremental machine learning capacities to help the user explore the Web. It first learns about a user's areas of interest by analyzing the user's visit records at a particular Web site, and then by assisting the user retrieve information by providing the user with update information about the Web site. SiteHelper carries out two types of incremental learning: interactive learning and silent learning.

Interactive incremental learning functions in cycles that interact with the user. SiteHelper prompts the user with a set of keywords that are likely of the user's interest, and asks for feedback. Considering the feedback SiteHelper makes changes to its search and selection heuristics and improves its performance.

Many Web site servers implement a log file system that records user access information to their sites. The log files normally consist of the computer name and location on the Internet, the time of access and accessed Web pages. Silent incremental learning uses the log information as its starting point. SiteHelper extracts a log file for each user from a local Web site server. From the log file, SiteHelper learns about the user's interest areas. It extracts a set of keywords about the user's interest areas according to the Web pages the user has visited in the past. In addition, SiteHelper examines the time the user spends on each page. If the user spends little or no time on certain pages, these pages will not be counted as interesting pages to the user.

SiteHelper is a site agent that learns a Web user's areas of interest and assists the user in finding information on one localized Web site. It works differently from search engines and other kinds of agents like WebWatcher [27] and World Wide Web Worm [65] that help the user on the global Web. However, other web sites can deploy SiteHelper to assist users find information in the same way. This design of SiteHelper has followed the "Guidelines for Robot Writers" [31]; therefore it avoids the drawbacks of existing robots. In addition, there are other advantages of having SiteHelper at a local Web site.

- Through incremental learning of the user's characteristics or interest areas, SiteHelper becomes an assistant to the user in retrieving relevant information. The idea of each Web site doing its own housekeeping and assisting its clients (Web users who visit the site) would have significant potential of becoming a good methodology or technique for information retrieval on the Web.
- SiteHelper has the potential of reducing user accessing and retrieval time. For example, six months ago, a user visited the Monash Computer Science departmental site looking for a particular paper. At the relevant Web page, the PostScript file of the paper was not yet available. During this user's next access,

at the point of entry to the department site, SiteHelper displays a list of changes that have been made since the user last visited. By viewing these changes, the user will know whether the PostScript file is now available, rather than accessing the Web page again.

- SiteHelper can be easily adopted for many other services, such as library sites, Internet music stores, and archives. With library sites, for example, different users have interests in different topics: SiteHelper can retrieve the books related to the user's topics during their visit. In their next visit, the system will let the user know what other additional books or materials have become available since their last visit.

B. Other Advanced Web Exploration Agents

Assisting Web users by identifying their areas of interest has attracted the attention of quite a few recent research efforts. Two research projects reported in [7] at Stanford University and [69] at Stanford University in cooperation with Hewlett–Packard laboratories are along this line. Two other projects, WebWatcher [6] and Letizia [33] also share similar ideas.

Balabonovic and Shoham [7] have developed a system that helps a Web user to discover new sites that are of the user's interest. The system uses artificial intelligence techniques to present the user every day with a selection of Web pages that it thinks the user would find interesting. The user evaluates these Web pages and provides feedback for the system. The user's areas of interest are represented in the form of (keyword, weight) pairs, and each Web page is represented as a vector of weights for the keywords in a vector space.[4] From the user's feedback, the system knows more about the users' areas of interest in order to better serve the users on the following day. If the user's feedback on a particular Web page is positive, the weights for relevant keywords of the web page are increased; otherwise, they are decreased.

This system adds learning facilities to existing search engines and, as a global Web search agent, does not avoid the general problems associated with search engines and Web robots. In addition, the (keyword, weight) pairs used in this system cannot represent logical relations between different keywords. For example, if the user's areas of interest are "data mining on Internet" and "rule induction," the logical representation should be ("data mining" AND "Internet") OR "rule induction."

Yan *et al.* [69] investigate a way to record and learn user access patterns in the area of designing on-line catalogues for electronic commerce. This approach identifies and categorizes user access patterns using unsupervised clustering techniques. User access logs are used to discover clusters of users that access similar pages. When a user arrives, the system first identifies the user's pattern, and then dynamically reorganizes itself to suit the user by putting similar pages together.

An (item, weight) vector, similar to the (keyword, weight) vector used to represent each web page in [7], is used in [69] to represent a user's access pattern. The system views each Web page as an item, and the weight of a user

[4]The vector space approach is one of the most promising paradigms in information retrieval [7].

on the item is the number of times the user has accessed the Web page. This system does not use semantic information (such as areas of interest) to model user interests, but just actual visits. Also, it does not aim to provide users with newly created or updated Web pages when they visit the same Web site again.

WebWatcher [6] designed at Carnegie Mellon University is an agent that helps the user in an interactive mode by suggesting pages relevant to the current page the user is browsing. It learns by observing the user's feedback to the suggested pages, and its objective is to guide the user to find a particular target page. A user can specify their areas by providing a set of keywords when they enter WebWatcher, mark a page as interesting after reading it, and leave the system at any time by telling whether the search process was successful. WebWatcher creates and keeps a log file for each user, and from the user's areas of interest and the "interesting" pages they have visited, it highlights hyperlinks on the current page and adds new hyperlinks to the current page.

WebWatcher is basically a search engine, and therefore does not avoid the general problems associated with search engines and Web robots. In addition, it helps the user to find a particular target page rather than incremental exploration of all relevant, newly created, and updated pages at a local site.

Letizia [33] learns the areas that are of interest to a user, by recording the user's browsing behavior. It performs some tasks at idle times (when the user is reading a document and is not browsing). These tasks include looking for more documents that are related to the user's interest or might be relevant to future requests.

Different from WebWatcher, Letizia is a user interface that has no predefined search goals, but it assumes persistence of interest; i.e., when the user indicates interest by following a hyperlink or performing a search with a keyword, their interest in the keyword topic rarely ends with the returning of the search results. There are no specific learning facilities in Letizia, but just a set of heuristics like the persistence of interest plus a best-first search.

VI. CONCLUSIONS

The exponential growth of the World Wide Web makes information retrieval on the Internet more and more difficult. Search engines and robots are useful for narrowing the user's search so that it is faster and more efficient. However, a search engine can still easily return a large number of Web pages for a single search, and it is time consuming for the user to go through the list of pages just to find the information. To address this problem, advanced information exploration with data-mining capacities is a direction for research and development.

In this chapter, we have introduced the two most popular Web browsers (Netscape Communicator and Internet Explorer), and some leading search engines. We have also reviewed some research efforts on advanced Web information exploration using data-mining facilities.

ACKNOWLEDGMENTS

The paper has benefited from discussions and joint work with Daniel Ngu at Monash University.

REFERENCES

1. Deleted in proof.
2. Deleted in proof.
3. Altavista, http://www.altavista.com, 1999.
4. AOL NetFind, http://www.aol.com/netfind/, 1999.
5. AskJeeves, http://www.askjeeves.com, 1999.
6. Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T. Web Watcher: A learning apprentice for the World Wide Web. Available at http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/webagent-plus/webagent-plus.html, 1995.
7. Balabanovic, M., and Shoham, Y. Learning information retrieval agents: Experiments with automated Web browsing. In *On-line Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments,* 1995.
8. Barlow, L. The Spider's apprentice—How to use Web search engines: How search engines work. Available at http://www.monash.com/spidap4.html, Nov. 1997.
9. Berners-Lee, T., Gailliau, R., Luotonen A., Nielsen, H. F., and Secret, A. The World-Wide Web. *Commun. ACM* 37(8): 1994.
10. Bowman, C. M., Danzig, P. B., Hardy, D. R., Manber, U., and Schwartz, M. F. The Harvest information discovery and access system. In *Proceedings of the Second International World-Wide Web Conference,* Chicago, IL, Oct. 1994.
11. Bowman, C. M., Manber, U., Danzig, P. B., Schwartz, M. F., Hardy, D. R., and Wessels, D. P. Harvest: A scalable, customizable discovery and access system. Technical report, University of Colorado—Boulder, Mar. 1995.
12. Deleted in proof.
13. Campbell, M. An evaluation of indexing techniques for text-based information retrieval systems. Master's thesis, Chisolm Institute of Technology, Nov. 1989.
14. Chang, C.-H., and Hsu, C.-C. Customizable multiengine search tool with clustering. In *Proceedings of Sixth International World Wide Web Conference,* 1997.
15. Cho, J., Garcia-Molina, H., and Page, L. Efficient crawling through URL ordering. In *Proceedings of Seventh International World Wide Web Conference,* 1998.
16. Dogpile, http://www.dogpile.com, 1999.
17. Drelinger, D., and Howe, A. E. Experiences with selecting search engines using metasearch. *ACM Trans. Inform. Systems* **15**(3): 195–222, 1997.
18. Excite Inc., Excite, http://www.excite.com, 1999.
19. Faloutsos, C. Access methods of text. *Comput. Surveys* March 1985.
20. Google, http://www.google.com, 1999.
21. Gordon, M. Probablistic and genetic algorithms for document retrieval. *Commun. ACM* Oct. 1998.
22. Deleted in proof.
23. Infoseek, http://www.infoseek.com, 1999.
24. Inktomi, http://www.inktomi.com/products/search/, 1999.
25. He, J. Search engines on the Internet. *Experiment. Tech.* 34–38, Jan./Feb. 1998.
26. HotBot, http://www.HotBot.com, 1999.
27. Joachims, T., Mitchell, T., Freitag, D., and Armstrong, R. WebWatcher: Machine learning and hypertext. Available at http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/webwatcher/mltagung-e.ps.Z, 1995.
28. Jones, S. *Text and Context: Document Storage and Processing.* Springer-Verlag, London, 1991.
29. Koster, M. ALIWEB—Archie-like indexing in the Web. In *Proceedings of the First International World-Wide Web Conference,* Geneva, Switzerland, May 1994.
30. Koster, M. Robots in the Web: Threat or treat? *ConneXions* 9(4): Apr. 1995.
31. Koster, M. Guidelines for Robot Writers, http://www.robotstxt.org/wc/guidelines.html.
32. Koster, M. A Standard for Robot Exclusion http://www.robotstxt.org/wc/norobots.html.
33. Lieberman, H. Letizia: An agent that assists Web browsing. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligent,* Montreal, Canada, Aug. 1995.
34. Loke, S. W., Davison, A., and Sterling, L. CIFI: An intelligent agent for citation. Technical Report 96/4, Department of Computer Science, University of Melbourne, Parkville, Victoria 3052, Australia.

35. LookSmart, http://www.looksmart.com.au/, 1999.
36. Lowder, J. *Wide Area Selection for Searching Distributed Hyperdocuments.* Ph.D. thesis, Computer Science and Software Engineering, Monash University, Sept. 1997.
37. Lycos, http://www.lycos.com, 1999.
38. Marchiori, M. The quest for correct information on the Web: Hyper search engines. In *Proceedings of Sixth International World Wide Web Conference,* 1997.
39. McMurdo, G. How the Internet was indexed. *J. Inform. Sci.* **21:** 479–489, 1995.
40. Medical Word Search, http://mwsearch.com/, 1999.
41. Megallan, http://magellan.excite.com/, 1999.
42. MetaCrawler, http://www.go2net.com/search.html, 1999.
43. Deleted in proof.
44. MSN, http://www.msn.com, 1999.
45. Mukherjea, S., Hirata, K., and Hara, Y. Towards a multimedia World-Wide Web information retrieval engine. In *Proceedings of Sixth International World Wide Web Conference,* 1997.
46. Ngu, D. S. W., and Wu, X. Interest discovery for incremental Web exploration. In *Proceedings of WebNet 98: The 1998 World Conference of the WWW, Internet and Intranet,* Orlando, FL, Nov. 7–12, 1998.
47. Ngu, D. S. W., and Wu, X., SiteHelper: A localized agent that helps incremental exploration of the World Wide Web. *Comput. Networks ISDN Systems: Internat. J. Comput. Telecommun. Networking* 29(8): 1249–1255, 1997.
48. Northern Light, http://www.northernlight.com/index.html, 1999.
49. Pazzani, M., Nguyen, L., and Mantik, S. Learning from hotlists and coldists: Towards a WWW information filtering and seeking agent. In *Proceedings of IEEE 1995 Intl. Conference on Tools with AI,* 1995.
50. Pringle, G., Allison, L., and Dowe, D. L. What is a tall poppy among Web pages? In *Proceedings of Seventh International World Wide Web Conference,* 1998.
51. Porterfield, K.W. WWWW (What's a WorldWideWeb?). *Internet World* 20–22, May 1994.
52. ProFusion, http://www.profusion.com/, 1999.
53. Deleted in proof.
54. Riecken, D. Intelligent agents. *Commun. ACM* 37(7): 1994.
55. Salton, G. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison-Wesley, Reading MA, 1989.
56. SavvySearch, http://www.ebdir.net/savvysearch.html.
57. Schwartz, C. Web search engines, *J. Amer. Soc. Inform. Sci.* 49(11): 973–982, 1998.
58. Search Engine Watch, http://searchenginewatch.com, 1999.
59. Spertus, E. ParaSite: Mining structural information on the Web. In *Proceedings of Sixth International World Wide Web Conference,* 1997.
60. Tenopir, C., and Lundeen, G. *Managing You information.* Neal-Schuman, New York, 1988.
61. WebCrawler, http://www.webcrawler.com, 1999.
62. Westera, G. Comparison of search engine user interface capabilities. Available at http://lisweb.curtin.edu.au/staff/gwpersonal/compare.html, Nov. 1997.
63. Westera, G. On the edge of the abyss: Locating information in the vortex of the World Wide Web. In *Seventh Asain Pacific Specials, Health and Law Librarians Conferences,* Oct. 1997.
64. Wiggins, R. W. Webolution: The evolution of the revolutionary World Wide Web. *Internet World.* 35–38, Apr. 1995.
65. World Wide Web Worm, http://www.inf.utfsm.cl/~vparada/html/wwww.html.
66. Deleted in proof.
67. Wu, S. and Manber, U. Fast text searching allowing errors, *Commun. ACM* 83–91, Oct. 1992.
68. Yahoo, http://www.yahoo.com, 1999.
69. Yan, T. W., Jacobsen, M., Garcia-Molina, H., and Dayal, U. From user access patterns to dynamic hypertext linking. In *Proceedings of the Fifth International World Wide Web Conference,* Paris, France, May 1996.
70. Zakon, R. H. Hobbess Internet Timeline v5.2. Available at http://info.isoc.org/guest/zakon/Internet/History/HIT.html.
71. Netscape, http://home.netscape.com, 1999.

This Page Intentionally Left Blank

19

ASYNCHRONOUS TRANSFER MODE (ATM) CONGESTION CONTROL IN COMMUNICATION AND DATA NETWORK SYSTEMS

SAVERIO MASCOLO

Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 70125 Bari, Italy

MARIO GERLA

Computer Science Department, University of California—Los Angeles, Los Angeles, California 90095

Database and Data Communication Network Systems, Vol. 3

Flow and congestion control is a major research issue in high-speed communication data networks. Link propagation delays along with higher and higher transmission speeds increase the bandwidth-delay product and have an adverse impact on the stability of closed loop control algorithms. Moreover, the growth of data traffic along with the ever-increasing demand of quality of service require more sophisticated congestion control techniques in order to manage the sharing of network resources. In this chapter, we propose classical control theory as a simple and powerful tool for tackling the issue of congestion control in high-speed communication networks. In particular, we propose the *Smith principle* for designing a congestion control algorithm for high-speed asynchronous transfer mode (ATM) data networks. Smith's principle is a classical control technique for time-delay systems. A major advantage of Smith's principle is that it allows the design problem for a system with delay being transformed into one without delay. In our case, we transform the dynamics of the controlled system into a simple first-order dynamics with a delay in cascade. The network supplies queue levels as a feedback to the sources, which, in turn, execute the control algorithm. The dynamics of the controlled data networks are analyzed using standard Laplace transform techniques. It is shown that it is necessary to provide a minimum buffer capacity at the switches equal to the bandwidth-delay product to guarantee *full link utilization* during transients, whereas under steady-state condition full link utilization can be obtained with any small buffer capacity. Moreover, *data loss prevention during both transient and steady-state conditions* can be guaranteed with any buffer capacity at the expense of less than full link utilization. A discrete time implementation of the control algorithm for ATM networks, which shows that a multiplicative decrease algorithm is required due to the fact that, in real communication networks, the delivery of feedback information can never be guaranteed at a fixed rate, is presented. Finally the advantages of the proposed algorithm in comparison with a popular explicit rate indication algorithm are discussed, and computer simulations are carried out.

I. INTRODUCTION

Nowadays, communication networks are among the fastest-growing engineering areas. This growth is fueled by the rapid progress in computer and communications technology and by the extraordinary increase in productivity generated by improved communications.

The first developed communication network was the telephone network (Alexander Bell, 1876). In telephone networks a *circuit* is set up from the calling party to the called party when a telephone call is started. This *circuit* is exclusively allocated to the phone conversation. Only one telephone conversation can be transmitted along one circuit. The switching of the circuits occurs at the beginning of a new telephone call (*circuit switched networks*) [1].

The key innovations in computer or data networks are the organization of data in packets and the *store-and-forward packet switching*. To send a packet from computer A to computer F, computer A puts the source address A and the destination address F into the packet header and sends the packet first to

computer *B*. When *B* gets the packet from *A*, it reads the destination address and finds out that it must forward the packet to *C* and so on. Thus, when a node receives a packet, it first stores it, then forwards it to another node. Using *store-and-forward packet switching*, links can be efficiently shared by a large number of intermittent transmissions. This method is called *statistical multiplexing*, and it contrasts with circuit switching that reserves circuits for the duration of the conversation even though the parties connected by the circuits may not transmit continuously. The reduction in delivery time due to the decomposition of data in small packets is called *pipelining gain* [1].

The U.S. Department of Defense Advanced Research Projects Agency (DARPA) have promoted the use of packet switched networks since the late 1960s. The resulting network, ARPANET, started operation in 1969 to evolve into the Internet, a set of interconnected networks that are used today by millions of people to exchange text, audio, video clips, documents, etc.

A major advantage of packet switched networks is the sharing of network resources. This sharing drastically reduces the cost of communications. A consequence is that sophisticated mechanisms of flow and congestion control are required to manage resource-sharing, without incurring congestion phenomena [1–4].

An increasing amount of research is devoted to different control issues, concerned with the goal of ensuring that users get their desired quality of service (QoS) [1,4–7].

The deployment of new communication networks that merge the capabilities of telephone and computer networks in order to transmit multimedia traffic over a fully integrated universal network has lead to the introduction of broadband integrated service digital networks (B-ISDNs). The emerging asynchronous transfer mode (ATM) technology has been selected as the transfer mode to be used in B-ISDNs [1,3].

ATM networks seek to provide an end-to-end transfer of fixed size cells (53 bytes) and with specified quality of service [1,3,8]. ATM is a class of *virtual circuit switching* networks conceived to merge the advantages of circuit switched technology (telephone networks), with those of packet switched technology (computer networks). In particular, ATM networks are connection-oriented in the sense that before two systems on the network can communicate, they should inform all intermediate switches about their service requirements and traffic parameter by establishing a *virtual circuit*. This is similar to the telephone networks, where an exclusive circuit is setup from the calling party to the called party, with the important difference that, in the case of ATM, many virtual circuits can share network links via *store-and-forward packet switching* [1].

The ATM Forum Traffic Management Group [8] defines five service classes for supporting multimedia traffic: (1) the *constant bit rate* (CBR) class, which is conceived for applications such as telephone, video conferencing, and television; (2) the *variable bit rate* (VBR) class, which allows users to send at a variable rate. This category is subdivided into two categories: real-time VBR (RT-VBR), and non-real-time VBR (NRT-VBR). An example of RT-VBR is interactive compressed video or industrial control (you would like a command sent to a robot arm to reach it before the arm crashes into something), while that of NRT-VBR is multimedia email. (3) The *unspecified bit rate* (UBR) class

is designed for those data applications, such as email and file transfer, that want to use any leftover capacity and are not sensitive to cell loss or delay. UBR does not require service guarantee and cell losses may result in retransmissions, which further increase congestion. (4) The *available bit rate* (ABR) class was defined to overcome this problem. It is the only class that responds to network congestion by means of a feedback control mechanism. This class is designed for normal data traffic such as file transfer and email. It does not require cell transfer delay to be guaranteed. However, the source is required to control its rate in order to take into account the congestion status of the network. In this way the cell loss ratio (i.e., lost cells/transmitted cells) is minimized, and retransmissions are reduced, improving network utilization [1,3,8].

Congestion control is critical in both ATM and Internet networks, and it is the most essential aspect of traffic management. In the context of ATM networks, binary feedback schemes were first introduced due to their easy implementation [9–13]. In these schemes, if the queue length in a switch is greater than a threshold, then a binary digit is set in the control management cell. However, they suffer serious problems of stability, exhibit oscillatory dynamics, and require large amounts of buffer in order to avoid cell loss. As a consequence, explicit rate algorithms have been largely considered and investigated [3]. Most of the existing explicit rate algorithms lack two fundamental parts in the feedback control design: (1) the analysis of the closed loop network dynamics; (2) the interaction with VBR traffic. In [14,15], an algorithm that computes input rates dividing the measured available bandwidth by the number of active connections is proposed. In [9], an analytic method for the design of a congestion controller, which ensures good dynamic performance along with fairness in bandwidth allocation, has been proposed. However, this algorithm requires a complex on-line tuning of control parameters in order to ensure stability and damping of oscillations under different network conditions; moreover, it is difficult to prove its global stability due to the complexity of the control strategy. In [16], a dual *proportional derivative* controller has been suggested to make easier the implementation of the algorithm presented in [9]. In [17], two linear feedback control algorithms have been proposed for the case of a *single connection* with a *constant service rate*. In [18], these algorithms have been extended to the case of multiple connections with the same round trip delay sharing the bottleneck queue, and the robustness of these algorithms for nonstationary service rate has been analyzed. In [19], the ABR source rate is adapted to the low-frequency variation of the available bandwidth; a linear dynamic model is presented, and H_2 optimal control is applied to design the controller. The controller is simple and its stability is mathematically shown. In [20], a single controlled traffic source, sharing a bottleneck node with other sources, is considered. The traffic is modeled by an ARMA process, whereas an H_∞ approach is used for designing the controller. In [21] a control scheme based on Smith's principle is proposed assuming that a *first in, first out* (FIFO) buffering is maintained at switch output links.

In this chapter, we state the problem of congestion control in a general packet switched network using classical control theory. A fluid model approximation of the network, where packets are infinitely divisible and small, is assumed. The advantages of assuming per-VC queuing versus FIFO queuing

are discussed. The dynamic behavior of each network queue in response to data input is modeled as the cascade of an integrator with a time delay. Then a controller based on Smith's principle is designed. A major advantage of the proposed control is that it is *effective over paths with any bandwidth-delay product,* including, f. i., large delay satellite wireless links. It ensures: (a) stability, i.e., no cell losses; (b) full and fair utilization of network links in the presence of multiple connections, with different round trip delays, which share the network; (c) exponential convergence of input rates to stationary values without oscillations or overshoots; and (d) "efficient coexistence" of quality-constrained services with "best effort" service.

Differently from [17,18], where links with constant available bandwidth are assumed, herein links with *time-varying available bandwidth* are considered in order to model the interaction of best effort traffic with quality constrained traffic. Moreover, since it is difficult to measure the available bandwidth, this bandwidth is modeled as a *disturbance* input.

A complete mathematical analysis of transient and steady-state dynamics, which shows that network queues are always bounded and greater than 0 in the presence of a *worst case disturbance,* which models the bandwidth leftover after time-varying, high-priority traffic has been served, is developed.

The application to ATM networks shows the advantages of the proposed control in comparison with explicit rate algorithms, such as the well-known explicit rate indication algorithm (ERICA) [15].

The chapter is organized as follows: Section II describes the data network model; Section III models the controlled data networks using classical control theory; in Section IV the controller is designed using Smith's principle; in Section V transient and steady-state dynamics are evaluated via mathematical analysis; Section VI describes the application of the proposed control law to ATM Networks; Section VII reports computer simulation results and a comparison with the explicit rate indication algorithm. Finally, Section VIII draws the conclusions.

II. THE DATA NETWORK MODEL

In this section we develop the model of a general network that employs a *store-and forward packet switching service;* that is, cells enter the network from the source edge nodes, and are then stored and forwarded along a sequence of intermediate nodes and communication links, finally reaching their destination nodes [1,4,9]. A network (see Fig. 1) can be considered as a graph consisting of:

(a) A set $N = \{n_i\}$ of nodes (switches) that store and forward the packets along the communication path. A node consists of a set of input queues where incoming packets are stored and of a set of output queues where outgoing packets are stored. Each node is characterized by the processing capacity $1/t_{pri}$ (packets/s) where t_{pri} is the time the switch i needs to take a packet from the input and place it on the output queue. It is assumed that the processing capacity of each node is larger than the total transmission capacity of its incoming links so that congestion is caused by transmission capacity only.

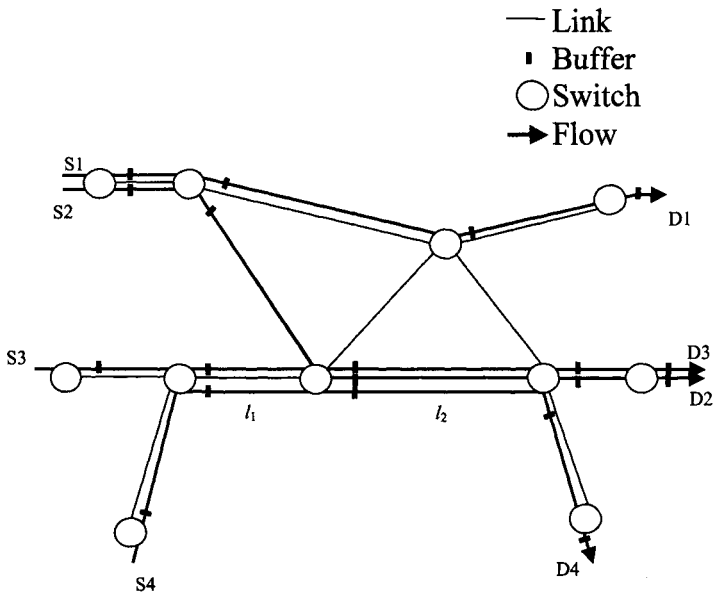


FIGURE 1 Store and forward packet switched network. Four (S_i, D_i) VC_i connections with per-flow queueing are shown.

(b) A set $L = \{l_i\}$ of communication links that connect the nodes to permit the exchange of information. Each link is characterized by the transmission capacity $c_i = 1/t_i$ (packets/s) and the propagation delay t_{di}. For each node $i \in N$, let $O(i) \subset L$ denote the set of its outgoing link and let $I(i) \subset L$ denote the set of its incoming links.

The network traffic is contributed by source/destination pairs $(S_i, D_i) \in N \times N$. To each (S_i, D_i) connection is associated a *virtual circuit* (VC_i) or a *flow* i mapped on the path $p(S_i, D_i) = (n_1, n_2, \ldots, n_n)$.

A deterministic fluid model approximation of packet flow is assumed; that is, each input flow is described by the continuous variable $u(t)$ measured in cells per second.

In high-speed networks, the *bandwidth–delay product* t_{di}/t_i is a key parameter that affects the stability of closed loop control algorithms. It represents a large number of packets "in flight" on the transmission link. These packets are also called *in pipe cells.*

A. Advantages of Per-Flow Queuing

In this section we discuss the advantages of using switches, which maintain per-flow queuing at the output link. *Per-VC* FIFO queuing has two important implications: (1) fairness can be easily enforced and (2) per-flow queue dynamics are uncoupled.

Per-flow buffering separates cells according to the flow to which they belong [4,22,23]. Thus, it is easy for a switch to enforce *fairness.* In fact, as shown in Fig. 2, the link bandwidth leftover by high-priority traffic can be assigned to the

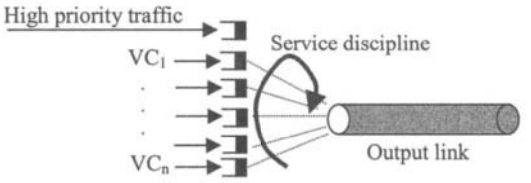


FIGURE 2 Output link shared by n ABR flows and high-priority traffic (CBR + VBR). Per-VC buffering is shown. The switch services packets belonging to different flows in accordance to a given service discipline.

ABR flows in accordance with any definition of fairness by means of a scheduling mechanism executed at the switch, f. i. via a round-robin service discipline of cells belonging to different queues. Thus, each VC gets a fair amount of the "best effort" available bandwidth. At the same time, the congestion control algorithm must ensure the full utilization of this bandwidth without incurring cell loss. Note that the assumption of per-VC queuing leads to separating the mechanism for ensuring fairness (i.e., the *scheduling at the switch*) and the mechanism for avoiding congestion (i.e., the *control algorithm executed at the source*). Figure 1 shows a network where switches allocate their memory on a per-flow basis, whereas output links are shared by different flows.

Link sharing determines that the per-flow available bandwidth is time varying; that is, each flow sees the effect of other flows only through a time-varying available bandwidth $d(t)$. In this way, *the dynamics of each per-flow queue level* in response to its input rate *is uncoupled by other flows*. The bandwidth that a generic virtual circuit i can utilize is the minimum of available bandwidth over all the links belonging to its communication path. This minimum is called the bottleneck available bandwidth, and the buffer feeding this link is the bottleneck queue. Note that this queue is the only queue with a level greater than 0 along the considered VC path; that is, it is the only queue level that must be controlled. For these considerations, the control problem design is a single-input/single-output (SISO) problem design, where the input is the rate of the VC connection and the output is the per-flow queue level that is the bottleneck for the considered VC. A major consequence of the fact that the flows are uncoupled is that any coexisting traffic scenario influences a single VC only through a time-varying available bandwidth at the bottleneck link.

Figure 3 shows a single VC extracted from the network shown in Fig. 1. T_{fw}, is the forward propagation delay from the source to the bottleneck buffer that feeds the bottleneck link l_b whereas T_{fb} is the backward propagation delay from

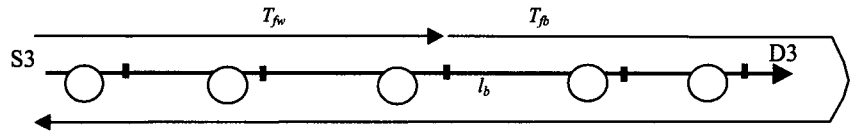


FIGURE 3 A single VC connection (S3,D3) with the bottleneck link $l_b = l_2$ is shown. The interaction with all other flows shown in Fig. 1 is completely captured by means of the unknown and bounded disturbance $d_{3,2}(t)$.

the bottleneck to the destination and then back to the source. The interaction of the considered flow with all other flows in the network is completely captured by means of the time-varying bandwidth available at the bottleneck link. The drawbacks of per-flow queuing is that it is not scalable, and it may be costly to be implemented.

B. A Classical Control Approach to Model the Input–Output Dynamics of a Per-VC Queue

The methodology of classical control theory to design a feedback control law consists of two steps: (1) A mathematical model of the input–output dynamics of the system to be controlled is derived in the form of a nth-order differential equation; and (2) a control law is designed to satisfy some given performance requirements.

In our case, the input of the system is the VC input rate and the output is the VC bottleneck queue level. To derive the input–output mathematical model of a per-VC queue, we note that a queue makes the integration of the input rate minus the output rate. Thus, by letting $x_{ij}(t)$ be the queue level associated with the VC_i connection (S_i, D_i) and the bottleneck link l_j, and by writing the flow conservation equation, the queue level $x_{ij}(t)$, starting at $t = 0$ with $x_{ij}(0) = 0$, results in

$$x_{ij}(t) = \int_0^t [u_i(\tau - T_{ij}) - d_{ij}(\tau)]d\tau, \tag{1}$$

where $u_i(t)$ is the queue inflow rate due to the ith VC, T_{ij} is the propagation delay from the ith source to the jth queue, and $d_{ij}(t)$ is the rate of packets leaving the jth queue, that is, the bandwidth left available at link l_j for the ith VC. Note that the available bandwidth $d_{ij}(t)$ depends on the global traffic loading the link; that is, it is the best effort capacity available for the ith VC. Since this bandwidth depends on coexisting traffic, we assume that its amount is unknown at the VC source. In control theory an unknown input that cannot be handled is called *disturbance input.*

The operation of integration is *linear.* Equation (1) can be transformed to an algebraic equation by using the Laplace transform as[1]

$$X_{ij}(s) = \frac{U_i(s)e^{-sT_{ij}} - D_{ij}(s)}{s}. \tag{2}$$

Figure 4 shows the block diagram of Eq. (2).

The transfer function of a linear system describes the input–output dynamics of the system in the Lapalce domain. Therefore Eq. (2) gives two input–output transfer functions:

$$\frac{X_{ij}(s)}{U_i(s)} = \frac{e^{-sT_{ij}}}{s} \text{ and } \frac{X_{ij}(s)}{D_{ij}(s)} = -\frac{1}{s}.$$

The objective of the control is to throttle the VC_i input rate $u_i(t)$ so that the VC_i bottleneck queue $x_{ij}(t)$ is bounded (this guarantees cell loss prevention) and

[1]The Laplace transform $L(\cdot)$ of the integral $\int_o^t u(\tau)d\tau$ is obtained by multiplying by $1/s$ the Laplace transform of $u(t)$; that is, $L(\int_o^t u(\tau)d\tau) = L(u(t))/s = U(s)/s$. The Laplace transform of the delayed signal signal $u(t - T)$ is obtained by multiplying by e^{-sT} the Laplace transform of $u(t)$; that is, $L(u(t - T)) = U(s)e^{-sT}$.

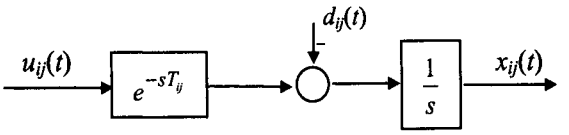


FIGURE 4 Block diagram of the bottleneck queue dynamics of a VC.

greater than 0 (this guarantees full link utilization) in the presence of an unknown and bounded disturbance $d_{ij}(t)$, which models the time-varying leftover bandwidth.

We choose to execute the control algorithm at the source, whereas the bottleneck queue level is fed back to the source from the switches. This choice is motivated by the fact that the VC source is the best place to look at where the bottleneck link of the VC path is. The mechanism to implement the feedback is that proposed by the ATM Forum [8]. In this scheme an ABR source sends one control cell (resource management (RM) cell) every NRM data cell. Each switch encountered by the RM cell along the VC path stamps on the RM cell the free buffer space if and only if this value is smaller than that already stored. At the destination, the RM cell carries the minimum available space over all the encountered buffers (i.e., the bottleneck-free space), and it comes back to the source conveying this value. Upon receiving this information, the source updates its input rate.

III. A CLASSICAL CONTROL APPROACH TO MODEL A FLOW-CONTROLLED DATA NETWORK

We assume that a connection establishes a virtual circuit and that per-VC buffering is maintained at the switch output links. A particular link along the VC path will be the bottleneck. The queue feeding this link will be the bottleneck queue for the VC, whereas other queues passed through by the VC will be empty. Thus, the control goal reduces to fully utilize the bottleneck link, whereas overflow of the bottleneck queue must be avoided. Since we have assumed per VC queuing, the connection bottleneck queue will be filled only by the input rate of the corresponding VC. As a consequence, the dynamics of the bottleneck queue level in response to the source input rate is a *single-input single-output dynamics*.

The block diagram of a closed-loop controlled flow and its interaction with network traffic is shown in Fig. 5. In particular, it consists of:

1. An integrator described in the Laplace domain by the transfer function $1/s$. It models the considered per-VC buffer. The output $x_{ij}(t)$ is the bottleneck queue length.
2. A disturbance $d_{ij}(t)$, which models the bandwidth that is available for the considered VC_i. $d_{ij}(t)$ is the bandwidth left unused by other flows sharing the bottleneck link l_j. Note that the "best effort" available bandwidth $d_{ij}(t)$ is modeled as an unknown deterministic function rather than a stochastic process, and that, by means of this disturbance $d_{ij}(t)$, all traffic interacting with the considered flow is easily modeled.

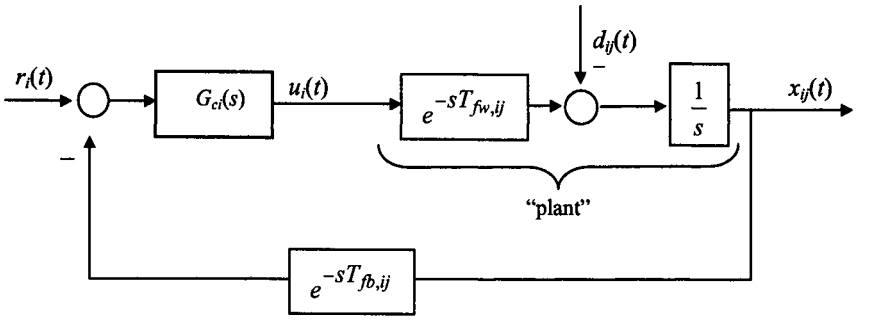


FIGURE 5 Block diagram of the controlled VC flow.

3. The transfer function $e^{-sT_{\mathrm{fw},ij}}$, which models the forward propagation time $T_{\mathrm{fw},ij}$ from the VC_i, source to the bottleneck queue.
4. The transfer function $e^{-sT_{\mathrm{fb},ij}}$, which models the backward propagation time $T_{\mathrm{fb},ij}$ from the bottleneck queue to the destination and then back to the source.
5. The controller transfer function $G_{ci}(s)$.
6. The VC_i input rate $u_i(t)$.
7. The reference signal $r_i(t)$.

The feedback control scheme consists of two inputs: the reference signal $r_i(t)$ and the disturbance $d_{ij}(t)$. Since the controlled system is linear, we can determine the output queue length by superposing the responses to each input.

Referring to the reference signal $r_i(t)$, the source compares $r_i(t)$ with the delayed bottleneck queue level $x_{ij}(t - T_{\mathrm{fb},ij})$ and then inputs the difference into the controller $G_{ci}(s)$, which calculates the input rate. The input rate reaches the queue after the propagation delay $T_{\mathrm{fw},ij}$, whereas the queue level reaches the destination and is relayed back to the source after the time $T_{\mathrm{fb},ij}$. Note that the *round trip time* (RTT_i) of the VC_i connection is always $\mathrm{RTT}_i = T_{fw,ij} + T_{fb,ij}$ anywhere the connection bottleneck link j may be positioned along the VC_i path. This means that the proposed *control scheme models also the general case of bottleneck that moves along the VC path*. Note that, in case the contribution of the bottleneck queuing time to the RTT cannot be neglected, then the queuing time can be considered part of $T_{\mathrm{fb},ij}$.

Referring to the disturbance $d_{ij}(t)$, it directly empties the bottleneck queue. The goal of the control law is to ensure full link utilization and cell loss prevention in the presence of time-varying available bandwidth $\mathrm{d}_{ij}(t)$.

Due to the possibly large propagation delays in the control loop, queue level dynamics might exhibit oscillations, and even become unstable. Therefore, the design of the linear controller $\mathrm{G}_{ci}(s)$ must be carried out carefully.

A. Why a Linear Feedback Control Scheme Is Proposed

We have seen that the idea of using ABR best-effort traffic to fully utilize ATM network bandwidth led to the introduction of closed-loop congestion control algorithms. Binary feedback schemes where, if the queue level of a switch is greater than a threshold, then a binary digit is set in the RM cell were first

proposed [9–13]. As a consequence, the dynamics of the controlled system becomes *nonlinear* even if, in our case, the dynamics of the system to be controlled is linear. Moreover, since the content of information delivered by the binary feedback is too poor, problems of stability and performance arise.

Following these considerations and noting that RM cells have enough room to store and convey the available buffer space as feedback information, we propose a linear feedback control so that the controlled system keeps linear (see Fig. 5). Linear systems have many appealing properties, and a complete and well-established control theory exists for them.

B. The Reference Signal and the Worst Case Disturbance

The controlled system reported in Fig. 5 is a SISO system with a disturbance. The input is the reference signal $r_i(t)$, which sets a threshold for the bottleneck queue length. The output is the queue level $x_{ij}(t)$. We start by considering the reference signal $r_i(t) = r_i^o \cdot 1(t - T_{fb})$, where $1(t)$ is the step function.[2] In other words, the source receives the feedback $r_i^o \cdot 1(t - T_{fb}) - x_{ij}(t - T_{fb})$ from the bottleneck. This value is the space that is free at the bottleneck buffer.

The bottleneck link transmission capacity is normalized to unity so that, if all link bandwidth is suddenly available for a single flow a $t = t_o$, then the disturbance $d_{ij}(t)$ is equal to the step function $1(t - t_o)$. The coexistence with other flows reduces the bandwidth available for the considered flow. Letting $b(t) \leq 1$ be the bandwidth used by competing flows and $b_m = \min_t\{b(t)\}$, it results in

$$0 \leq d_{ij}(t) \leq 1(t) - b_m \cdot 1(t) = a \cdot 1(t),$$

where $0 \leq b_m \leq 1$ and $a = (1 - b_m) \leq 1$.

In this way, the available bandwidth is modeled via a deterministic, unknown, and bounded disturbance function $d_{ij}(t)$ upper bounded by the *worst case disturbance* $a \cdot 1(t)$.

Remark 1. The control scheme reported in Fig. 5 models the dynamics of the bottleneck queue level $x_{ij}(t)$ in response to the flow input rate $u_i(t)$ and the unknown leftover link capacity $d_{ij}(t)$. The scheme is completely general; that is, it is able to control congestion in all network topology and traffic scenario. In fact, due to per-VC queuing assumption, each VC senses other flows only through $d_{ij}(t)$. Note that, nowadays, many switch vendors are implementing per-VC queuing, and per-flow buffering is also increasingly auspicated for Internet routers in order to ensure QoS [4,22,23].

IV. DESIGNING THE CONTROL LAW USING THE SMITH PRINCIPLE

A. The Smith Principle: Basic Concepts

The Smith principle is an important classical control technique for time-delay systems [24–26]. It is an effective dead-time compensator for a *stable* process with large time delay. Consider the goal of designing a controller $G_c(s)$ for the time delay system $G(s)\exp(-sT)$ shown in Fig. 6. Smith's principle pursues the

[2]The step function is $1(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$.

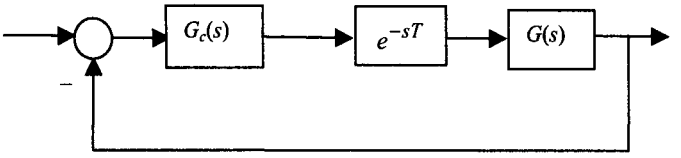


FIGURE 6 A closed loop time delay system.

goal of designing a controller $G_c(s)$ such that the resulting closed-loop dynamics is delay-free. More precisely, $G_c(s)$ is chosen so that the system becomes equivalent to the *reference* system reported in Fig. 7. This system consists of: (1) the delay-free "plant" $G(s)$; (2) the controller $K(s)$; and (3) the delay $\exp(-sT)$, which is out of the feedback loop.

The chosen reference system is appealing because it is a delay-free system whose output is delayed by the time T.

By equating the transfer functions of the systems in Fig. 6 and Fig. 7,

$$\frac{K(s)G(s)}{1+K(s)G(s)}e^{-sT} = \frac{G_c(s)G(s)e^{-sT}}{1+G_c(s)G(s)e^{-sT}},$$

the required controller $G_c(s)$ results in

$$G_c(s) = \frac{K(s)}{1+K(s)G(s)(1-e^{-sT})}. \tag{3}$$

The block diagram of the controller is reported in Fig. 8. It should be noted that, using the Smith principle, the problem of designing the controller for the time-delay system in Fig. 6 has reduced to the design of the controller $K(s)$ for the delay-free system in Fig. 7 [25]. Note that an accurate model of the plant $G(s)\exp(-sT)$ is necessary because this is part of the controller G_c.

B. The Proposed Control Law

The objective of the congestion control algorithm is to guarantee that the VC input rate promptly utilizes all available bandwidth even in the presence of a bandwidth suddenly available. At the same time, buffer overflow must be avoided. Formally, this can be stated saying that the VC input rate u_i must be throttled so that in the presence of the worst case disturbance $a\cdot 1(t)$, the two following conditions must be satisfied:

(1) *stability condition*

$$x_{ij}(t) \leq r^o \text{ for } t > 0, \tag{4}$$

which guarantees no packet loss at the bottleneck queue and

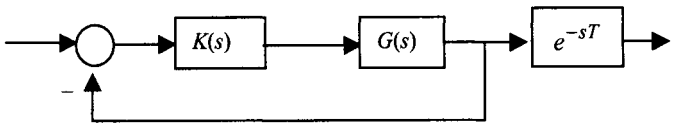


FIGURE 7 Desired input–output dynamics.

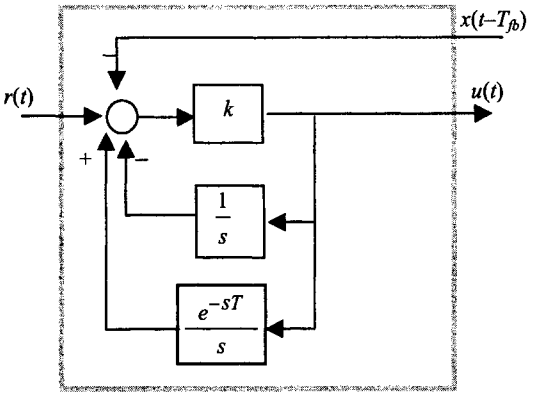


FIGURE 8 Block diagram of the controller.

(2) *full link utilization condition*

$$x_{ij}(t) > 0 \text{ for } t \geq T = \text{RTT}_i, \tag{5}$$

which guarantees full utilization of the bottleneck link because the bottleneck always has data to send. Clearly, T cannot be less than the round trip propagation time.

The system to be controlled (see Fig. 5) consists of an integrator with a delay in cascade. We assume that the delay is well known[3] so that a controller can be successfully designed following the Smith principle [24–26].

The main advantage of the Smith principle is that the time delay is eliminated from the characteristic equation of the closed-loop system. Thus the design problem for a process with delay can be transformed into one without delay. A problem is that it cannot be used for processes having an integral mode since a constant disturbance will result in a steady-state error. However, the rejection of the disturbance is not appropriate in the context of the design problem herein discussed. In fact, due to propagation time and to time-varying disturbance, it is never possible to reject the disturbance during the transient, which is the normal operation mode of communication networks. For these considerations, control goals are expressed by (4) and (5).

Following the Smith principle, the idea is to look for a controller $G_c(s)$ so that the input–output dynamics of the system reported in Fig. 5 becomes equal to the input–output dynamics of the system reported in Fig. 9. A major advantage of the system reported in Fig. 9 is that it is a simple *first-order system* with a delay in cascade. The first-order system is obtained by choosing a proportional controller k for the delay-free plant. As a consequence, letting the set point $r_i(t)$ be the step function $r_i^o \cdot 1(t - T_{\text{fb},ij})$, where r_i^o is the capacity of the bottleneck queue, the output exponentially converges to a steady state without oscillations or overshoots. More precisely, $x_{ij}(t)$ is the well-known response of

[3]In ATM networks, this can be guaranteed by giving priority to RM cells over data cells at the per-VC FIFO queues.

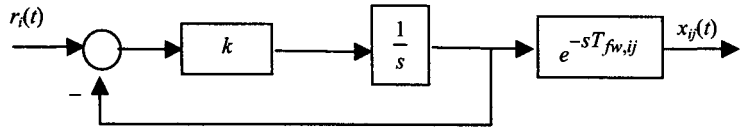


FIGURE 9 Desired input–output dynamics.

a first-order dynamic system to a step function, delayed by $T_{\mathrm{fw},ij}$, i.e.,

$$x_{ij}(t) = r_i^o\left(1 - e^{-k(t-T_{\mathrm{fb},ij}-T_{\mathrm{fw},ij})}\right) \cdot 1(t - T_{\mathrm{fb},ij} - T_{\mathrm{fw},ij})$$
$$= r_i^o\left(1 - e^{-k(t-\mathrm{RTT}_i)}\right) \cdot 1(t - \mathrm{RTT}_i),$$

where $\tau = 1/\mathrm{k}$ is the system time constant.

Now we are ready to derive the controller.

PROPOSITION 1. *The transfer functions X(s)/R(s)*[4] *of the systems reported in Fig. 5 and Fig. 9 are made equal by using the controller described by the transfer function*

$$G_c(s) = \frac{k}{1 + \frac{k}{s}(1 - e^{-\mathrm{RTT}\cdot s})}. \tag{6}$$

Proof. By equating the transfer functions of the systems reported in Fig. 5 and Fig. 9

$$\frac{\frac{G(s)}{s}e^{-T_{\mathrm{fw}}\cdot s}}{1 + \frac{G(s)}{s}e^{-\mathrm{RTT}\cdot s}} = \frac{\frac{k}{s}}{1 + \frac{k}{s}}e^{-T_{\mathrm{fw}}\cdot s},$$

the controller (6) is derived after a little algebra. ■

Looking at the controller (6) shown in Fig. 8, it is easy to write the rate control equation, that is,

$$u(t) = k\left(r(t - T_{\mathrm{fb}}) - x(t - T_{\mathrm{fb}}) - \int_{t-\mathrm{RTT}}^{t} u(\tau) \cdot d\tau\right). \tag{7}$$

The control equation (7) is very simple and can be intuitively interpreted as follows: the computed input rate is proportional, through the coefficient k, to the space free in the bottleneck queue, that is, $r(t - T_{\mathrm{fb}}) - x(t - T_{\mathrm{fb}})$, decreased by the number of cells released by the VC_i during the last round trip time RTT, i.e., the "in flight cells." By denoting the bottleneck-free space as

$$r(t - T_{\mathrm{fb}}) - x(t - T_{\mathrm{fb}}) = \mathit{Bottleneck\_free\_space}$$

and

$$\int_{t-\mathrm{RTT}}^{t} u(\tau) \cdot d\tau = \mathit{In\_flight\_cells},$$

[4]From now on, the subscripts in $x_{ij}(t)$, $d_{ij}(t)$, $G_{ci}(t)$, $u_i(t)$, RTT_i, which are used to refer the jth output link and the VC_i connection, are dropped.

Eq. (7) can be rewritten as

$$u(t) = k(Bottleneck\_free\_space - In\_flight\_cells). \tag{8}$$

In the next section we show via mathematical analysis that the proposed control law guarantees bottleneck queue stability and full link utilization during both transient and steady-state conditions.

V. MATHEMATICAL ANALYSIS OF STEADY-STATE AND TRANSIENT DYNAMICS

Classical control theory provides an established set of tools that enables us to design algorithms whose performance can be predicted analytically rather than relying on simulations. Thus, to analyze the performance of the proposed algorithm is sufficient to use standard Laplace transform techniques. The important advantage of mathematical analysis is that it allows us to demonstrate the properties of the proposed control law in a general setting, whereas the validation via computer simulations is restricted to the simulated scenarios. Note that the analysis of transient dynamics is extremely important in the context of communication networks because these systems never reach a steady-state condition due to continuous joining and leaving of connections.

A major result of the transient and steady-state analysis is that a minimum bottleneck buffer capacity is necessary to ensure full link utilization.

We start by showing that the controller (6) guarantees cell loss prevention.

PROPOSITION 2. *Considering the reference signal* $r(t) = r^o \cdot 1(t - T_{fb})$, *the disturbance* $d(t) = a_1 \cdot 1(t - T_1)$, *where* $T_1 \geq \mathrm{RTT}$ *is the instant when the bandwidth* a_1 *is suddenly available, and the controller (6), the bottleneck queue is bounded, that is,* $x(t) \leq r^o$ *for any* $t \geq 0$.

Proof. To analyze the controlled system dynamics, we compute the transfer functions $\frac{X_r(s)}{R(s)}$ and $\frac{X_d(s)}{D(s)}$, which, after a little algebra, result to be

$$\frac{X_r(s)}{R(s)} = -\frac{1}{(1 + s/k)} \cdot e^{-T_{fw}s} \tag{9}$$

$$\frac{X_d(s)}{D(s)} = -\frac{1}{s} + \frac{k}{s(s+k)} e^{-\mathrm{RTT}\cdot s} \tag{10}$$

The Laplace transform of $r(t)$ and $d(t)$ are, respectively,

$$R(s) = \frac{r^o}{s} e^{-T_{fb}\cdot s} \quad \text{and} \quad D(s) = \frac{a_1}{s} e^{-T_1 \cdot s}.$$

By transforming back to time domain $X_r(s)$ and $X_d(s)$, it results in

$$x_r(t) = r^o\left(1 - e^{-k(t-\mathrm{RTT})}\right) \cdot 1(t - \mathrm{RTT}) \tag{11}$$

$$x_d(t) = -a(t - T_1) \cdot 1(t - T_1) + a(t - T_1 - \mathrm{RTT}) \cdot 1(t - T_1 - \mathrm{RTT}) - \frac{a}{k}\left(1 - e^{-k(t-T_1-\mathrm{RTT})}\right) \cdot 1(t - T_1 - \mathrm{RTT}). \tag{12}$$

It can be noted that $0 < x_r(t) \leq r^o$ for $t >$ RTT, $x_r(\text{RTT} = 0, x_r(\infty) = r^o, x_d(t) < 0$ for $t > T_1$ and $x_d(T_1) = 0$. Therefore,

$$x(t) = x_r(t) + x_d(t) \leq x_r(t) \leq r^o \text{ for } t \geq 0,$$

that is, the stability condition is satisfied. ■

PROPOSITION 3. *Under steady-state conditions the bottleneck queue level is* $x(\infty) = x_s = r^o - a \cdot (\tau + \text{RTT})$.

Proof. From (11) and (12) it results in

$$x(\infty) = x_r(\infty) + x_d(\infty) = x_s = r^o - a \cdot (\tau + \text{RTT}). \blacksquare$$

Now we show that the bottleneck *link is fully utilized during both transient and steady-state conditions* if the bottleneck buffer capacity is at least equal to $a \cdot (\tau + \text{RTT})$.

PROPOSITION 4. *Considering the reference signal* $r(t) = r^o \cdot 1(t - T_{fb})$, *the disturbance* $d(t) = a_1 \cdot 1(t - T_1)$, *and the controller (6), the bottleneck link is fully utilized for* $t >$ RTT *if the bottleneck buffer capacity satisfies the condition*

$$r^o > a \cdot (\text{RTT} + \tau). \tag{13}$$

Proof. Computing the time derivative of (11) and (12) it results in

$$\dot{x}_r(t) = k \cdot r^o \cdot e^{-k(t-\text{RTT})} \cdot 1(t - \text{RTT})$$

$$\dot{x}_d(t) = -a \cdot 1(t - T_1) + a \cdot 1(t - T_1 - \text{RTT}) - a \cdot e^{-k(t-T_1-\text{RTT})} \cdot 1(t - T_1 - \text{RTT});$$

that is, $\dot{x}_r(t) > 0$ for $t \geq$ RTT, and $\dot{x}_d(t) < 0$ for $t \geq T_1$. Figure 10 shows a typical plot of $x_r(t)$, $x_d(t)$, and $x_r(t) + x_d(t)$.

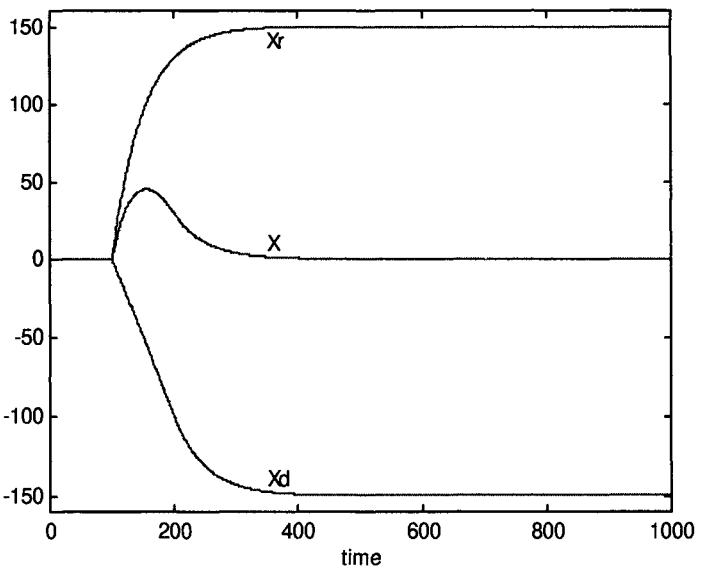


FIGURE 10 Time waveform $x_r(t)$ of the queue in response to the set point $r(t)$, time waveform $x_d(t)$ of the queue in response to the disturbance $d(t)$, and time waveform of the queue length $x(t) = x_r(t) + x_d(t)$.

In the time interval [RTT, T_1) it results in $\dot{x}(t) = \dot{x}_r(t) = k \cdot r^o \cdot e^{-k(t-\mathrm{RTT})} > 0$, whereas in the interval [T_1, RTT + T_1), it results in $\dot{x}(t) = k \cdot r^o \cdot e^{-k(t-\mathrm{RTT})} - a$. This derivative is zero at $t = \mathrm{RTT} + \frac{1}{k} \ln \frac{r^o}{a\tau}$, which corresponds to a positive maximum since $\ddot{x}(t) = -k^2 r^o e^{-k(t-\mathrm{RTT})} < 0$. Clearly, a condition for the existence of this maximum is that $T_1 \leq \mathrm{RTT} + \frac{1}{k} \ln \frac{r^o}{a\tau} < T_1 + \mathrm{RTT}$. If there is no zero for $\dot{x}$, because $T_1 > \mathrm{RTT} + \tau \cdot \ln \frac{r^o}{a\tau}$, then $\dot{x}(T_1^-) = k \cdot r^o \cdot e^{-k(T_1-\mathrm{RTT})} > 0$ and $\dot{x}(T_1^+) = k \cdot r^o \cdot e^{-k(T_1-\mathrm{RTT})} - a < 0$. Thus $t = T_1$ is still a point of maximum. If there is no zero for $\dot{x}$ because $T_1 \leq \frac{1}{k} \ln \frac{r^o}{a\tau}$, then $x(t)$ is strictly increasing in [T_1, RTT +T_1).

In the interval [RTT + T_1, ∞) it results in

$$\dot{x}(t) = k \cdot r^o \cdot e^{-k(t-\mathrm{RTT})} - a \cdot e^{-k(t-T_1-\mathrm{RTT})} = (k \cdot r^o - a \cdot e^{kT_1}) e^{-k(t-\mathrm{RTT})}.$$

This derivative is zero for any t if and only if $T_1 = \frac{1}{k} \ln \frac{r^o}{a\tau}$, whereas it is strictly positive if $T_1 < \frac{1}{k} \ln \frac{r^o}{a\tau}$ and strictly negative if $T_1 > \frac{1}{k} \ln \frac{r^o}{a\tau}$. Therefore, it can be concluded that if $T_1 \leq \frac{1}{k} \ln \frac{r^o}{a\tau}$, then $x(t)$ is positive, always increasing and upper bounded by $x(\infty) = r^o - a \cdot (\tau + \mathrm{RTT}) > 0$. On the other hand, if $T_1 > \frac{1}{k} \ln \frac{r^o}{a\tau}$, then $x(t)$ has a positive maximum for $t_m = \mathrm{RTT} + \frac{1}{k} \ln \frac{r^o}{a\tau}$ or for $t_m = T_1$.[5] Proposition 2 ensures that this maximum is less than r^o. Considering that $\dot{x}(t) < 0$ for $t > t_m$ and $x(\infty) > 0$, it follows that $x(t)$ is lower bounded by $x(\infty)$; that is, $x(t) \geq x(\infty) > 0$ for $t > t_m$. Thus, it results in $x(t) > 0$ for t > RTT; i.e. *full link utilization* is shown during transient and steady-state conditions if a buffer capacity at least equal to $r^o = a \cdot (\tau + \mathrm{RTT})$ is provided. ■

Remark 2. Note that, assuming $r^o = a \cdot (\tau + \mathrm{RTT})$, it results in $x(\infty) = r^o - a \cdot (\tau + \mathrm{RTT}) = 0$; i.e., the steady-state queue level is zero. Moreover, note that the control law (7) gives the steady-state input rate $u(\infty) = k(a \cdot (\tau + \mathrm{RTT}) - u(\infty) \cdot \mathrm{RTT})$, i.e., $u(\infty) = a$. This means that, under steady-state conditions, full link utilization can be obtained without buffering.

VI. CONGESTION CONTROL FOR ATM NETWORKS

In this section the control law (7) is applied to control ABR traffic in ATM networks. Moreover, a comparison with the explicit rate indication algorithm (ERICA) is carried out [15].

To implement the feedback control of ABR traffic in ATM networks, an ABR source interleaves data cells with control cells, called resource management cells. At the destination, control cells are relayed back to the source carrying the feedback information.

It is worth noting that the algorithm we have described in Section IV could be run at the switches too (see [21]). In this case a switch would stamp the computed rate, i.e., the explict rate in the RM cell. The RM with the minimum explicit rate would reach the destination and would be relayed back to the source. A drawback of this implementation is that a switch along the path does not know the minimum explicit rate, i.e., the bottleneck rate, since a switch downstream could set this rate. Therefore, it would

[5] For $t_m = T_1$ the maximum is $x(t_m) = r^o(1 - e^{-k(T_1-\mathrm{RTT})})$, whereas for $t_m = \mathrm{RTT} + \tau \ln \frac{r^o}{a\tau}$ the maximum is $x(t_m) = r^o - a \cdot (\mathrm{RTT} + \tau) + a \cdot (T_1 - \tau \ln \frac{r^o}{a\tau})$.

be difficult for a switch to compute the "in-flight cells." In contrast, the source does not need to determine where the bottleneck switch is and can easily compute the number of "in-flight cells." The ATM Forum prescribes that an ABR source must send one RM cell every NRM data cell (NRM = 32), for conveying the feedback information. We assume that RM cells have priority over data cells at the queue. With this assumption the queuing time is zero, and as a consequence, the *round trip time is constant* and equal to the propagation time. The round trip time is measured when the VC is established by marking a RM cell with a timestamp. Each switch encountered by the RM cell along the VC path stamps on the RM cell the feedback information, which is the available buffer space $(r^o(t - T_{fb}) - x(t - T_{fb}))$. At the destination, the RM cell comes back to the source conveying the minimum available buffer space encountered along the path. Upon receiving of this information, the source updates its input rate [8].

Note that the feedback information is relayed in cells and thus not available in continuos time, but rather in sampled form. This is not a problem since the discrete time implementation of the Smith predictor is simpler than the continuos one [27]. A problem arises because the feedback supplied by RM cells is not received at a constant rate but rather at a frequency proportional to the input rate (i.e., at the frequency *input rate/NRM*). This is not the ideal situation since digital control theory assumes that signals are sampled at a constant rate. For this consideration, NRM should be a function of the input rate u in order to provide a constant sampling rate. Thus, letting the frequency of feedback u/NRM be the constant $1/T_s$, where T_s is the sampling period, it should be $\mathrm{NRM}(t) = \mathrm{u}(t)T_s$.

A. Discrete Time Control Equation

To discretize the continuos control equation (7), we simply invoke the Shannon sampling theorem and the rule of thumb [27], which requires that, in order to have a "continuos-like" performance of the system under digital control, the ratio of the time constant $\tau = 1/k$ of the system over the sampling time T_s must belong to the interval (2,4), that is,

$$T_s \in \left[\frac{\tau}{4}, \frac{\tau}{2}\right].$$

However, the mechanism of conveying feedback via RM cells cannot guarantee a constant sampling rate. To overcome this problem we propose to *estimate* the feedback in case it is not available after the sampling time has elapsed. Thus the algorithm can compute the input rate at least every T_s regardless of whether the source gets the feedback information. We define a *worst case estimate* in order to guarantee no cell loss even if the RM cells are not received at the required frequency. Two cases need to be considered:

(1) If the source receives at $t = t_h$ the RM cell, which conveys the value $(r^o(t_h - T_{fb}) - x(t_h - T_{fb}))$, then the input rate is computed as

$$u(t_h) = k\left(r^o(t_h - T_{fb}) - x(t_h - T_{fb}) - \sum_{i=1}^{m} u(t_{h-i}) \cdot \Delta_i\right), \tag{14}$$

where $\Delta_i \leq T_s$ and $t_{h-m} = t_h - \mathrm{RTT}$.

(2) If the time T_s since the last RM cell was received at time t_{h-1} expires and no RM cell is yet received, then the source performs a *worst case estimate* of the feedback information at $t_h = t_{h-1} + T_s$. To be conservative and to prevent cell loss we propose the following worst case estimate of the missing feedback. We assume that in the time interval $[t_{h-1}, t_{h-1} + T_s]$ the queue has zero output rate. Thus, the worst case estimate of the bottleneck-free space is the last received or the last estimate $(r^o(t_{h-1} - T_{fb}) - x(t_{h-1} - T_{fb}))$ minus what the source has pumped into the network during the interval $[t_{h-1} - \mathrm{RTT}, t_{h-1} - \mathrm{RTT} + T_s]$. Without loss of generality and for sake of simplicity we assume that in the interval $[t_{h-1} - \mathrm{RTT}, t_{h-1} - \mathrm{RTT} + T_s)$ the rate is constant and equal to $u(t_{h-1} - \mathrm{RTT})$. Thus the estimate is

$$r^o(t_h - T_{fb}) - x(t_h - T_{fb}) = r^o(t_{h-1} - T_{fb}) - x(t_{h-1} - T_{fb}) - u(t_{h-1} - \mathrm{RTT}) \cdot T_s,$$

and the rate is computed as

$$\begin{aligned} u(t_h) &= k\left(r^o(t_{h-1} - T_{fb}) - x(t_{h-1} - T_{fb}) - u(t_{h-1} - \mathrm{RTT}) \cdot T_s - \sum_{i=1}^{m} u(t_{h-i}) \cdot \Delta_i\right) \\ &= u(t_{h-1}) - k \cdot u(t_{h-1}) \cdot T_s. \end{aligned} \tag{15}$$

Now, if the RM cell is not received in the interval $[t_{h-m}, t_h]$, then the calculated rate is

$$u(t_h) = u(t_{h-1}) \cdot (1 - k \cdot T_s) = u(t_{h-m}) \cdot (1 - k \cdot T_s)^m. \tag{16}$$

Noting that $k \cdot T_s \in \left[\frac{1}{4}, \frac{1}{2}\right]$, it results that $(1 - k \cdot T_s) < 1$. Therefore in case of missing feedback, Eq. (15) implements a *multiplicative decrease algorithm.* When a new feedback is received the rate will increase because the actual $(r^o(t_h - T_{fb}) - x(t_h - T_{fb}))$ cannot be smaller than the worst case estimate.

Remark 3. Note that the behavior of the multiplicative decrease equation (15) is similar to the multiplicative decrease algorithm of the TCP control window when a loss is detected [2]. Thus, it can be concluded that the multiplicative decrease behavior should be a general feature of any closed-loop congestion control algorithm applied to communication networks because, in these systems, the delivery of the feedback at a constant rate cannot be guaranteed.

VII. PERFORMANCE EVALUATION OF THE CONTROL LAW

In this section, computer simulations are carried out to evaluate the performance of the control equation (7) to control the input rate of an ABR connection that shares the bottleneck link with competing (ABR + VBR + CBR) traffic. Moreover, a comparison with the ERICA is reported.

A. Computer Simulation Results

We report the simulation results regarding a VC controlled by Eq. (7). The VC is characterized by a bandwidth-delay product of 10,000 cells and shares

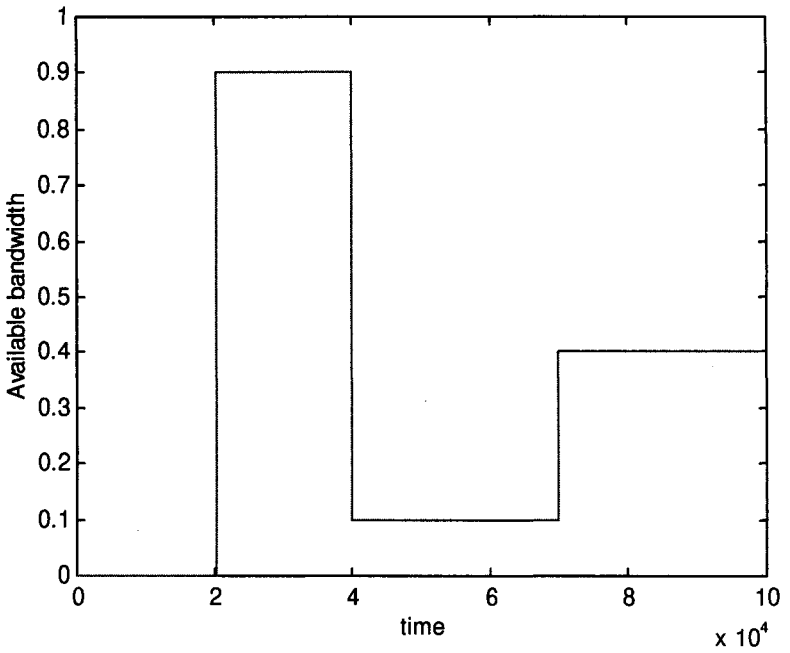


FIGURE 11 Per-VC time-varying available bandwidth $d(t)$.

the bottleneck with (ABR + VBR + CBR) traffic. Figure 11 shows the time-varying best-effort bandwidth, which is available for the considered VC. The bottleneck link capacity is normalized to unity, so that RTT = 10,000 time slots. This can correspond to a cross-continental connection (≈24,000 km round trip) through a wide area network (WAN) with a typical bottleneck link capacity of 155 Mb/s. The constant gain is set to $k = 1/750$, and according to (13), the allocated buffer capacity is set to $r^o = 1 \cdot (10000 + 750)$. The sampling time is $T_s = (2/5)\tau = 300$. Figure 12 shows that the queue length is upper bounded by 9,000 cells $< r^o$ (i.e., cell loss is prevented) and always greater than 0, i.e., 100% link utilization is guaranteed. Figure 13 shows the time waveform of the ABR source input rate.

B. A Comparison with the ERICA

For a comparison with an explicit rate algorithm, we use the well-known ERICA [15]. Explicit rate algorithms require that switches send an explicit rate indication to the source. Basically the ERICA switch measures the ABR available bandwidth ABR_{BW}

$$\text{ABR}_{\text{BW}} = \text{Target Utilization } (U) \times (\text{Link capacity} - (\text{VBR} + \text{CBR})_{\text{BW}}),$$

where the target utilization (U) is a parameter set to a fraction of the available capacity. Typical values of U are 0.9 and 0.95. The fair share of each VC is computed as

$$\text{Fair Share} = \frac{\text{ABR}_{\text{BW}}}{N_{\text{VC}}},$$

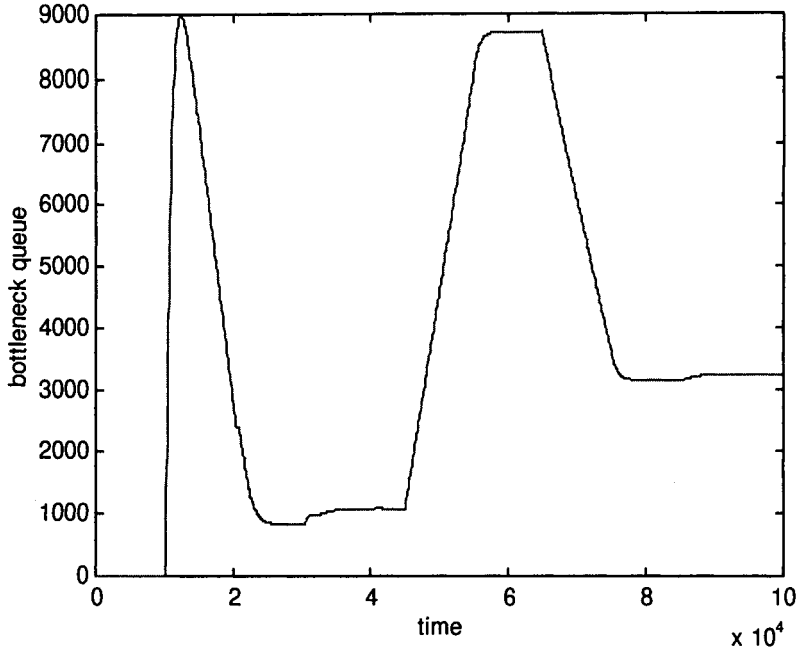


FIGURE 12 Bottleneck queue length.

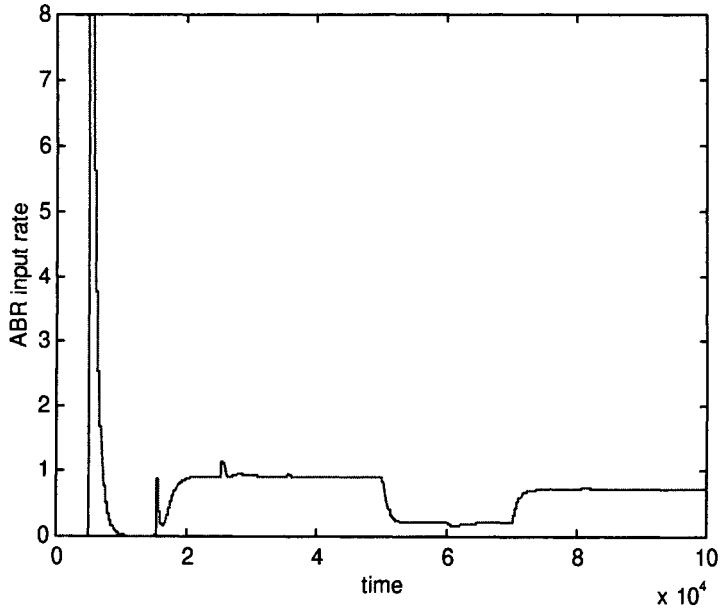


FIGURE 13 Input rate of the ABR source.

where N_{VC} is the number of active connections. The weakness of this algorithm is the computation of fair share. In fact, (1) it is difficult to measure the available bandwidth, which is bursty; and (2) a switch calculates a Fair Share that can be effectively used only if no connection is bottlenecked downstream [14,15]. To overcome point (2) the load factor is defined as

$$z = \frac{\text{ABR}_{\text{input rate}}}{\text{ABR}_{\text{BW}}},$$

and a VC share is computed as

$$\text{VC}_{\text{share}} = \frac{\text{CCR}}{z},$$

where CCR is the source current cell rate stored in the RM cells. The load factor z is an indicator of the congestion level of the link. The optimal operating point is $z = 1$. The goal of the switch is to maintain the network at unit overload [15]. Thus the explicit rate (ER) is computed as [15]

$$\begin{aligned}
\text{ER}_{\text{calculated}} &\leftarrow \text{Max}(\text{FairShare}, \text{VCShare}) \\
\text{ER}_{\text{calculated}} &\leftarrow \text{Min}(\text{ER}_{\text{calculated}}, \text{ABR}_{\text{BW}}) \\
\text{ER}_{\text{calculated}} &\leftarrow \text{Min}(\text{ER}_{\text{calculated}}, \text{ER}_{\text{in RM cell}}).
\end{aligned}$$

ERICA depends upon the measurements of ABR_{BW} and N_{VC}. If there are errors in these measurements, ERICA may diverge; i.e., queues may become unbounded and the capacity allocated to drain queues becomes insufficient. ERICA+ is an enhancement obtained by using queue length as a secondary metric [15]. Other drawbacks of ERICA are: (1) an oscillatory behavior due to the decrease/increase mechanism introduced by the load factor; (2) the selection of the target utilization (U) and of the switch measurement interval; and (3) there are cases in which the algorithm does not converge to max–min fair allocation [15].

Here we show that, even if we assume, despite points (1) and (2), that a switch is able to perform a precise calculation of the bottleneck explicit rate, the lowest upper bound for the queue length is still equal to the the *bandwidth×delay product*. Thus, the *complexity of bandwidth measurement with calculation of bottleneck explicit rate is not justified in comparison with the simplicity of feeding back the free buffer space*. To show this result, consider Fig. 14, which contains a block diagram depicting the basic dynamics of the ERICA. It is assumed that the switch, which is the bottleneck for the considered VC, is able to measure and feedback the per-VC available bandwidth filtered

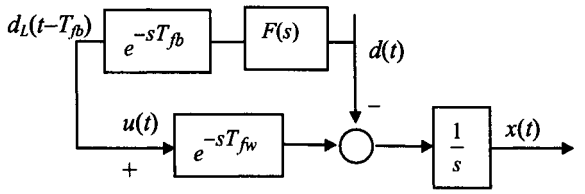


FIGURE 14 Block diagram of the dynamics of the ERICA.

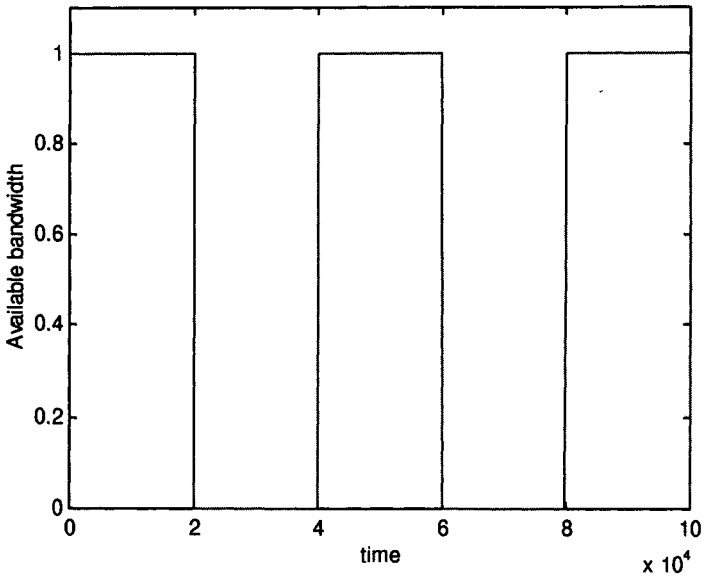


FIGURE 15 Per-VC ABR bandwidth.

by a low-pass filter $F(s)$.[6] This assumption is made for two reasons: (1) congestion occurs whenever the low-frequency input traffic rate exceeds the link capacity [19,28]; and (2) since the controlled system is sampled via RM cells, it is not possible to reconstruct the higher-frequency component of $d(t)$ due to the Shannon sampling theorem. Suppose that the available bandwidth $d(t)$ changes from 0 to a at $t = t_0$. Looking at the block diagram shown in Fig. 14, it is easy to see that, after the time $4\tau + T_{fw} + T_{fb}$, the VC input rate at the input of the queue is a; that, is all available bandwidth is used. Now suppose that the available bandwidth suddenly drops to 0. The queue input rate remains equal to a for a time equal to $4\tau + \text{RTT}$, then it drops to 0. Thus, during this time the queue level reaches the value $a \cdot (4\tau + \text{RTT})$. For instance, assuming the on–off available bandwidth shown in Fig. 15, with RTT = 1000, simulation results give the queue behavior reported in Fig. 16. In general, it can be noted that an on–off available bandwidth with amplitude a and period equal to 2*RTT causes a maximum queue length equal to $a \cdot (4\tau + \text{RTT})$.

VIII. CONCLUSIONS

Classical control theory is proposed for modeling the dynamics of high-speed ATM communication networks. Smith's principle is exploited to design a congestion control law that guarantees no data loss and full link utilization over communication paths with any bandwidth–delay product. The properties of the proposed control law are demonstrated in a general setting via mathematical analysis. In particular, it is shown that full link utilization and no cell loss

[6]For instance, $F(s)$ can be the first-order low-pass filter $1/(1 + \tau s)$.

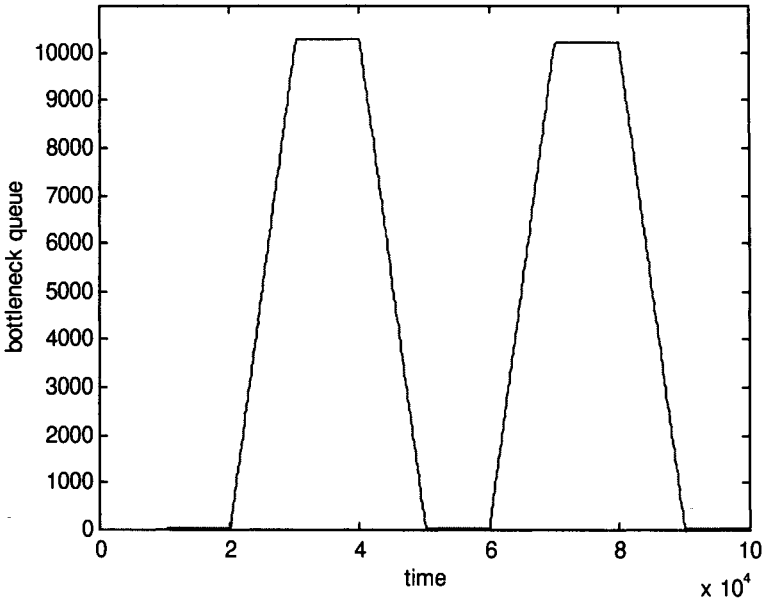


FIGURE 16 Bottleneck queue length of the ERICA.

are guaranteed during both transient and steady-state conditions. The advantages of the proposed algorithm over a heuristic algorithm such as the ERICA algorithm are discussed. Finally, computer simulations confirm the validity of the theoretical results.

REFERENCES

1. Varaiya, P., and Walrand, J. *High-Performance Communication Networks*. Morgan Kaufmann, San Francisco, CA, 1996.
2. Jacobson, V. *ACM Comput. Commun. Rev.* 18(4): 314–329, 1988.
3. Jain, R. *Comput. Networks ISDN Systems* 28(13): 1723–1738, 1996.
4. Peterson, L. L., and Davie, B. S. *Computer Networks*. Morgan Kaufman, San Francisco, CA, 1996.
5. Ding, W. *IEEE Trans. Circuits Systems Video Technol.* 7(2): 266–278, 1997.
6. Le Boudec, J., de Veciana G., and Walrand, L. In *Proc. of IEEE Conf. on Dec. and Control '96*, Kobe, Japan, Vol. 1, 1996, pp. 773–778.
7. Liew, S. C., and Chi-yin Tse, D. *IEEE/ACM Trans. Networking* 6(1): 42–55, 1998.
8. ATM Forum Technical Committee TMWG, "ATM Forum Traffic Management Specification Version 4.0," af-tm-0056.000, 1996. Available at http://www.atmforum.com.
9. Benmohamed, L., and Meerkov, S. M. *IEEE/ACM Trans. Networking* 1(6): 693–708, 1993.
10. Bonomi, F., Mitra, D., and Seery, J. B. *IEEE J. Selected Areas Commun.* 13(7): 1267–1283, 1995.
11. Fendick, K. W., Rodrigues, M. A., and Weiss, A. *Perform. Eval.* 16(1–3): 67–84, 1992.
12. Ramakrishnan, K., and Jain, R. *ACM Trans. Comput. Systems.* 8(2): 158–181, 1990.
13. Yin, N., and Hluchyj, M. G. In *Proc. IEEE Infocom 94*, vol. 1, pp. 99–108, 1994.
14. Charny, A., Clark, D. D., and Jain, R. In *Proc. of IEEE ICC95*, vol. 3, pp. 1954–1963, 1995.
15. Jain, R., Kalyanaraman, S., Goyal, R., Fahmy, S., and Viswanathan, R. "ERICA Switch Algorithm: A Complete Description," ATM Forum, af-tm 96-1172, August 1996. Available at http://www.cis.ohio-state.edu/jain/.

16. Kolarov, A., and Ramamurthy, G. In *Proc. of IEEE Infocom 97*, vol. 1, pp. 293–301, 1997.
17. Izmailov, R. *IEEE Trans. Automat. Control* 40(8): 1469–1471, 1995.
18. Izmailov, R. *SIAM J. Control Optim.* 34(5): 1767–1780, 1996.
19. Zhao, Y., Li, S. Q., and Sigarto, S. In *Proc. of IEEE Infocom 97*, vol. 1, pp. 283–292, 1997.
20. Pan, Z., Altman, E., and Basar, T. In *Proc. of the 35th IEEE Conf. on Dec. and Control*, vol. 2, pp. 1341–1346, 1996.
21. Mascolo, S. In *Proc. IEEE Conf. on Dec. and Control '97*, vol. 5, pp. 4595–4600, 1997.
22. Benmohamed, L., and Wang, Y. T. In *Proc. of IEEE Infocom 98*, vol. 1, pp. 183–191, 1998.
23. Suter, B., Lakshman, T. V., Stiliadis D., and Choudhury A. K. *IEEE Proc. Infocom 98*, San Francisco, Mar. 1998.
24. Smith, O. *ISA J.* 6(2): 28–33, 1959.
25. Marshall, J. E. *Control of Time-Delay Systems*. Peter Peregrinus, London, 1979.
26. Franklin, G. F., Powell, J. D., and Emami-Naeini, A. *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, MA, 1994.
27. Astrom, K. J., and Wittenmark, B. *Computer-Controlled Systems*. Prentice–Hall, Englewood Cliffs, NJ, 1984.
28. Li, S. Q., and Hwang, C. *IEEE/ACM Trans. Networking* 3(1): 10–15, 1995.

This Page Intentionally Left Blank

20

OPTIMIZATION TECHNIQUES IN CONNECTIONLESS (WIRELESS) DATA SYSTEMS ON ATM-BASED ISDN NETWORKS AND THEIR APPLICATIONS

RONG-HONG JAN

Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

I-FEI TSAI

Wistron Corporation, Taipei 221, Taiwan

Database and Data Communication Network Systems, Vol. 3

I. INTRODUCTION

To provide the interconnection between LANs/MANs is the most important applications in the ATM-based B-ISDN. ATM networks are inherently connection-oriented (CO); however, conectionless (CL) service is the predominant mode of communications in legacy LANs/MANs. This implies that the CL data will make up a dominant portion of ATM network's traffic load in the long term. To support the CL data service on a B-ISDN, two alternatives, *direct* and *indirect* approaches, for the ATM networks have been specified by ITU-T [1,2].

For the *direct* approach, CL service is provided directly in the B-ISDN. Connectionless data traffics between the B-ISDN customers are transferred via *connectionless servers* (CLSs) in which *connectionless service functions* (CLSFs) are provided. Connectionless protocol functions such as addressing, routing, and quality of service (QoS) selection are handled by CLSs. To serve as a ATM cell router, the CLS routes cells to their destination or intermediate CLS according to the routing information included in the user data. Although logically a separate entity, the CLSF should be physically located in the same block as the ATM switch or may even be a physical part of the switch itself [19,36].

For the *indirect* approach, CL data traffics between customers are transparently transferred above the ATM layer via the connection-oriented service. However, the use of fully meshed semipermanent connections may not be feasible due to the limit of the number of virtual path identifiers (VPIs) that can be supported by a switch. On the other hand, the use of switched connections will introduce call setup delay and overhead for call control functions within the network. Besides, the indirect approach cannot fully utilize the benefits from multiple path routing of CL data traffic that is inherently free from being out of sequence. Thus, from the point of scalability and efficiency, *direct* CL services in a large-scale network realized by means of the CLSF seem to be a more reasonable approach. There is growing consensus that providing CL service for the interconnection of LANs and MANs in a public ATM environment will more likely entail the use of CLSs [20].

For the sake of load balancing and congestion avoidance, a large-scale B-ISDN necessitates more than one CLS. In general, the fewer the number of CLSs, the higher the load on each VP, and the higher the total transport cost in the virtual overlay network. On the other hand, it is not necessary to attach a CLSF at every switching node. The number of switching nodes with a CLSF depends upon the volume of CL traffic to be handled and the cost of the network. So the determination of where to place CLSs and how to interconnect CLSs has important performance consequences. These problems are similar to network optimization problems, such as access facility location, warehouse location, and partitioning. It has been proved [35] that the problem of determining the optimal number of CLSs in an ATM network is as difficult as the problem of determining the minimum size of *vertex cover* in the corresponding graph. Note that the problem of determining the minimum size of vertex cover in a network is known as the vertex cover problem, which is NP-complete [27]. Optimization techniques for obtaining optimal CLS configurations in an ATM network have been presented in [35]. These include the branch and bound method, the greedy method, the heuristic approach, and the simulating annealing method.

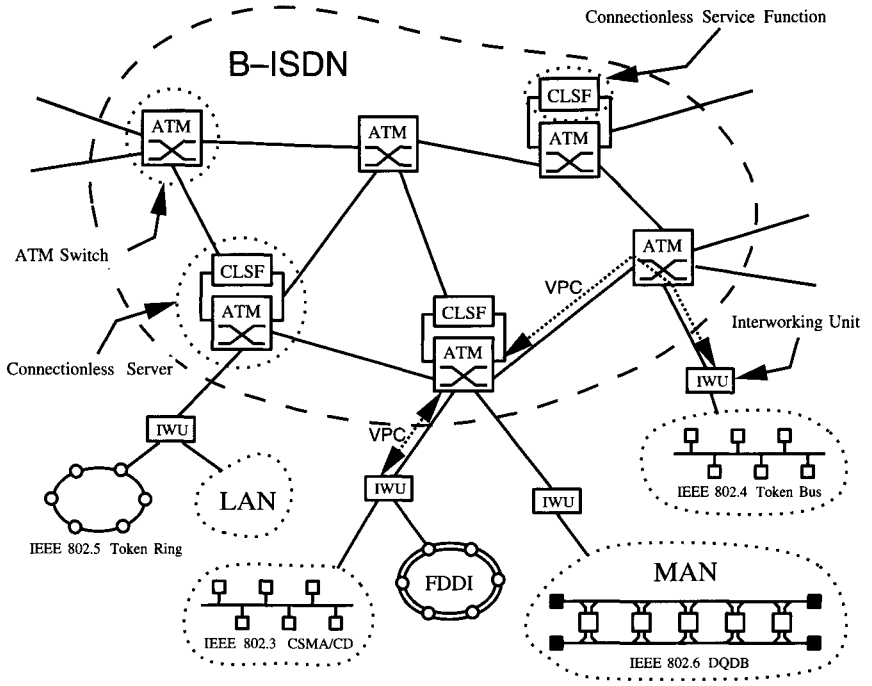


FIGURE 1 Interconnection of legacy LANs/MANs with a wide area public ATM network.

After a determinant number of CLSs are located among the ATM switching nodes, an efficient semipermanent VP layout should be preconfigured around the CLSs and the interworking unit (IWUs) to which LANs/MANs are attached. Despite virtually any embedded topology being possible, it is desirable that there is at least one CLS lying on the virtual path connection for each IWU pair. The IWUs between the CLSs and the LANs/MANs work as an interface of CL and CO environment. CL traffic between LANs/MANs will be transported by way of the VPs between the intermediate CLSs and IWUs (see Fig.1).

As a packet forwarder, CLS can operate in two modes, say the *reassembly mode* and the *on-the-fly mode* [38]. In the reassembly mode, each CLS segments an entire data packet from the upper layer into multiple ATM cells, then forwards them to the same downstream CLS. If any cell of the same packet is lost, then all the other cells will be discarded. In the on-the-fly mode, a CLS routes each cell independently, and the segmentation/reassembly is done only on the source and destination IWUs or ATM hosts. That is, ATM cells are forwarded as a pipeline without regards to their entirety.

Although the former seems more inefficient and incurs more processing delay than the latter, only the reassembly mode server survives under high traffic load. This is because the on-the-fly server forwards each cell blindly regardless of whether the cells belong to broken packets destined to be dropped at the destination end. The functionality of *loss and retransmission* is done only by the upper layer (such as TCP) of both ends, so the *effective throughput* degrades as traffic load (/cell loss rate) becomes high.

To improve the performance of the on-the-fly server, Boiocchi *et al.* [38] proposed a data unit discard algorithm for the buffer management in a CLS.

The main idea is that if the occupation of the shared output buffer exceeds a certain threshold, all the new "head cells" (beginning of message) and their succeeding cells (continuation and end of message) should be dropped. The performance was then evaluated in terms of packet loss rate as a function of traffic intensity and buffer size. With this strategy, the on-the-fly CLS performs consistently better than the reassembly CLS. It is remarkable that this simple buffer management scheme can be introduced not only to CLSs but also to the ordinary ATM switches without a CLSF. In [39], a packet discard strategy named *early packet discard* that has the same idea as that in [38] was investigated to alleviate the effects of fragmentation of TCP traffic over ATM networks. The detailed hardware architecture of the on-the-fly CLS can be found in [36,37], where only [37] employed the early packet discarding as an enhanced switching node.

A number of studies concerned with supporting CL data service on a B-ISDN, such as the general framework of the protocol architectures [6–14], hardware architecture and buffer management [36–38], the CLS location problem [34,35], and bandwidth management and traffic control [15–17], have been proposed. In [35], the authors considered how to locate a certain amount of connectionless servers among the switching nodes in a public ATM network for the internetworking of connectionless data networks. The problem, so-called the p-CLS problem, which is similar to the p-median problem [13], was formulated as a network optimization problem and shown to be NP-complete even when the network has a simple structure. The p-median problem is to minimize the sum of distances from demand sites to their respective nearest supply sites. This kind of network optimization problem occurs when new facilities are to be located on a network.

The objective function of the CLS location problem is to minimize total transport cost. Minimized total transport cost is equivalent to minimized average transport cost. Therefore, the minimum average end-to-end delay or bandwidth requirement can be achieved. Two algorithms, one using the *greedy method* and the other based on *branch-and-bound strategy,* for determining the locations of CLSs were presented. By finding the optimal locations of a CLS, an optimal virtual overlay network that has minimized *total transport costs* for the CL data traffics can be constructed.

Most of the optimization techniques available for this kind of location problem rely on integer programming for a final solution. The emphasis of [35] is to develop two efficient algorithmic approaches for the optimal location of CLSs in the ATM networks and to determine the optimal virtual topology.

The optimal virtual path connections between all the gateway pairs can be determined as long as the optimal location of CLS sites have been determined. Thus the CL traffic routing is also decided. A conceptual *virtual overlayed network* for the direct provision of CL data service in the ATM networks is shown in Fig. 2. In this figure VPs are preconfigured between the gateways and CLSs, and the traffics from the source gateway are routed to the destination gateway by CLS. It is clear that the proposed solution can be applied to homogeneous ATM LANs by substituting gateways with ATM stations. It can also be employed to design the *broadband virtual private networks* [32].

There exists two other related paper works on the problem of where to place the CLSs and how to connect them [26,34]. However, only in [35] is

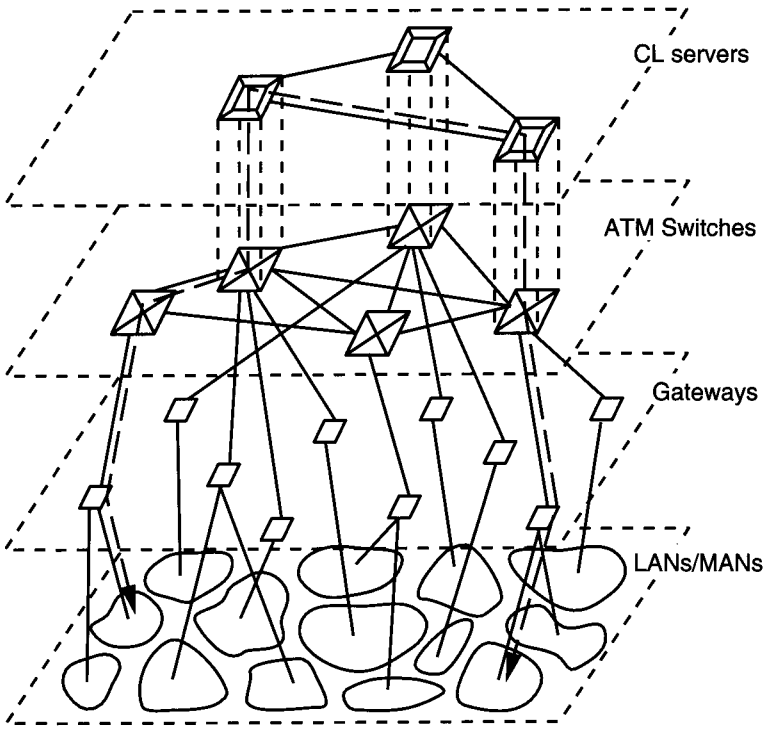


FIGURE 2 A virtual overlayed network.

a definite mathematical formulation given, and an unambiguous prescription can be derived from the proposed procedures. Besides, the numerical results are carefully analyzed. However, the evaluation of statistical multiplexing gain for inter-LAN/MAN CL data traffic was absent.

The organization of this chapter is as follows. In the next section, we first introduce the aspects of connectionless data service on a B-ISDN. Two approaches [1], *indirect* and *direct*, offering CL communications in the ATM network are discussed. After comparing these two approaches, we will briefly introduce the general protocol structure and the CLSF for the *direct* provision of CL service. In Section III, we will describe the CLS location problem formally. In Section IV, a heuristic algorithm based on the greedy method for finding the optimal locations of CLSs is presented. It can be shown that, by using this algorithm, we can get an optimal solution if p is not smaller than a threshold with respect to the number of ATM switches in the network model; otherwise, the solution is near-optimal. Since the value of p (the number of CLSs) may be restricted in the hardware costs and routing complexity (e.g., a network with the capabilities of broadcasting and multicasting may influence the design of virtual topology), we still must consider those cases (a smaller p) where the optimal solution cannot be found by the greedy method. Therefore, a branch-and-bound strategy is also proposed to find the exact solution for this problem. In Section V, three heuristics algorithms for the optimization problem subject to bandwidth constraint are given. In Section VI, according to the location of CLSs, we discuss the efficient VP layout for CL data traffic. Conclusions and discussions are given in the last section.

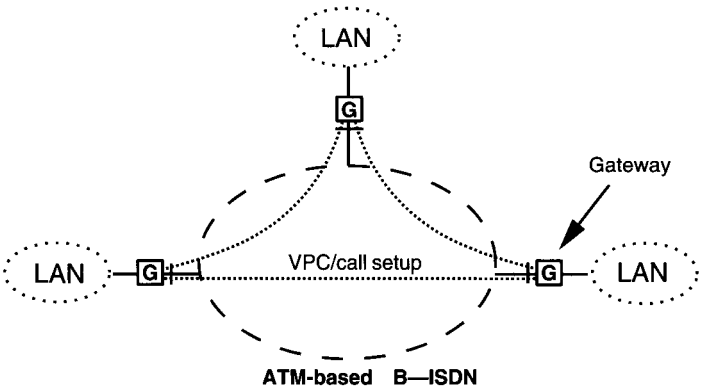


FIGURE 3 Indirect provision of connectionless data service.

II. CONNECTIONLESS DATA SERVICES IN AN ATM-BASED B-ISDN

A connectionless data service supports data transfer between users based on connectionless data transfer techniques. It does not imply connectionless methods implemented directly within a B-ISDN. In the B-ISDN, virtual channels are established at the ATM layer only by means of the connection-oriented technique. Therefore, connectionless data service can be supported in a B-ISDN in one of the following two ways:

(1) *Indirectly via a B-ISDN connection-oriented service.* See Fig. 3; in this case a transparent connection of the ATM layer, either permanent, reserved, or on demand, is used between B-ISDN interfaces. Connectionless protocols operating on and above the adaptation layer are transparent to the B-ISDN. The connectionless service and adaptation layer functions are implemented outside the B-ISDN. The B-ISDN thus imposes no constraints on the connectionless protocols to be adopted.

(2) *Directly via a B-ISDN connectionless service.* See Fig. 4; in this case the connectionless service function would be provided within the B-ISDN. The

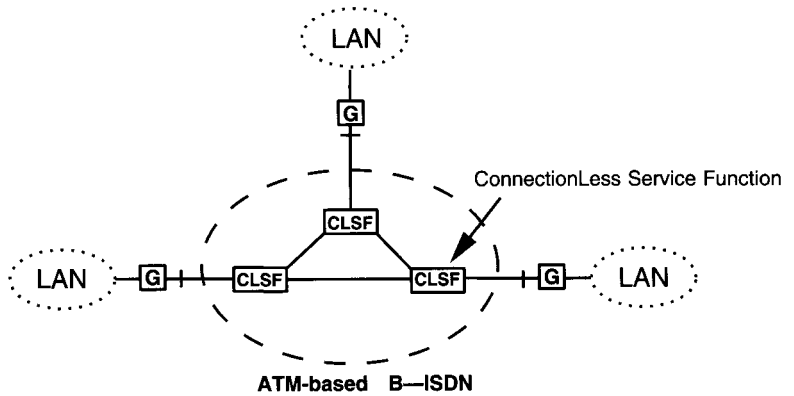


FIGURE 4 Direct provision of connectionless data service.

CLSF handles connectionless protocols and routes data to a destination user according to routing information included in the user data. Thus a connectionless service above the adaptation layer is provided in this case.

Service (1) may lead to an inefficient use of virtual connections of user–network interface and network–node interface, if permanent or reserved connections are configured among users. ATM connections require the allocation of connection identifiers (VPIs and VCIs) at each intermediate ATM switch. This means that some bandwidth amount at each switch will be reserved *implicitly* for these connections. With the availability of signaling capabilities, an end-to-end connection may be established on demand at the commencement of connectionless data service. This on-demand operation of Service (1) may cause a call setup delay, and may introduce a load on call control functions within the network.

For Service (2), there are also two options, depending on the availability of B-ISDN signaling capabilities. Option one is to use preconfigured or semipermanent virtual connections between users and connectionless service functions to route and switch connectionless data across the network. Option two is to establish virtual connections at the commencement of the connectionless service session.

Support of Service (1) will always be possible. The support of a direct B-ISDN connectionless service (Service (2)) and the detailed service aspects are in [2].

The provision of the connectionless data service in B-ISDN can be realized by means of ATM switched capabilities and CLSFs. The ATM switched capabilities support the transport of connectionless data units in a B-ISDN between specific CLSF functional groups to realize the adaptation of the connectionless data units into ATM cells to be transferred in a connection-oriented environment. The CLSF functional groups may be located outside the B-ISDN, in a private connectionless network or in a specialized service provider, or inside the B-ISDN. The relevant reference configuration for the provision of the connectionless data service in the B-ISDN is depicted in Fig. 5.

The ATM switched capabilities are performed by the ATM nodes (ATM switch/cross-connect), which realize the ATM transport network. The CLSF functional group terminates the B-ISDN connectionless protocol and includes functions for the adaptation of the connectionless protocol to the intrinsically connection-oriented ATM layer protocol. These latter functions are those performed by the ATM adaptation layer type 3/4 (AAL 3/4), while the former ones are those related to the layer above AAL 3/4, denoted CLNAP (*c*onnection*l*ess *n*etwork *a*ccess *p*rotocol). The general protocol structure for the provision of CL data service in a B-ISDN is shown in Fig. 6.

The CL protocol includes functions such as routing, addressing, and QOS selection. In order to perform the routing of CL data units, the CLSF must interact with the control/management planes of the underlying ATM network.

The CLSF functional group can be considered implemented in the same equipment together with the ATM switched capabilities as depicted in Fig. 7 (option A). In this case it is not necessary to define the interface at the **P** reference

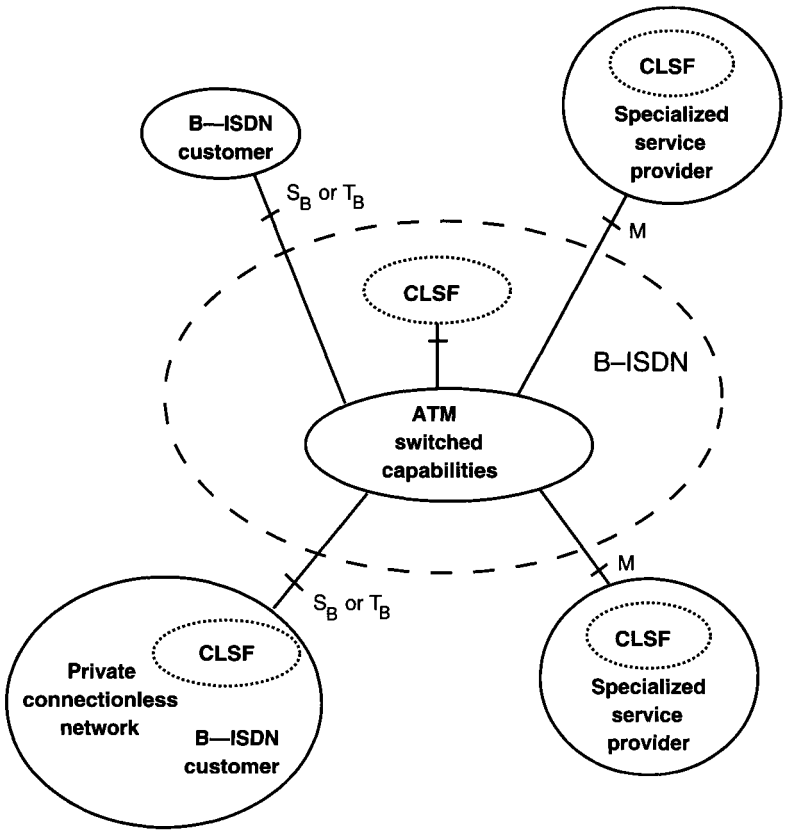


FIGURE 5 Preference configuration for the provision of the CL data service in a B-ISDN.

point. CLSF functional group and ATM switched capabilities can be implemented also in separate equipment (option B). In this case interfaces shall be defined at the M or P reference points depending on whether the CLSF is located outside or inside the B-ISDN.

From the above discussion, the *indirect* provision of CL service is only applicable in the case where a few users of CL services are attached to the

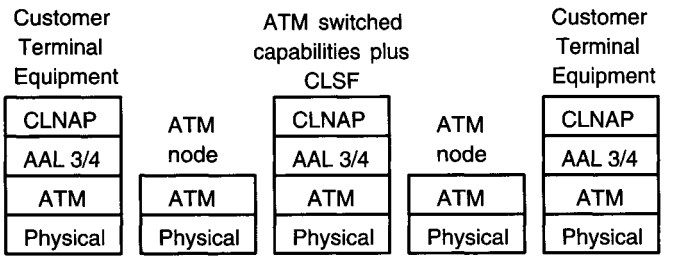


FIGURE 6 General protocol structure for provision of CL data service in a B-ISDN.

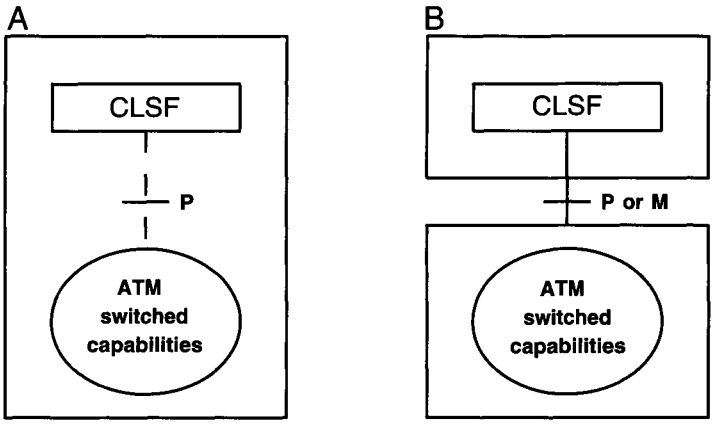


FIGURE 7 Two options for the CLSF functional group and ATM switched capabilities, implemented in (A) the same and (B) different equipment.

B-ISDN. Maintaining a full mesh between all the gateway pairs is impractical for a large number of gateways. So the *direct* CL service realized by means of the CLSF seems to be a more reasonable approach. CL store-and-forward nodes (CLSs) make it possible to implement arbitrary topologies (not necessarily fully meshed), thus reducing the number of VPCs required. Its drawback, of course, is the processing overhead at each CLS site. In the next section, based on the *direct* approach, the problem of internetting LANs/MANs with a wide area public ATM network is formulated as an optimization problem.

III. CONNECTIONLESS DATA SYSTEM OPTIMIZATION

In this section, based on the *direct* approach, we focus on the topological optimization of internetting LANs/MANs with a wide area public ATM network.

A. System Models

Consider the interconnection of LANs/MANs with a wide area public ATM networks as shown in Fig. 1. We can model this system into a graph model. That is, let $G = (V_g \cup V_s, E_{gs} \cup E_{ss})$ be an ATM network, where vertex sets V_g is the set of gateways (IWUs), V_s is the set of ATM switches, link sets E_{gs} is the set of links between a gateway and an ATM switch, and E_{ss} is the set of links between two ATM switches. Let vertex v_{ijk} represent a LAN or MAN as shown in Fig. 8.

Note that, in our model, LANs/MANs v_{ijk} are attached to gateways g_{ij} and gateways g_{ij} connected to switches s_i. In addition, we assume that all LANs/MANs and gateways are single-homed. That is, every v_{ijk} is only adjacent to a gateway and every g_{ij} is only connected to an ATM switch s_i.

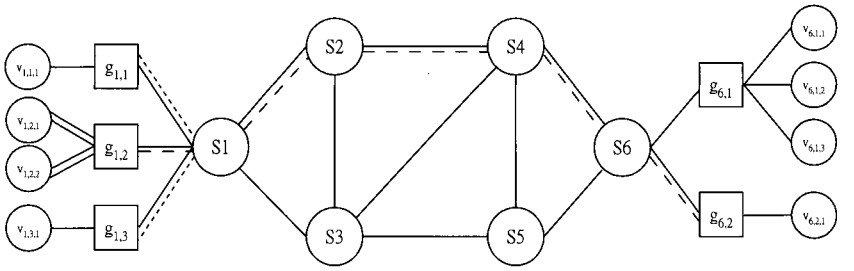


FIGURE 8 A sample network model. (——) Class 1, local traffic; (- - - - -) Class 2, CL traffic between two gateways attached to the same switch; and (– – –) Class 3, CL traffic between two gateways attached to different switches.

The connectionless traffics can be divided into three types as follows.

Type 1 is the traffic from LAN/MAN v_{ijh} to $v_{ijk}(h \neq k)$ via gateway g_{ij}. This traffic is the local traffic and can be handled by gateway g_{ij}.

Type 2 is the traffic between two gateways attached to the same switch, e.g., the traffic from g_{ih} to $g_{ij}(h \neq j)$ via switch s_i. Permanent or semi-permanent VPs should be configured in advance for this type of traffic.

Type 3 is the traffic between two gateways connected to different switches, e.g., the traffic from g_{ih} to $g_{jk}(i \neq j)$. We are only concerned with type 3 traffic in this chapter.

For ATM networks, the CL service cost may include (1) the transmission cost needed to connect each pair of gateways, (2) the buffer cost required in gateways, (3) the cost of CLSs, and (4) the buffer cost required in CLSs [34].

Consider the transmission cost for gateway pair (g_{ij}, g_{hk}) in ATM network G. Let matrix $T = [w(g_{ij}, g_{hk})]_{u \times u}$ represent the traffic load matrix, where $w(g_{ij}, g_{hk})$ is the traffic flow between two gateways g_{ij} and g_{hk} and $|V_g| = u$. Let $C = [c(i, j)]_{v \times v}$ be the cost matrix, where $c(i, j)$ represents the cost per unit flow of the link (i, j) and $|V_g \cup V_s| = v$. The link costs can be interpreted as matrices other than the distances. The matrix might be the number of hops, time delay in milliseconds, cell loss rate, penalties, or something similar. Let $P_{(i,j)} : i = v_0, v_1, \ldots, v_k = j$ denote a path from i to j. The cost of path $P_{(i,j)} : i = v_0, v_1, \ldots, v_k = j$, denoted as $d(i, j)$, is the sum of the link costs in path $P_{(i,j)}$. That is,

$$d(i, j) = \sum_{n=0}^{k-1} c(v_n, v_{n+1}).$$

Thus, the transmission cost for gateway pair (g_{ij}, g_{hk}) is equal to the traffic flow $w(g_{ij}, g_{hk})$ multiplied by the path cost $d(g_{ij}, g_{hk})$. It can be written as $C_{cl}(g_{ij}, g_{hk}) = w(g_{ij}, g_{hk}) \times d(g_{ij}, g_{hk})$. Note that the CL data flows between all gateway pairs are transported via the CLSs. That is, for each gateway pair, CL data flows must pass through at least one intermediate CLS. Because the extra delay being introduced into each intermediate CLS, in general, CL data flows

are handled by exactly one CLS. Therefore the transmission cost for the CL traffic between gateway pair (g_{ij}, g_{hk}) can be defined as

$$C_{cl}(g_{ij}, g_{hk}) = w(g_{ij}, g_{hk}) \times \{d(g_{ij}, s) + d(s, g_{hk})\},$$

where switch node s is a CLS.

The cost of CLS can be defined as the fixed cost to install CLS and the processing cost of CL traffic that passes CLS. Thus, the cost of CLS s_i can be written as

$$C_s(s_i) = f + \alpha \sum_{j=N(s_i)} a(j, s_i),$$

where f is the fixed cost to install CLS s_i, α is the processing cost per unit flow, $N(s_i)$ is the set of all the neighboring nodes of CLS s_i, and $a(j, s_i)$ is the data flow of link (j, s_i).

The buffer cost for CLS s_i is assumed to be proportional to the buffer size (memory size) of server s_i. Thus, the buffer cost for s_i can be written as

$$C_b(s_i) = \beta m_{s_i},$$

where β is the memory cost per unit size (e.g., 1 Mbits) and m_{s_i} is the buffer size of server s_i. Similarly, we defined the buffer cost for gateway g_j as

$$C_b(g_j) = \gamma m_{g_j},$$

where γ is the memory cost per unit size (e.g., 1 Mbits) and m_{g_j} is the buffer size of server g_j. Therefore, the total cost of the CL service is

$$TC = \sum_{\forall g_{ij}, g_{hk} \in V_g} C_{cl}(g_{ij}, g_{hk}) + \sum_{\forall s_i \in M} (C_s(s_i) + C_b(s_i)) + \sum_{\forall g_j \in V_g} C_b(g_j), \quad (1)$$

where M is the set of CLSs in the ATM network.

B. Statement of the Various CLSs Allocation Problems

In this section three CLSs allocation problems are introduced. The first problem of CLSs allocation is to optimize the virtual topology embedded in network G. The objective is to locate p CLSs to $|V_s|$ ATM switches such that the total CL service cost is minimized [35]. In this problem, we assume that the number of CLSs is given and the total buffer cost for gateways is a constant. Thus, the objective function for the optimal locating of CLSs M is to minimize

$$f(M) = \sum_{\forall g_{ij}, g_{hk} \in V_g} C_{cl}(g_{ij}, g_{hk}). \quad (2)$$

Note that the minimum virtual path connection from g_{ij} to g_{hk} via CLS s is

$$\min_{\forall s \in M}\{d(g_{ij}, s) + d(s, g_{hk})\}.$$

The function (2) can be rewritten as

$$\begin{aligned} f(M) &= \sum_{\forall g_{ij}, g_{hk} \in V_g} w(g_{ij}, g_{hk}) \times \min_{\forall s \in M}\{d(g_{ij}, s) + d(s, g_{hk})\} \\ &= \sum_{\forall g_{ij}, g_{hk} \in V_g} \min_{\forall s \in M} w(g_{ij}, g_{hk}) \times \{d(g_{ij}, s) + d(s, g_{hk})\}. \end{aligned} \tag{3}$$

Our goal is to choose a set of M from V_s such that (3) is minimized. This can be stated as:

PROBLEM 1.

$$\min_{M \subseteq V_s} f(M) = \min_{M \subseteq V_s} \sum_{\forall g_{ij}, g_{hk} \in V_g} \min_{\forall s \in M} w(g_{ij}, g_{hk}) \times \{d(g_{ij}, s) + d(s, g_{hk})\}. \tag{4}$$

Problem 1 is a unconstrainted optimization problem and only considers transmission cost as its objective function. However, the servers setup cost and CL service processing cost are also important in CLS allocation problems. Therefore, the second optimization problem, denoted as Problem 2, includes these cost items. This problem is a simple version of the problem proposed by Eom *et al.* [34]. Problem 2 is to find an optimal number of CLSs and their locations such that total cost is minimized.

Actually, the formulation of Problem 1 can be shown to be NP-complete by restricting its instance to the *vertex cover* problem. The optimization problem in Eq. (4) can be converted into the following decision problem, the p-CLS **problem:**

Instance. Graph $G'(V_s, E_{ss})$, for each node pair (s_i, s_j) of G' weight $w(s_i, s_j) \in Z^+$, for each edge $\{s_i, s_j\} \in E_{ss}$, length $c(s_i, s_j) \in Z^+$, positive integer $p \leq |V_s|$, and positive integer B.

Question. Let $d(s_i, s_j)$ be the length of the shortest path from s_i to s_j. Is there a node set of size p or less for G', i.e., a subset $M \subseteq V_s$ with $|M| \leq p$, such that

$$\mathrm{TTC}_M = \sum_{\substack{\forall s_a, s_c \in V_s \\ a \neq c}} \min_{\substack{m_i \in M \\ 1 \leq i \leq p}} w(s_a, s_c)\{d(s_a, m_i) + d(m_i, s_c)\} \leq B? \tag{5}$$

We can restrict the p-CLS problem to a *vertex cover* problem by allowing only instances having $c(s_i, s_j) = 1$ for all $\{s_i, s_j\} \in E_{ss}$ and having B being equal to the lower bound of our objective function in Eq. (4), say LB_{ss}. The NP-completeness of the p-CLS problem can be derived from the following lemmas.

LEMMA 1. *A subset M^* of V_s, with cardinality p, is a feasible solution of the p-CLS problem for $G'(V_s, E_{ss})$ if and only if for each node pair of*

G' there always exists at least one node of M^* lying on one of its shortest paths.

Proof. As we have described at Section II, the lower bound of Eq. (4) can be reached only if, in the ideal case, for each gateway pair there always exists at least one CLS on one of its shortest paths. In other words, for each node pair of G' there always exists at least one node of M^* lying on one of its shortest paths if and only if the total transport cost for M^*, say TTC_{M^*}, is equal to LB_{ss}. Since LB_{ss} is the lower bound of the value of optimal solution, i.e., $\mathrm{TTC}_M \geq \mathrm{LB}_{ss}$ for all subsets M of V_s, therefore, M^* is of size p for which total transport cost TTC_{M^*} is equal to LB_{ss} if and only if M^* is an optimal solution of p-CLS of G'. ■

LEMMA 2. *For each node pair of G' there always exists at least one node of M^* lying on one of its shortest paths if and only if for each edge $\{s_i, s_j\} \in E_{ss}$ at least one of s_i and s_j belongs to M^*.*

Proof. $\Rightarrow$] For those adjacent node pairs (s_i, s_j), there always exists at least one node of M^* lying on its only shortest path $< s_i, s_j >$, indicating that $s_i \in M^*$ or $s_j \in M^*$.

$\Leftarrow$] For each node pair, its shortest path always contains no less than one edge. Thus, if for each edge $\{s_i, s_j\} \in E_{ss}$ at least one of s_i and s_j belongs to M^*, then for each node pair there always exists at least one node of M^* lying on one of its shortest paths. ■

THEOREM 1. *The p-CLS problem is NP-complete even if the network is a planar graph of maximum vertex degree 3, all of whose links are of cost 1 and all of whose node pairs have weight 1.*

Proof. We shall argue first that $p\text{-CLS} \in NP$. Then we shall show that p-CLS is NP-hard.

First, to show that p-CLS belongs to NP, suppose we are given a graph $G' = (V_s, E_{ss})$, and integers p and B. The certificate M we choose is the subset of V_s. The verification algorithms affirm that $|M| \leq p$, and then it checks whether the inequation (6) is satisfied. This verification can be performed straightforwardly in polynomial time.

Garey and Johnson have shown that the *vertex cover* in planar graphs with maximum degree 3 is NP-complete [27]. Let $G'(V_s, E_{ss})$ be a planar graph of maximum degree 3, edge length $c(s_i, s_j) = 1$ for all $\{s_i, s_j\} \in E_{ss}$, and weight $w(s_a, s_c) = 1$ for all node pairs (s_a, s_c) of G'. According to Lemma 1 and Lemma 2, we can conclude that M^* is a vertex cover of size p for G. Therefore, the vertex cover problem is a special case of the p-CLS problem, thus completing the proof. ■

PROBLEM 2.

$$\min_{M \subseteq V_s} \left\{ \sum_{\forall g_{ij}, g_{hk} \in V_g} C_{\mathrm{cl}}(g_{ij}, g_{hk}) + \sum_{\forall s_i \in M} (C_s(s_i)) \right\}$$

$$= \min_{M \subseteq V_s} \left\{ \sum_{\forall g_{ij}, g_{hk} \in V_g} \min_{\forall s \in M} \{w(g_{ij}, g_{hk}) \times \{d(g_{ij}, s) + d(s, g_{hk})\}\} + \sum_{\forall s_i \in M} \left(f + \alpha \sum_{j=N(s_i)} a(j, s_i) \right) \right\},$$

where f is the fixed cost to install server s_i, α is the processing cost per unit flow, $N(s_i)$ is the set of all the neighboring nodes of server s_i, and $a(j, s_i)$ is the data flow of link (j, s_i).

Problem 3 is proposed by Eom *et al.* [34]. Their problem is to find an optimal number of CLSs and their locations subject to maximum delay constraints and packet loss probability constraints. Given the ATM network topology, the number of gateways and their locations, and the traffic between each pair of gateways $T = [w(g_{ij}, g_{hk})]_{u \times u}$, the problem can be stated as:

PROBLEM 3.

$$\min_{M \subseteq V_s} \left\{ \sum_{\forall g_{ij}, g_{hk} \in V_g} C_{cl}(g_{ij}, g_{hk}) + \sum_{\forall s_i \in M} (C_s(s_i) + C_b(s_i)) + \sum_{\forall g_j \in V_g} C_b(g_j) \right\} \quad (6)$$

subject to

$$\sum_{\forall s_i \in VP(g_{ij}, g_{hk})} t(s_i) + t(g_{ij}) + t(g_{hk}) \leq t_{\max}(g_{ij}, g_{hk}), \forall g_{ij}, g_{hk} \in V_g \quad (7)$$

$$\prod_{\forall s_i \in VP(g_{ij}, g_{hk})} (1 - p(s_i))(1 - p(g_{ij}))(1 - p(g_{hk})) \leq 1 - plp(g_{ij}, g_{hk}), \quad \forall g_{ij}, g_{hk} \in V_g \quad (8)$$

$$\forall M \subseteq V_s, \forall VP(g_{ij}, g_{hk}) \in G, \forall g_{ij}, g_{hk} \in V_g$$

where $t(s_i)$ is the maximum queuing delay at the buffer of s_i for VP (g_{ij}, g_{hk}), $t(g_{ij})$ is the maximum queuing delay at the buffer of gateway g_{ij} for VP(g_{ij}, g_{hk}), $t_{\max}(g_{ij}, g_{hk})$ is the maximum end-to-end packet delay from g_{ij} to g_{hk}, $p(s_i)$ is the packet loss probability at s_i for VP(g_{ij}, g_{hk}), $p(g_{ij})$ is the packet loss probability at g_{ij} for VP(g_{ij}, g_{hk}), $plp(g_{ij}, g_{hk})$ is the end-to-end packet loss probability for pair (g_{ij}, g_{hk}), and VP(g_{ij}, g_{hk}) is a virtual path of pair (g_{ij}, g_{hk}).

Note that the left-hand side of constraint (8) can be approximated as

$$1 - \sum_{\forall s_i \in VP(g_{ij}, g_{hk})} p(s_i) - p(g_{ij}) - p(g_{hk}).$$

Thus, constraint (8) can be rewritten as

$$1 - \sum_{\forall s_i \in VP(g_{ij}, g_{hk})} p(s_i) - p(g_{ij}) - p(g_{hk}) \leq 1 - plp(g_{ij}, g_{hk}), \forall g_{ij}, g_{hk} \in V_g.$$

For a set of CLSs M and a set of VPs, the end-to-end packet delay and loss probability can be determined. Then we check whether obtained VPs satisfy

the given maximum end-to-end packet delay and packet loss probability constraints. Among all feasible M and VPs, a solution with minimum total network cost can be obtained.

IV. SOLUTION METHODS FOR THE UNCONSTRAINED OPTIMIZATION PROBLEM

In the following we will introduce two solution methods, the greedy method and the branch-and-bound method, for solving Problem 1.

A. A Greedy Method for Solving Problem 1

First, we simplify the objective function (4). Note that there is only one incoming and outgoing port at switch s_i for each gateway g_{ij} (see Fig. 8). Thus, the optimal value of objective function (4), z_{opt}, is composed of costs on link sets E_{gs} and E_{ss}. That is,

$$z_{\text{opt}} = z_{gs} + z_{ss},$$

where

$$z_{gs} = \sum_{i=1}^{|V_s|} \sum_{j \in N(s_i)} \sum_{h=1, h \neq i}^{|V_s|} \sum_{k \in N(s_h)} w(g_{ij}, g_{hk}) \times \{c(g_{ij}, s_i) + c(s_h, g_{hk})\}$$

and

$$z_{ss} = \min_{M \subseteq V_s} \sum_{\forall s_i, s_h \in V_{s_i}, i \neq h} \min_{\forall m \in M} w(s_i, s_h) \times \{d(s_i, m) + d(m, s_h)\}. \tag{9}$$

Note that the aggregated traffic flow $w(s_i, s_h)$ between s_i and s_h is

$$w(s_i, s_h) = \sum_{a \in N(s_i)} \sum_{b \in N(s_h)} w(g_{ia}, g_{hb}).$$

Since z_{gs} is a constant, Problem 1 is equivalent to (9).

At first, we apply the Floyd–Warshall algorithm to find the shortest paths for all gateway pairs. Since the shortest path from g_{ij} to g_{hk} must start at g_{ij} and goes to s_i, through the shortest path to s_h, and then to g_{hk}, the shortest distance $d(g_{ij}, g_{hk})$ can be written as

$$d(g_{ij}, g_{hk}) = c(g_{ij}, s_i) + d(s_i, s_h) + c(s_h, g_{hk}).$$

This means that we only must find the shortest paths for all switch pairs (s_i, s_h) on the subgraph $\hat{G} = (V_s, E_{ss})$. Let P be the set of the shortest paths for all switch pairs (s_i, s_h), initially. Next, we compute the traffic passing through the switch s_a, for all $s_a \in V_s$. Let $v(s_a)$ be the accumulated flows on switching node s_a for all switch pairs' shortest paths passing through s_a, and let $P_{(s_i, s_h)}$ denote the shortest path from s_i to s_h. Then $v(s_a)$ can be determined by traversing all switch pairs (s_i, s_h) shortest paths and incrementing $v(s_a)$ by aggregated traffic flow $w_{(s_i, s_h)}$ whenever s_i is visited. After all the shortest paths have been

traversed, we find the switch node with the maximum value of $v(s_a)$; i.e., find s^* such that

$$v(s^*) = \max\{v(s_a) | s_a \in V_s\}.$$

We say the switch node s^* is a candidate node for CLS, and then we remove the shortest paths passing through s^* from P. Repeat the same procedure for all shortest paths in P and find another switch node for CLS until P becomes empty or the number of candidate nodes is p.

The greedy algorithm is summarized as follows.

ALGORITHM 1.

Step 1. Apply the Floyd–Warshall algorithm to find all pairs' shortest paths for $\hat{G} = (V_s, E_{ss})$. Let P be the set that includes the shortest paths for all switch pairs (s_i, s_h). Set $L = \emptyset$.

Step 2. If $P = \emptyset$ or $|L| = p$, then stop. Otherwise, traverse every shortest path in P and compute $v(s_a)$ for each $s_a \in V_s - L$. Find s^* such that

$$v(s^*) = \max\{v(s_a) | s_a \in V_s - L\}.$$

Set $L = L \cup \{s^*\}$. Remove the shortest paths passing through s^* from P. Go to Step 2.

Let us consider the complexity of Algorithm 1. It costs $O(|V_s|^3)$ to find all pairs' shortest paths in Step 1. Traversing a shortest path takes $O(|V_s|)$ time. The number of all pairs' shortest paths is $O(|V_s|^2)$. Thus, Step 2 takes $O(p \cdot |V_s|^3)$. Therefore the computational complexity of Algorithm 1 is $O(p \cdot |V_s|^3)$.

EXAMPLE 1. Consider an ATM network with 12 gateways and 6 switch nodes as shown in Fig. 9.

Suppose that $w(g_{ij}, g_{hk}) = 1, \forall g_{ij}, g_{hk} \in V_g$. That is,

$$w(s_i, s_h) = \sum_{a \in N(s_i)} \sum_{b \in N(s_h)} w(g_{ia}, g_{hb}) = 4,$$

for $s_i = 1, \ldots, 6, s_h = 1, \ldots, 6$ and $s_i \neq s_h$. The costs of the links are given as

$$\begin{bmatrix} c(s_1,s_1) & c(s_1,s_2) & c(s_1,s_3) & c(s_1,s_4) & c(s_1,s_5) & c(s_1,s_6) \\ c(s_2,s_1) & c(s_2,s_2) & c(s_2,s_3) & c(s_2,s_4) & c(s_2,s_5) & c(s_2,s_6) \\ c(s_3,s_1) & c(s_3,s_2) & c(s_3,s_3) & c(s_3,s_4) & c(s_3,s_5) & c(s_3,s_6) \\ c(s_4,s_1) & c(s_4,s_2) & c(s_4,s_3) & c(s_4,s_4) & c(s_4,s_5) & c(s_4,s_6) \\ c(s_5,s_1) & c(s_5,s_2) & c(s_5,s_3) & c(s_5,s_4) & c(s_5,s_5) & c(s_5,s_6) \\ c(s_6,s_1) & c(s_6,s_2) & c(s_6,s_3) & c(s_6,s_4) & c(s_6,s_5) & c(s_6,s_6) \end{bmatrix} = \begin{bmatrix} 0 & 8 & 3 & 0 & 0 & 0 \\ 8 & 0 & 1 & 5 & 7 & 0 \\ 3 & 1 & 0 & 2 & 7 & 0 \\ 0 & 5 & 2 & 0 & 7 & 8 \\ 0 & 7 & 7 & 7 & 0 & 7 \\ 0 & 0 & 0 & 8 & 7 & 0 \end{bmatrix}$$

and $c(g_{ij}, s_i) = 1, i = 1, \ldots, 6, j = 1, 2$. The goal of this example is to locate three CLSs to 6 ATM switches such that the total CL service cost is minimized.

The detailed procedure to obtain the solution is given as follows.

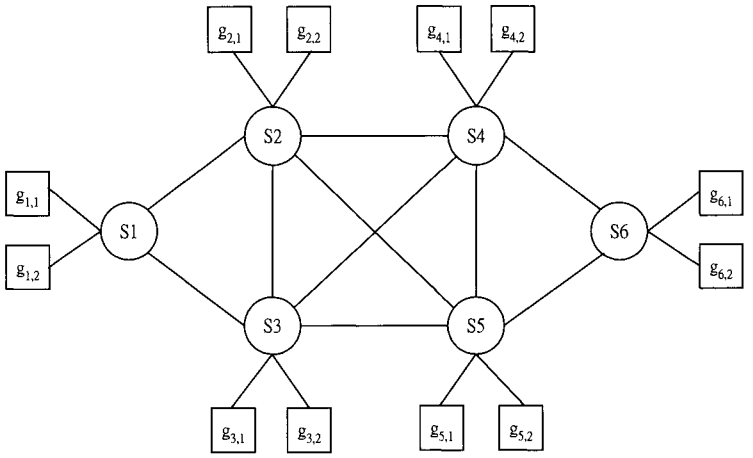


FIGURE 9 A sample network.

Step 1. Apply the Floyd–Warshall algorithm to find the all pairs shortest paths for the graph shown in Fig. 9. Then

$$P = \{(s_1, s_3, s_2), (s_1, s_3), (s_1, s_3, s_4), (s_1, s_3, s_5), (s_1, s_3, s_4, s_6), (s_2, s_3), (s_2, s_3, s_4), (s_2, s_5), (s_2, s_3, s_4, s_6), (s_3, s_4), (s_3, s_5), (s_3, s_4, s_6), (s_4, s_5), (s_4, s_6), (s_5, s_6)\}.$$

Step 2. Traverse every shortest path in P and find

$$v(s_1) = 4 \times 5 = 20, \quad v(s_2) = 4 \times 5 = 20, \quad v(s_3) = 4 \times 11 = 44,$$
$$v(s_4) = 4 \times 8 = 32, \quad v(s_5) = 4 \times 5 = 20, \quad v(s_6) = 4 \times 5 = 20,$$

and then $v(s_3) = \max\{20, 20, 44, 32, 20, 20\} = 44$. *Set* $L = L \cup \{s_3\} = \{s_3\}$ and

$$P = \{(s_2, s_5), (s_4, s_5), (s_4, s_6), (s_5, s_6)\}.$$

Go to Step 2.

Step 2. Traverse every shortest path in P and find

$$v(s_2) = 4 \times 1 = 4, v(s_4) = 4 \times 2 = 8, v(s_5) = 4 \times 3 = 12, v(s_6) = 4 \times 2 = 8$$

and then $v(s_5) = \max\{4, 8, 12, 8\} = 12$. Set $L = L \cup \{s_5\} = \{s_3, s_5\}$ and $P = \{(s_4, s_6)\}$. Go to Step 2.

Step 2. Traverse every shortest path in P and find

$$v(s_4) = 4 \times 1 = 4, v(s_6) = 4 \times 1 = 4$$

and then $v(s_4) = \max\{4, 4\} = 4$. Set $L = L \cup \{s_4\} = \{s_3, s_4, s_5\}$ and $P = \emptyset$. Go to Step 2.

Step 2. Since $P = \emptyset$ (or $|L| = 3$), and then Stop.

The optimal solution for Example 1 is $L = \{s_3, s_4, s_5\}$. The switches s_3, s_4, and s_5 are selected as the CLSs. A virtual shortest path tree rooted at $g_{1,x}$ for routing CL data traffic is shown in Fig 10.

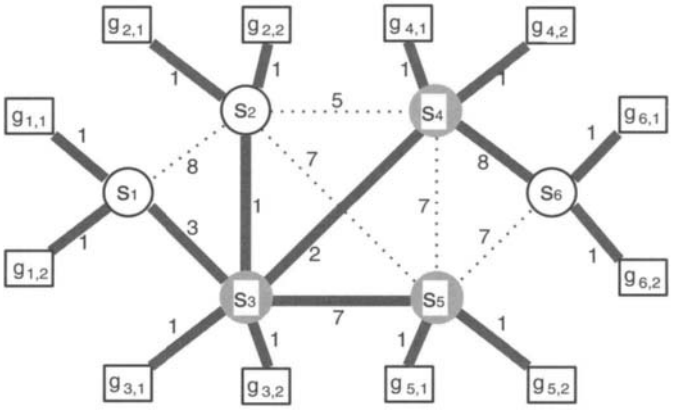


FIGURE 10 A virtual shortest path tree.

The optimal virtual path connections for all switch pairs are

$$P = \{(s_1, s_3, s_2), (s_1, s_3), (s_1, s_3, s_4), (s_1, s_3, s_5), (s_1, s_3, s_4, s_6), (s_2, s_3), (s_2, s_3, s_4), (s_2, s_5), (s_2, s_3, s_4, s_6), (s_3, s_4), (s_3, s_5), (s_3, s_4, s_6), (s_4, s_5), (s_4, s_6), (s_5, s_6)\}.$$

In Algorithm 1, if we only check whether $P = \emptyset$ in Step 2, then we can obtain set L^*, which contains all candidate locations of CLSs. It is easy to verify that the total transmission cost for node set L^* is a lower bound of Eq. (9). This is because all gateway pairs (g_{ij}, g_{hk}) use their shortest path to transport CL data via at least a CLS. Thus, we can conclude that:

1. If $|L^*| = p$, then L^* is exactly the optimal solution for the objective function (9).
2. If $|L^*| < p$, then any switch node set containing L^* with size p will be optimal solution. In fact, it is not necessary to locate so many CLSs in the network.
3. If $|L^*| > p$, then set L, the solution found by Algorithm 1 is a near-optimal solution. This is because the first $p(|L| = p)$ nodes have the largest accumulated weight, and the near-optimality is a direct result of transporting the majority of traffic volume along their shortest paths.

B. A Branch-and-Bound Method for Solving Problem 1

In the previous subsection, Algorithm 1 based on the greedy method may not find the optimal solution for the case of $|L^*| > p$. In this subsection, we will introduce a branch-and-bound method for finding the optimal solution. Initially, we apply Algorithm 1 to obtain a current optimal solution L^*. If $|L^*| > p$, then apply the following branch-and-bound method. Otherwise, the optimal solution is found. In the branch-and-bound method, a possible tree organization for the state space of the network model is a binary tree in which a left branch represents the inclusion of a particular switch node (located as a CLS)

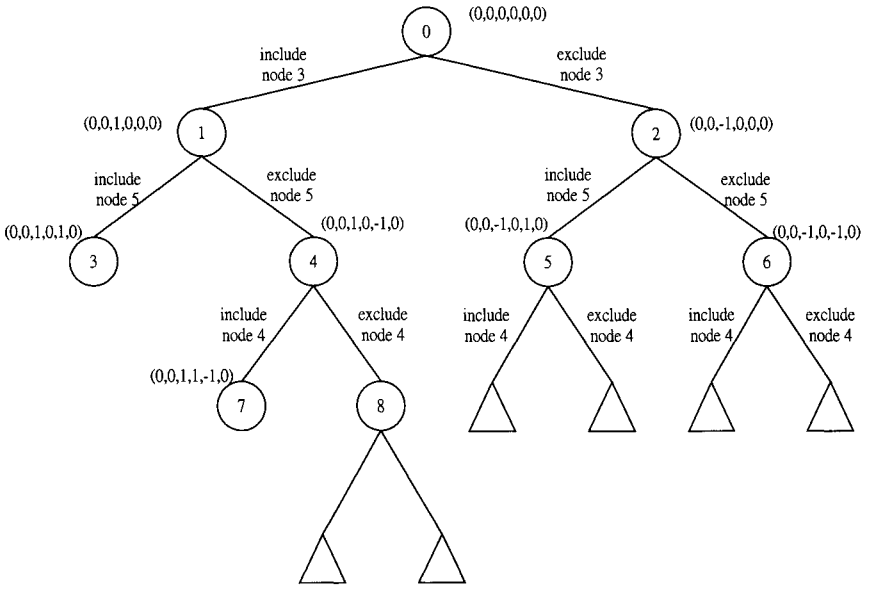


FIGURE 11 A state space tree.

while the right branch represents the exclusion of that switch node (not located as a CLS). Figure 11 represents a state space tree for the network model shown in Fig. 9. Note that many state space trees are possible for a given problem formulation. Different trees differ in the order in which decisions are made. In Fig. 11, we first consider node 3 (the first node obtained by Algorithm 1) and then consider node 5 (the second node obtained by Algorithm 1). That is, the order of nodes for consideration is the same as the order of nodes put in set L in Algorithm 1.

For each node i in the state space tree, we maintain its lower bound $\text{lb}(i)$ and an array $a_i(s_j)$, $j = 1, \ldots, |V_s|$, to record their status, where

$$a_i(s_j) = \begin{cases} 1, & \text{if node } s_j \text{ is included,} \\ -1, & \text{if node } s_j \text{ is excluded,} \\ 0, & \text{otherwise (node } s_j \text{ is in the don't care condition).} \end{cases}$$

Initially, let array $a_0(s_j) = 0$, $\forall j$, for node 0 and

$$\text{lb}(0) = \sum_{\forall s_i, s_h \in V_s, i \neq h} w(s_i, s_h) \times d(s_i, s_h),$$

where

$$w(s_i, s_h) = \sum_{a \in N(s_i)} \sum_{b \in N(s_h)} w(g_{ia}, g_{hb})$$

$$d(s_i, s_h) = \text{ the shortest distance from } s_i \text{ to } s_h.$$

Note that lb(0) is the lower bound of Eq. (9). At the first branch step, two child nodes are generated. In the left child (node 1), the status of a certain

switching node, e.g., switch node 3, is set to be included (i.e., set $a_1(s_3) = 1$ and $a_1(s_j) = a_0(s_j), \forall j, j \neq 3$). On the other hand, in the right child, switch node 3 is set to be excluded (i.e., set $a_2(s_3) = -1$ and $a_1(s_j) = a_0(s_j), \forall j, j \neq 3$). The following child nodes will be further individually split into two nodes in the same way, e.g., include switch node 5 and exclude switch node 5. The status of the other nodes inherits their parent's status (i.e., $a_i(s_k) = a_{p(i)}(s_k)$, where node i is a child of $p(i)$). A node is said to be a leaf node in state space tree if its status has p number of 1's. For example, node 7 in Fig. 11 is a leaf node. In this way a state-space tree can be generated.

For every nonleaf node, a lower bound $\text{lb}(i)$ can be found as follows. If there always exists at least one unexcluded intermediate switching node on its shortest paths for all node pairs, then the lower bound $\text{lb}(i)$ for node i in state space tree is just the same as its parent node (e.g., the lower bound of the left child will always inherit from its parent). Otherwise, the $\text{lb}(i)$ for node i should be updated. That is, if all the intermediate switching nodes along a gateway pair's shortest path are excluded, another shortest path with an unexcluded node lying on it should be found instead of the original one and compute its lower bound. For example, in Fig. 11, node 2 has the status $(a_2(s_1), a_2(s_2), \ldots, a_2(s_6)) = (0, 0, -1, 0, 0, 0)$, then the virtual connection from s_1 to s_5 cannot be s_1, s_3, s_5 but s_1, s_2, s_5. This is because switch node 3 is excluded, i.e., $a_2(s_3) = -1$. Thus, the shortest path from s_i to s_j with restriction $(a_i(s_1), a_i(s_2), \ldots, a_i(s_{|V_s|}))$ can be determined by $\min\{d(s_i, m) + d(m, s_j) | \forall a_i(m) \neq -1\}$.

To facilitate the generation of the state space tree, a data structure *heap,* called live-node heap, is used to record all live nodes that are waiting to be branched. The search strategy of this branch-and-bound is least cost first search. That is, the node, say node x, selected for the next branch is the live node whose $\text{lb}(x)$ value is the smallest among all of the nodes in the live-node heap. (Note that the minimal value is at the top of the heap.) Two child nodes of node x are then generated. Traversing the state space tree begins from root. Initially, apply the greedy algorithm (Algorithm 1) and obtain a current optimal value UB. When a complete solution (leaf node y) is reached, compute its total transmission cost TTC_y and compare TTC_y with that of the current optimal value UB. If $\text{TTC}_y < \text{UB}$ then the current optimal solution will be replaced, i.e., $\text{UB} \leftarrow \text{TTC}_y$. If a node v in live-node heap satisfies $\text{lb}(v) \geq \text{UB}$, then node v is bounded since further branching from v will not lead to a better solution. When live-node heap becomes empty, we obtain the optimal solution with optimal value UB. The detailed algorithm is presented as follows.

ALGORITHM 2. PROCEDURE OF BRANCH-AND-BOUND (G', L, p).

begin
  Apply Algorithm 1 to find L and its TTC
  $\text{UB} \leftarrow \text{TTC}$
  Set $(a_0(s_1), a_0(s_2), \ldots, a_0(s_{|V_s|})) = (0, 0, \ldots, 0)$ and $\text{lb}(0) = z_{ss}$
  live-node heap $Q \leftarrow \{0\}$
  while $Q \neq \emptyset$ **do**
  begin

```
remove i from Q such that lb(i) = min{lb(v)|v ∈ Q}
if i is a leaf node then
  begin
    compute TTC_i
    if TTC_i < UB then UB ← TTC_i
    set (a_i(s_1), a_i(s_2), ..., a_i(s_|V_s|)) is the current solution
  end
else
  begin
    generate left child v and right child u
    set lb(v) ← lb(i) and compute lb(u)
    set Q ← Q ∪ {v} ∪ {u}
  end
Q ← Q − {k | lb(k) ≥ UB, ∀k ∈ Q}
end
```

V. SOLUTION METHODS FOR THE CONSTRAINED OPTIMIZATION PROBLEM

In the previous section, we focused on the virtual topology architecture only. An important factor, the *capacity* of each link, is ignored. We all know that the total traffic flows through a link should not exceed its available bandwidth. In this section, we present a heuristic algorithm to solve Problem 1 subject to the new constraint. It is a variant of *multicommodity flow problem*. Then a heuristic algorithm [34] is introduced for Problem 2. When capacities allocation and feasibility testing are added to the algorithm for Problem 2, then Problem 3 can be solved.

A. A Heuristic Algorithm for Solving Problem 1 Subject to Bandwidth Constraints

The problem of *enumerating simple paths in nondecreasing order* is a critical subproblem when we are trying to find a virtual path that has available bandwidth for a given traffic flow. Such a listing is very useful in the situation where we are interested in obtaining the shortest path that satisfies some complex set of constraints. One possible solution to such a problem would be to generate in nondecreasing order of path length all paths between the two vertices. Each path generated could be tested against the other constraints and the first path satisfying all these constraints would be the path we were looking for.

1. Enumerating Simple Paths in Nondecreasing Order

In the following we will demonstrate how to list all the possible simple paths between two vertices in order of nondecreasing path length.

Let $G = (V, E)$ be a digraph with n vertices. Let v_1 be the source vertex and v_n the destination vertex. We shall assume that every edge has a positive cost. Let $p_1 = r[0], r[1], \ldots, r[k]$ be a shortest path from v_1 to v_n; i.e., p_1 starts

at $v_{r[0]} = v_1$, goes to $v_{r[1]}$, and then goes to $v_{r[2]}, \ldots, v_n = v_{r[k]}$. If P is the set of all simple v_1 to v_n paths in G, then it is easy to see that every path in $P - \{p_1\}$ differs from p_1 in exactly one of the following k ways:

(1) It contains the edges $(r[1], r[2]), \ldots, (r[k-1], r[k])$ but not $(r[0], r[1])$.
(2) It contains the edges $(r[2], r[3], \ldots, (r[k-1], r[k])$ but not $(r[1], r[2])$.
...
(k) It does not contain the edge $(r[k-1], r[k])$.

More compactly, for every path p in $P - \{p_1\}$, there is exactly one j, $1 \leq j \leq k$, such that p contains the edges $(r[j], r[j+1]), \ldots, (r[k-1], r[k])$ but not $(r[j-1], r[j])$.

The set of paths $P - \{p_1\}$ may be partitioned into k disjoint sets $P^{(1)}, \ldots, P^{(k)}$ with set $P^{(j)}$ containing all paths in $P - \{p_1\}$ satisfying condition j above, $1 \leq j \leq k$.

Let $p^{(j)}$ be a shortest path in $P^{(j)}$ and let q be the shortest path from v_1 to $v_{r[j]}$ in the digraph obtained by deleting from G the edges $(r[j-1], r[j]), (r[j], r[j+1]), \ldots, (r[k-1], r[k])$. Then one readily obtains $p^{(j)} = q, r[j], r[j+1], \ldots, r[k] = n$. Let $p^{(l)}$ have the minimum length among $p^{(1)}, \ldots, p^{(k)}$. Then $p^{(l)}$ also has the least length amongst all paths in $P - \{p_l\}$ and hence must be a second shortest path. The set $P^{(l)} - \{p^{(l)}\}$ may now be partitioned into disjoint subsets using a criterion identical to that used to partition $P - \{p_1\}$. If $p^{(l)}$ has k' edges, then this partitioning results in k' disjoint subsets. We next determine the shortest paths in each of these k' subsets. Consider the set Q, which is the union of these k' shortest paths and the paths $p^{(1)}, \ldots, p^{(l-1)}, \ldots, p^{(k)}$. The path with minimum length in Q is a third shortest path. The corresponding set may be further partitioned. In this way we may successively generate the v_1 to v_n paths in nondecreasing order of path length.

At this point, an example would be instructive. The generation of the v_1 to v_6 path of the graph is shown in Fig. 12.

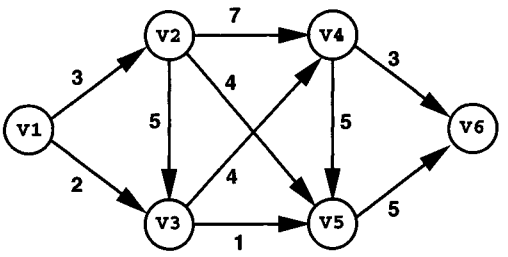
2. The Heuristic Algorithm

A brief description of the algorithm appears as the following procedure:

Step 1. Initialization: Bind the given capacity or maximum available bandwidth $b(i, j)$ to each link (i, j). Establish a variable $v(s_i)$ with the initial value of zero on each switching node s_i. Maintain a variable $x(i, j)$ on each link (i, j) to record the available residual bandwidth. $x(i, j)$ is set to $b(i, j)$ initially.

Step 2. For all the gateway pairs, compute their *legal* shortest paths' distance D subject to the constraint: the available residual bandwidth on all the edges along the shortest path of gateway pair $(g_{s,t}, g_{u,v})$ should not be below $w(g_{s,t}, g_{u,v})$.

Step 3. Sort the value $w(g_{s,t}, g_{u,v}) D(g_{s,t}, g_{u,v})$ for all the pairs in a decreasing order.



shortest path	cost	included edges	excluded edges	new path
V1,V3,V5,V6	8	none	none	
		none	(V5,V6)	V1,V3,V4,V6=9
		(V5,V6)	(V3,V5)	V1,V2,V5,V6=12
		(V3,V5) (V5,V6)	(V1,V3)	V1,V2,V3,V5,V6=14
V1,V3,V4,V6	9	none	(V5,V6)	
		none	(V4,V5) (V5,V6)	
		(V4,V6)	(V3,V4) (V5,V6)	V1,V2,V4,V6=13
		(V3,V4) (V4,V6)	(V1,V3) (V5,V6)	V1,V2,V3,V4,V6=15
V1,V2,V5,V6	12	(V5,V6)	(V3,V5)	
		(V5,V6)	(V2,V5) (V3,V5)	V1,V3,V4,V5,V6=16
		(V2,V5) (V5,V6)	(V1,V2) (V3,V5)	
V1,V2,V4,V6	13	(V4,V6)	(V3,V4) (V5,V6)	
		(V4,V6)	(V2,V4) (V3,V4) (V5,V6)	
		(V2,V4) (V4,V6)	(V1,V2) (V3,V4) (V5,V6)	
V1,V2,V3,V5,V6	14	(V3,V5) (V5,V6)	(V1,V3)	
		(V3,V5) (V5,V6)	(V2,V3) (V1,V3)	
		(V2,V3) (V3,V5) (V5,V6)	(V1,V2) (V5,V6)	
V1,V2,V3,V4,V6	15	(V3,V4) (V4,V6)	(V1,V3) (V5,V6)	
		(V3,V4) (V4,V6)	(V2,V3) (V1,V3) (V5,V6)	
		(V2,V3) (V3,V5) (V5,V6)	(V1,V2) (V1,V3) (V5,V6)	
V1,V3,V4,V5,V6	16	(V5,V6)	(V2,V5) (V3,V5)	
		(V5,V6)	(V4,V5) (V2,V5) (V3,V5)	
		(V4,V5) (V5,V6)	(V3,V4) (V2,V5) (V3,V5)	V1,V2,V4,V5,V6=20
		(V3,V4) (V4,V5) (V5,V6)	(V1,V3) (V2,V5) (V3,V5)	V1,V2,V3,V4,V5,V6=22

FIGURE 12 Example of enumerating simple paths in nondecreasing order.

Step 4. Traverse all the pairs' legal paths in the sorted order. Whenever passing by a switching node s_i, add $w(g_{s,t}, g_{u,v})\ \{d(g_{s,t}, s_i) + d(s_i, g_{u,v})\}$ to $v(s_i)$. Whenever going through a link (i, j), subtract $x(i, j)$ from $w(g_{s,t}, g_{u,v})$. If the latter pairs cannot go through their legal shortest paths for the sake of congestion (the residual bandwidth is not enough), then use the algorithm of the previous subsection to find the legal detouring paths.

Step 5. Select a switching node s^* with a maximum value of $v(s_i)$. Traverse all those pairs' legal shortest path passed by s^*, subtracting $b(i, j)$ from traffic flow w whenever going through the link (i, j).

Step 6. Mark all these pairs and set $x(i, j)$ to $b(i, j)$ for all links (i, j).

Step 7. Go to step 2 for all those unmarked pairs until p CLSs are selected or all the pairs are marked. If the p CLSs are selected before all the pairs are marked, the remaining unmarked pairs' legal shortest paths can be searched like the method described above subsequentially.

B. A Heuristic Algorithm for Solving Problem 2

A two-phase algorithm proposed by Eom *et al.* [34] can be applied to solve Problem 2 as follows:

Phase 1. A perference, denoted as node metric, that the switch node be selected as a CLS is defined and calculated. This phase is called the node-metric calculation phase.

Phase 2. There are N iterations in phase 2. In iteration i, the switch nodes are divided into i clusters. In each cluster, a node is chosen as a CLS. Then we have i CLSs for iteration i, and the total cost of this configuration is computed. After N iterations, an optimal configuration with minimum total cost is found. That is, an optimal number of connectionless servers and their locations are determined.

1. Node-Metric Calculation Phase

Consider an ATM network modeled by $G' = (V_s, E_{ss})$. Let N_g be a set of switch nodes from which the gateway node is connected. Let matrix $T' = [w(s_i, s_j)]_{k\times k}$, where $w(s_i, s_j)$ represents the traffic flow between two switches $s_i \in N_g$ and $s_j \in N_g$, and $|N_g| = k$. Thus, the total traffic flow from node s_i is $\sum_{s_j \in N_g} w(s_i, s_j)$. Let $C = [c(i, j)]_{u\times u}$, where $c(i, j)$ represents the cost per unit flow of the link (i, j) and $|V_s| = u$. We can apply the Floyd–Warshall algorithm to find the all pairs' shortest paths. Let $\mathrm{VP}(s_i, s_j)$ be the shortest path from s_i to s_j and $C(\mathrm{VP}_{(s_i,s_j)})$ be the cost of $\mathrm{VP}_{(s_i,s_j)}$. Thus, the total transmission cost for the shortest path starting from s_i is $\sum_{s_j \in N_g} C(\mathrm{VP}_{(s_i,s_j)})$.

The node metric for switch s_i is defined as

$$m(s_i) = \frac{\sum_{s_j \in N_g} w(s_i, s_j)}{\sum_{s_j \in N_g} C(VP_{(s_i,s_j)})}. \tag{10}$$

Note that if more traffic traverses the switch node s_i, it is more preferable to select s_i. On the other hand, if the cost of the connection from s_i is higher, it should be avoided. Therefore, the preference is to place the CLS function with the switch node with the higher node-metric value.

In this phase, the node-metric $m(s_i)$ is computed for each node s_i. Then sort $m(s_i)$, $s_i \in V_s$ such that

$$m(s_{(1)}) \geq m(s_{(2)}) \geq \cdots \geq m(s_{(|V_s|)}).$$

For example, consider an ATM network as given in Example 1. Then $N_g = \{s_1, s_2, \ldots, s_6\}$:

$$T' = \begin{bmatrix} 0 & 4 & 4 & 4 & 4 & 4 \\ 4 & 0 & 4 & 4 & 4 & 4 \\ 4 & 4 & 0 & 4 & 4 & 4 \\ 4 & 4 & 4 & 0 & 4 & 4 \\ 4 & 4 & 4 & 4 & 0 & 4 \\ 4 & 4 & 4 & 4 & 4 & 0 \end{bmatrix} \text{ and } C = \begin{bmatrix} 0 & 8 & 3 & 0 & 0 & 0 \\ 8 & 0 & 1 & 5 & 7 & 0 \\ 3 & 1 & 0 & 2 & 7 & 0 \\ 0 & 5 & 2 & 0 & 7 & 8 \\ 0 & 7 & 7 & 7 & 0 & 7 \\ 0 & 0 & 0 & 8 & 7 & 0 \end{bmatrix}.$$

Apply the Floyd–Warshall algorithm to find the all pairs' shortest paths as

$$P = \{(s_1, s_3, s_2), (s_1, s_3), (s_1, s_3, s_4), (s_1, s_3, s_5), (s_1, s_3, s_4, s_6), (s_2, s_3), (s_2, s_3, s_4), (s_2, s_5), (s_2, s_3, s_4, s_6), (s_3, s_4), (s_3, s_5), (s_3, s_4, s_6), (s_4, s_5), (s_4, s_6), (s_5, s_6)\}.$$

Thus,

$$m(s_1) = \frac{\sum_{j=2}^{6} w(s_1, s_j)}{\sum_{j=2}^{6} C(VP_{(s_1,s_j)})} = \frac{4+4+4+4+4}{4+3+5+10+13} = \frac{20}{35}.$$

In this way, we obtain $m(s_2) = \frac{20}{26}$, $m(s_3) = \frac{20}{23}$, $m(s_4) = \frac{20}{25}$, $m(s_5) = \frac{20}{38}$, $m(s_6) = \frac{20}{49}$. Sort $m(s_i), i = 1, 2, \ldots, 6$. We have the sequence $m(s_3), m(s_4), m(s_2), m(s_1), m(s_5), m(s_6)$.

2. Clustering Phase

In the clustering phase, there are three steps (i.e., clustering, center finding, and cost computing) in an iteration and $|V_s|$ iterations in the algorithm. The steps in iteration i can be described as follows.

a. Clustering Step (The Switch Nodes are Divided into i Clusters.)

The i nodes $s_{(1)}, s_{(2)}, \ldots, s_{(i)}$ with the highest i node metrics are selected. Let $S = \{s_{(1)}, s_{(2)}, \ldots, s_{(i)}\}$ and $R = V_s - S$. Initially, set i clusters as $S(s_{(k)}) = \{s_{(k)}\}, k = 1, \ldots, i$. For each node $s_r \in R$, find i shortest paths $\mathrm{VP}_{(s_r,s_{(1)})}$, $\mathrm{VP}_{(s_r,s_{(2)})}, \ldots, \mathrm{VP}_{(s_r,s_{(i)})}$ from s_r to $s_{(1)}, s_{(2)}, \ldots, s_{(i)}$, respectively, and then node s_j such that $C(\mathrm{VP}^*_{(s_r,s_j)}) = \min\{C(\mathrm{VP}_{(s_r,s_{(1)})}), C(\mathrm{VP}_{(s_r,s_{(2)})}), \ldots, C(\mathrm{VP}_{(s_r,s_{(i)})})\}$. Thus, we assign node s_r to cluster $S(s_j)$ i.e., $S(s_j) = S(s_j) \cup \{s_r\}$. This is because switch node s_r has a minimum shortest path to s_j. By assigning all nodes in R, we have i clusters $S(s_{(1)}), S(s_{(2)}), \ldots, S(s_{(i)})$ and each cluster has at least one node.

For example, consider iteration $2(i = 2)$. Then $S = \{s_3, s_4\}$ and $R = \{s_1, s_2, s_5, s_6\}$. For s_1, find the shortest paths $\mathrm{VP}_{(s_1,s_3)} = (s_1, s_3)$ and $\mathrm{VP}_{(s_1,s_4)} = (s_1, s_3, s_4)$ and compute their costs. We have $C(\mathrm{VP}_{(s_1,s_3)}) = 3$, and $C(\mathrm{VP}_{(s_1,s_4)}) = 5$. Thus, set $S(s_3) = \{s_3\} \cup \{s_1\}$. In this way, we obtain 2 clusters, $S(s_3) = \{s_1, s_2, s_3\}$ and $S(s_4) = \{s_4, s_5, s_6\}$.

b. Center-Finding Step (For Each Cluster, a Switch Node is Chosen as a CLS.)

For each cluster $S(s_{(k)})$, we consider every node in the cluster is a candidate for CLS. Thus, for each $s \in S(s_{(k)})$, we find the shortest paths from s to all the nodes in $S(s_{(k)})\backslash\{s\}$ and sum up all shortest paths' cost as its weight. A node in $S(s_{(k)})$ with the least weight is selected as a CLS. Thus, a configuration with i CLSs is constructed.

For example, in iteration $i = 2$, two clusters, $S(s_3) = \{s_1, s_2, s_3\}$ and $S(s_4) = \{s_4, s_5, s_6\}$, are obtained. By computing all pairs' shortest paths for cluster $S(s_3)$, node s_3 is chosen as a CLS. Similarly, node s_5 is selected as a CLS for cluster $S(s_4)$. Figure 13 shows the CLS locations for two clusters.

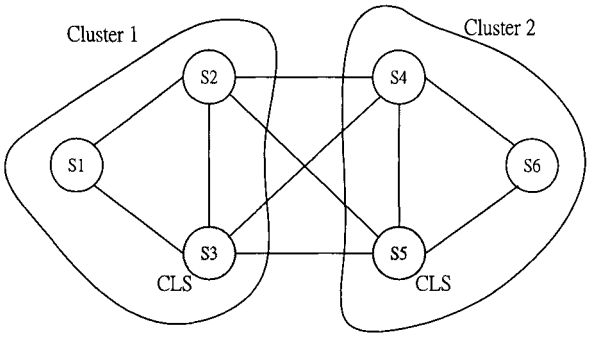


FIGURE 13 Clustering and center finding.

c. Cost-Computing Step (Compute the total cost of the configuration i.)

Note that in the previous step a configuration i is found such that local transmission cost is minimized. In this step the total cost will be computed. We assume that i CLSs are fullymeshed by connecting every two CLSs using their shortest path. Thus, a virtual topology for an ATM network is defined as:

1. Every cluster forms a rooted tree with a CLS as its root.
2. The CLSs are fully meshed.

For example, configuration $i = 2$ is shown as in Fig. 14.

Under this virtual topology, there exists a unique path between any gateway pair (g_{ij}, g_{hk}). Then the total cost for configuration i can be determined as

$$\sum_{\forall g_{ij}, g_{hk} \in V_g} C_{cl}(g_{ij}, g_{hk}) + \sum_{\forall s_i \in S} C_s(s_i),$$

where $C_{cl}(g_{ij}, g_{hk})$ is the transmission cost for gateway pair (g_{ij}, g_{hk}) and $C_s(s_i)$ is the cost of CLS s_i.

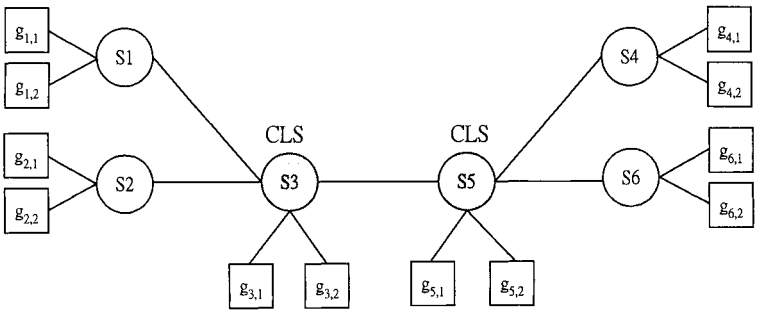


FIGURE 14 A sample network.

Thus the algorithm for phase 2 can be summarized as follows.

ALGORITHM 3. ALGORITHM FOR CLUSTERING PHASE

input: a graph G', $T' = [w(s_i, s_j)]_{k\times k}$, $[c(i, j)]_{u\times u}$ and $m(s_{(1)}) \geq m(s_{(2)}) \geq \cdots \geq m(s_{(|V_s|)})$;
output: an optimal number of connectionless servers and their locations

For $i = 1$ to $|V_s|$ do

begin
1. Clustering (The switch nodes are divided into i clusters.)
2. Center finding (For each cluster, a switch nodes is chosen as a CLS.)
3. Cost computing (Compute the total cost of the configuration i.)
end

Find a configuration with minimum cost among $|V_s|$ configuration.

C. A Heuristic Algorithm for Solving Problem 3

Problem 3 can be solved by a heuristic algorithm proposed by Eom *et al.* [34]. The algorithm is the same as that presented in the last subsection. Besides, in the cost-computing subphase of Phase 2, the capacities of all virtual paths satisfying the given maximum delay constraints and packet loss probability constraints are computed. For the detailed method one can refer to the paper [34]. After the capacities are determined, we can compute the total network cost for the configuration with i CLSs.

VI. CONSTRUCTION OF VIRTUAL OVERLAYED NETWORK

We have mentioned that once the locations of the CLSs have been determined, the virtual path connections between all the gateway pairs are also determined. Since the CL traffic flow for a gateway pair may pass by more than one CLS, we must select one CLS to handle this CL traffic. In the next step, we determine the locations of endpoints for each VP along the virtual path connection, and then allocate them proper bandwidth to them according to the traffic pattern matrix.

It is worth noting that it is possible for us to have more than one CLS as the intermediate endpoint, despite the fact that the more the intermediate endpoints, the more the delay for data processing and routing (and segmentation and reassembly for the reassembly mode server [12]) at each CLS. However, if we do always have only *one* CLS as the intermediate endpoint for all the gateway pairs, the required minimum number of VPIs will be greater than that of having *more than one,* since all the VPs with the same endpoints can be statistically multiplexed to one large VP to utilize the physical trunk.

If the CL traffic flow for gateway pair $(g_{s,t}, g_{u,v})$ passed by exactly one CLS s_m, say, then there is only one choice: allocate bandwidth $w(g_{s,t}, g_{u,v})$ for the virtual path connections from $g_{s,t}$ to s_m and from s_m to $g_{u,v}$. For example, as shown in Fig. 15, if the 3-CLS is $\{s_3, s_7, s_9\}$, we will set up two VPCs for

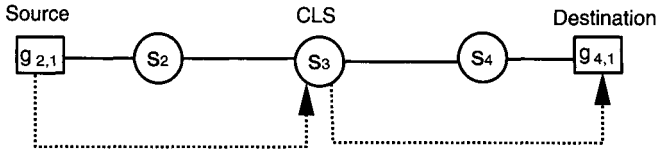


FIGURE 15 Pass by only one CLS.

$(g_{2,1}, g_{4,1})$ from $g_{2,1}$ to s_3 and from s_3 to $g_{4,1}$ (i.e., the intermediate endpoint is on s_3). The allocated bandwidth for them is proportional to $w(g_{2,1}, g_{4,1})$.

If the flow passed by more than one intermediate CLS, then it can have more than one intermediate endpoint. For example, the CL traffic flow from $g_{2,1}$ to $g_{6,1}$ will go through the virtual path $\{(g_{2,1}, s_2), (s_2, s_3), (s_3, s_{11}), (s_{11}, s_7), (s_7, s_6), (s_6, g_{6,1})\}$ with two CLSs, i.e., s_3 and s_7, lying on it. In this case, we have three choices to set up the VPCs, as shown in Fig. 16. The first choice is to have both s_3 and s_7 as the intermediate endpoints, the second choice is to only have s_3, and the third is to have s_7. Since the last two choices have the same effect, we can consider it has only two alternatives: have *one* or *more than one* CLS as the intermediate endpoint.

The first alternative (having only one CLS as the intermediate endpoint) has the advantage of less delay but is defective is scalability. On the other hand, the second alternative (having more than one CLS as the intermediate endpoints) will have the benefits of bandwidth efficiency and scalability (a side effect of statistical multiplexing) but will suffer from more delay. To take scalability into

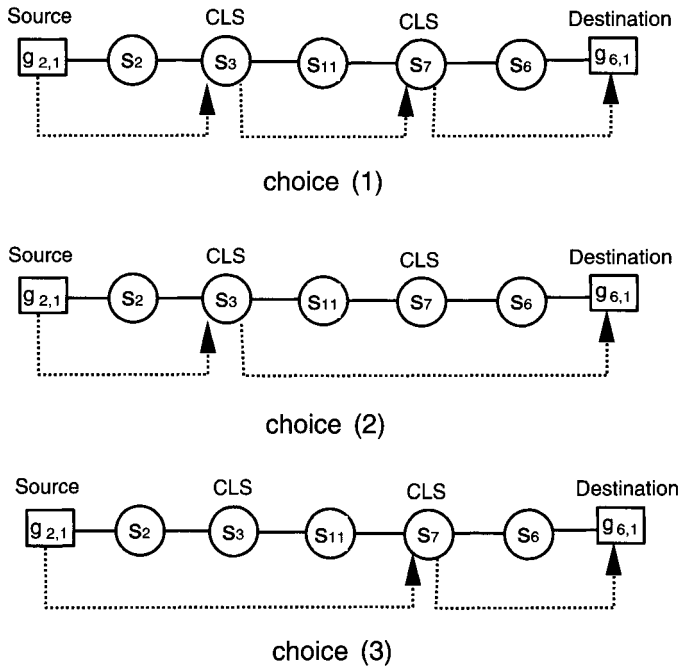


FIGURE 16 Pass by more than one CLS.

account is to consider the limited number of VPIs an ATM switch can provide. The size of a VPI field in a UNI cell header is only 8 bits (12 bits for NNI). An example of this limitation occurs in an 8-CLS network where 16 nodes are connected to a switch S. Since VP is unidirectional, the number of VPs going through S may be up to $2 \times 16 \times 8 = 256$, which is the maximum available VPIs without any one remaining for CO traffic flow.

The tradeoffs are very similar to that between the *direct* and *indirect* approaches for CL data service involving balancing efficiency with delay. The problem can be solved easily by a hybrid approach for setting up the proper VPs. We can choose a threshold carefully such that the delay on CLSs can be reduced as much as possible. If the intensity of traffic flow is greater than the predefined threshold, then we can adopt the first alternative; otherwise, we can adopt the latter one. In the case of having only one CLS as the intermediate endpoint, for the sake of load balancing, we prefer to select the one with the less number of endpoints on it. On the other hand, a maximum value on the number of CLS hops for each VPC should also be restricted to satisfy the delay constraint.

First, we chose a threshold T carefully for the bandwidth allocation. If $w(g_{s,t}, g_{u,v}) > T$ then we adopt the first alternative; otherwise, we adopt the later one. Second, we concentrate all the VPCs with the same endpoints to one VPC. Since the bursty nature of CL data traffic may cause underutilization of the bandwidth allocated to VPCs, we can preallocate less bandwidth, say, 70 or 80%, for the aggregated flows of those original VPCs. If a burst exceeds the preallocated bandwidth at a front endpoint, it is buffered in the CLS's memory for a short term. When the queue length or data rate exceeds a certain threshold (e.g., in the rush hours [16]), bandwidth is renegotiated, to add a given increment to the preallocated one. An algorithm should be well defined to request additional increments while progressively large queue depths or data rates are reached. Bandwidth is later released while utilization goes below a specified threshold. Consequently, in addition to the efficient VPI usage, the hybrid approach will also result in a higher utilization and flexibility. The appropriate choice of T depends critically on the traffic characteristic. Thus, the parameters must be tuned to the particular environment.

By backtracking the virtual path connection for each gateway pair $(g_{a,b}, g_{c,d})$, the CLSs $M[1], M[2], \ldots, M[n]$ consecutively being passed by the VPC of $(g_{a,b}, g_{c,d})$ can be obtained. Let $\ll i, j \gg$ denote the VP for the shortest path from i to j. The following pseudocode illustrates how to layout a virtual overlay network.

```
0  Procedure LAYOUT (G, p, T, h)
1    for i = 1 to p do v(M[i]) ← 0
2    for each gateway pair (g_{a,b}, g_{c,d}) do
3      <M[1], M[2], ..., M[n] > ← backtracking (g_{a,b}, g_{c,d})
4    for those gateway pairs with weight greater than T do
5      v(m) = min{v(M[i])|i = 1, ..., n}
6      setup ≪ g_{a,b}, m ≫ and ≪ m, g_{c,d} ≫
7      v(m) ← v(m) + w(g_{a,b}, g_{c,d})
8    sort the remaining gateway pairs by n (the number of CLS passed by)
```

```
 9    for each gateway pair (g_{a,b}, g_{c,d}) do
10      if n ≤ h then
11        SET-UP-VCP (g_{a,b}, g_{c,d}, M, n)
12      else
13        sort M by v(M[i])
14        SET-UP-VPC (g_{a,b}, g_{c,d}, M, h)

 0  Procedure SET-UP-VPC (x, y, N, t)
 1    set up ≪ x, N[1] ≫
 2    v(N[1]) ← v(N[1]) + w(x, y)
 3    for i = 1 to t − 1 do
 4      if no VP from N[i] to N[i + 1] then
 5        setup ≪ N[i], N[i + 1] ≫
 6        v(N[i + 1]) ← v(N[i + 1])+ w(x, y)
 7    set up ≪ N[t], y ≫
```

VII. CONCLUSIONS AND DISCUSSIONS

Connectionless data service in an ATM-based B-ISDN can be realized *directly* by means of the *connectionless service function,* which is provided in *connectionless servers.* In this chapter, we have considered how to locate a certain amount of connectionless servers among the switching nodes in a public ATM network for the internetworking of connectionless data networks. The problem is formulated as network optimization problems, which are similar to the *p-median* problem. Two optimization techniques, one based on the *greedy method* and the other using the *branch-and-bound strategy,* for determining the locations of connectionless servers are presented. By finding the optimal locations of connectionless servers, an optimal virtual overlay network that has minimized *total transport costs* for the connectionless data traffic can be constructed.

Direct CL service is particularly suitable for delay-tolerant or non-real-time applications, i.e., those not requiring tightly constrained delay and delay variation, such as traditional computer communication applications. From the viewpoint of efficiency, *direct* provision of CL data service is a better choice in a large-scale ATM network. On the other hand, if the bandwidth requirement for a transfer session is sufficiently high, it may be preferable to use a dedicated gateway-to-gateway connection rather than via the CL virtual overlayed network. Since only the ATM layer is used to transport data, all data frames will arrive in sequence and with the lowest delay possible due to more specific routing at the ATM layer. In this case, the use of CLSs in the direct approach not only increases the end-to-end delay due to frame processing at each CLS, but also due to an increased propagation delay, since in most cases, CLSs will not lie on the real shortest path for a gateway-to-gateway connection.

We have considered taking one CLS as the intermediate endpoint for every gateway pair in this chapter. Compared with having more than one intermediate endpoint, it has the advantage of less delay but is defective in *scalability.* In general, a smaller p will result in a higher multiplexing gain if having more than one CLS is not prohibited. However, the smaller p implies a larger total transport

cost. A good compromise can be found, in addition to introducing CLS cost under a certain budget consideration, by unifying the cost metric for both the transport cost and multiplexing gain. (The evaluation of statistical multiplexing gain for inter-LAN/MAN CL data traffic is beyond the scope of this chapter.)

One may argue that, first, why the objective function of problem 1 is not subject to the constraint on link capacity; and second, while the connectionless flow is additive, the bandwidth required to carry the flow does not grow linearly with flow. However, connectionless data traffic tends to be bursty by nature; thus it is difficult to predict the accurate values of traffic parameters (e.g., peak rate and burstiness) on the vast majority of existing data networking applications. To optimize the link utilization, several control methods (e.g., rate-based and credit-based mechanisms) for two new QoS service classes in the ATM network, say ABR (*available bit rate)* and UBR (*unspecified bit rate*), for carrying bursty and unpredictable traffic in a best-effort manner are proposed [33]. No resource reservation is performed before transmission. They are supposed to make an efficient use of bandwidth, by dynamically allocating the available bandwidth on an *as need* basis to all the active connections, with extremely low cell loss. It is remarkable that best-effort services do not reserve guaranteed nonpreemptable bandwidth. Thus, the effective gain of statistical multiplexing is almost intractable. It is our belief that it makes sense to neglect the effect of statistical multiplexing on VP-connecting CLSs while we are trying to find the optimal location of CLSs in our formulation.

REFERENCES

1. CCITT Recommendation I.211, B-ISDN service aspects, CCITT SGXVIII, TD42, 1992.
2. CCITT Recommendation I.364, Support of broadband connectionless data service on B-ISDN, CCITT SGXVIII, TD58, 1992.
3. Iisaku, S., and Ishikura, M. ATM network architecture for supporting the connectionless service. In *IEEE INFOCOM,* pp. 796–802, 1990.
4. Ujihashi, Y., Shikama, T., Watanabe, K., and Aoyama S. An architecture for connectionless data service in B-ISDN. In *IEEE GLOBECOM,* pp. 1739–1743, 1990.
5. Kawasaki, T., Kuroyanagi, S., Takechi, R., and Hajikano, K. A study on high speed data communication system using ATM. In *IEEE GLOBECOM,* pp. 2105–2109, 1991.
6. Fioretto, G., Demaria, T., Vaglio, R., Forcina, A., and Moro, T. Connectionless service handling within B-ISDN. In *IEEE GLOBECOM,* pp. 217–222, 1991.
7. Landegem, T. V., and Peschi R. Managing a connectionless virtual overlay network on top of an ATM Network. In *IEEE ICC,* pp. 988–992, 1991.
8. Box, D. F., Hong, D. P., and Suda, T. Architecture and design of connectionless data service for a public ATM network. In *IEEE INFOCOM,* pp. 722–731, 1993.
9. Cherbonnier, J., Boudec, J.-Y. L., and Truong, H. L. ATM direct connectionless service. In *ICC,* pp. 1859–1863, 1993.
10. Sistla, S., Materna, B., Petruk, M., and Gort, J. E. Bandwidth management for connectionless services in a B-ISDN environment. In *ITS,* pp. 21.1.1–21.1.6, 1990.
11. Lai, W. S. Broadband ISDN bandwidth efficiency in the support of ISO connectionless network service. In *ICC,* p. 31.4.1, 1991.
12. Gerla, M., Tai, T. Y., and Gallassi, G. Internetting LAN's and MAN's to B-ISDN's for connectionless traffic support. *IEEE J. Selected Area Commun.* 11(8): 1145–1159, 1992.
13. Kariv, O., and Hakimi, S. L. An algorithmic approach to network location problems. Part 2: The p-medians. *SIAM J. Appl. Math.* 37(3): 539–560, 1979.
14. Kariv, O., and Hakimi S. L. Congestion control for connectionless network via alternative routing. In *GLOBECOM,* pp. 11.3.1–11.3.8, 1991.

15. Krishnan, R., and Silvester J. A. An approach to path-splitting in multiplepath networks. In *ICC*, pp. 1353–1357, 1993.
16. Schmidt, A., and Campbell R. Internet protocol traffic analysis with applications for ATM switch design. *Comput. Commun. Rev.* 39–52, Apr. 1993.
17. Yamamoto, M., Hirata, T., Ohta, C., Tode H., Okada, H., and Tezuka, Y. Traffic control scheme for interconnection of FDDI networks through ATM network. In *IEEE INFOCOM*, pp. 411–420, 1993.
18. Venieris, I. S., Angelopoulos, J. D., and Stassinopolous, G. I. Efficient use of protocol stacks for LAN/MAN-ATM interworking. *IEEE J. Selected Area Commun.* **11**(8): 1160–1171, 1992.
19. Newman, P. ATM local area networks. *IEEE Commun. Mag.* 86–98, Mar. 1994.
20. Vickers, B. J., and Suda, T. Connectionless service for public ATM networks. *IEEE Commun. Mag.* 34–42, Aug. 1994.
21. Yamazaki, K., Wakahara, Y., and Ikeda, Y. Networks and switching for B-ISDN connectionless communications. *IEICE Trans. Commun.* **E76-B**(3): 229–235, 1993.
22. Corcetti, P., Fratta, L., Gerla, M., Marsiglia, M. A., and Romanò, D. Interconnection of LAN/MANs through SMDS on top of an ATM network. *Comput. Commun.* **16**(2): 83–92, 1993.
23. Boudec, J. L., Meier, A., Oechsle, R., and Truong, H. L. Connectionless data service in an ATM-based customer premises network. *Comput. Networks ISDN Systems* **26**: 1409–1424, 1994.
24. Schödl, W., Briem, U., Kröner, H., and Theimer, T. Bandwidth allocation mechanism for LAN/MAN internetworking with an ATM network. *Comput. Commun.* **16**(2): 93–99, 1993.
25. Monteiro, J. A. S., Fotedar, S., and Gerla M. Bandwidth allocation, traffic control and topology design in ATM overlay networks. In *IEEE GLOBECOM*, pp. 1404–1408, 1994.
26. Stavrov, B., de Jogh, J., van Lomwel, A., in't Velt, R. *Bnet*: Designing a connectionless broadband data service in an ATM network by simulated annealing. In *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'95)*, pp. 253–256, 1995.
27. Garey, M. R., and Johnson, D. S. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.* **32**(4): June 1977.
28. Katoh, M., Mukai, H., Kawasaki, T., Soumiya, T., Hajikano, K., and Murakami, K. A network architecture for ATM-based connectionless data service. *IEICE Trans. Commun.* **E76-B**(3): 237–247, 1993.
29. Corcetti, P., Fratta, L., Gerla, M., and Marsiglia, M. A. SMDS multicast support in ATM networks. *Comput. Networks ISDN Systems* 27: 117–132, 1994.
30. Floyd, R. W. Algorithm 97 (SHORTEST PATH). *Commun. ACM* 5(6): 345, 1962.
31. Rose, C. Mean internodal distance in regular and random multihop Networks. *IEEE Trans. Commun.* **40**(8): 1310–1318, 1992.
32. Fotedar, S., Gerla, M., Crocetti, P., and Fratta, L. ATM virtual private networks. *Commun. ACM* 38(2): 101–109, 1995.
33. ATM Forum, Traffic Management Specification, ver. 4.0, 1995.
34. Eom, D. S., Murata, M., and Miyahra, H. Performance comparisons of approaches for providing connectionless service over ATM networks. *IEICE Trans. Commun.* **E80-B**(10): 1454–1465, 1997.
35. Tsai, I.-F., and Jan, R.-H. Internetting connectionless data networks with a wide area public ATM network. *Comput. Networks ISDN Systems* **29**(7): 797–810, 1997.
36. Omundsen, D. S., Kaye, R., and Mahmoud, S. A. A pipelined, multiprocessor architecture for a connectionless server for broadband ISDN. *IEEE/ACM Trans. Networking* 2(20): 181–192, 1994.
37. Ushijima, S., Ichikawa, H., and Noritake, K. Scalable Internet backbone using multi gigabit ATM-based connectionless switching. *IEICE Trans. Commun.* **E81-B**(2): 324–332, 1998.
38. Boiocchi, G., Crocetti, P., Fratta, L., Gerla, M., and Marsiglia, M. A. ATM connectionless server: Performance evaluation. In *Modelling and Performance Evaluation of ATM Technology (Perros, H., Pujolle, G. and Takahashi, Y., Eds.)*, pp. 185–195. *Elsevier (North-Holland)*, Amsterdam, 1993.
39. Romanow, A., and Floyd, S. Dynamics of TCP traffic over ATM networks. *IEEE J. Selected Areas Commun.* 13(4): 633–641, 1995.

21

INTEGRATING DATABASES, DATA COMMUNICATION, AND ARTIFICIAL INTELLIGENCE FOR APPLICATIONS IN SYSTEMS MONITORING AND SAFETY PROBLEMS

PAOLO SALVANESCHI
MARCO LAZZARI

ISMES, via Pastrengo 9, 24068 Seriate BG, Italy

I. SETTING THE SCENE

Ten years ago our group started applying AI techniques to improve the ability of automatic systems to provide engineering interpretations of data streams

Database and Data Communication Network Systems, Vol. 3

coming from monitored structures, in order to support safety management procedures of such structures.

In the following years several applications were developed and are now in service. They range from monitoring of dams to monuments, and finally the same set of concepts and software components was applied to landslides. During the various developments, we moved, with some success, from applied research to the delivery of industrial solutions for our clients.

This led us to acquire knowledge and experience and to create a set of software components, which can be adapted to develop a particular solution for a specific requirement.

The aim of this paper is to provide an overview of the problem of safety management via data interpretation and to show how the integration of databases, data communication, and artificial intelligence may help solve this problem.

For this purpose we shall describe the existing set of software components we have developed, recall their conceptual structure and behavior,[1] and give quantitative data and qualitative comments about the results gained until now.

A. Engineering Software

Since its early years, computer science has been offering to structural engineering powerful tools for performing complex and heavy numerical processing. This main stream of engineering software development generated sophisticated codes based on the advances of numerical analysis, such as those for finite element modeling.

On the other hand, the integration of other computer science disciplines has fostered the studies and the developments of clever software tools for facing structural engineering problems. In this field a key role has been played by artificial intelligence (AI). Artificial intelligence technology was presented to the international structural community in the early 1980s and experienced a significant growth of applications in the latter part of the same decade.

From the broad spectrum of the AI subfields, several applications to civil engineering belong to the so-called *expert systems,* also known as *knowledge-based systems.* They are software systems that include a *corpus* of expert knowledge and processing abilities to deal with data in a way that can be compared with that of experts of the domain. Therefore, their development and success is strictly related to the availability of theoretical and practical experience on the application field, and to its formalization toward its embodiment into a software system. This process emerges from a joint effort of field experts and AI people to identify and correctly exploit the knowledge necessary for solving the problem to be faced by the expert system.

From the point of view of the applications, they belong to two main threads, that is, *design* and *diagnosis,* and the latter is the most relevant with reference to safety problems.

[1] We do not mean to deal here with the details of our interpretation systems, but we provide references of other publications more concerned with modeling techniques and implementation issues.

In the meanwhile, a parallel evolution affected the database technology, which produced robust database management systems based on the relational theory, which can be used both as simple archives to store and retrieve data and as sources of information to be interfaced by automatic agents able to process these data. We shall see that integrating databases and processing tools, such as expert systems, enhances the capabilities of safety procedures.

Eventually, as a result of the incredible explosion of communication technologies in the 1990s, we could exploit tools to link easily and efficiently remote acquisition systems and data processors, connect users over geographic networks, and distribute and share knowledge among all those responsible for safety management.

In this paper we shall present several *real-world* applications that can be regarded as exemplar successful instances of the integration of artificial intelligence, data communication, and databases for managing safety problems related to structural engineering and applicable to other safety domains.

B. Safety Management

This chapter derives from a project that aims to improve the capabilities of an existing information system supporting the management of *dam safety*. The improvement achieved in that original field has been then *exported* to other fields of safety management.

The management of dam safety can be considered as an arduous task both for the need to guarantee the safety of such structures and because of the value of hydroelectric energy.

Significant resources are expended in this field; in Italy the collection, storage, and analysis of information concerning dams as well as the development of methodologies and control systems is considered a critical part of managing safety. Safety management is intended as a process of applying quality procedures designed to test for hazard through design, construction, and operation and the subsequent decision making. The quality procedures require the collection and testing of information concerning the dam against normative or reference models. The application of these procedures is viewed as a method of hazard auditing where the precursors to failure that can accumulate through a project can be identified and dealt with.

Automatic instrumentation and data acquisition systems are used to monitor the real-time behavior of dams. The output of monitoring systems is presented locally to dam wardens to alert them to possible dangerous situations. Telemetry systems are used to send the information to a central database where experts evaluate the status of the structure through the interpretation of data.

The difficulty associated with this data interpretation is due to different factors, such as the large amount of data, the uncertainty and incompleteness of information, the need for engineering judgement, knowledge of the particular structure, experience of the behavior of dams, and a background of general engineering knowledge.

The safety evaluation of dams cannot be based on numerical analysis of data alone, since many physical phenomena cannot be adequately modeled at present. ICOLD stated that "the phenomena which start long before total

failure [...] are still extremely difficult or inaccessible for analytical modeling" and "in most cases, computation of the overall probability of failure of a given dam would require so many assumptions, affected by such a high degree of uncertainty, that the final figure would not be of any practical value for project design and a judgement of its safety" [1]. This implies that a safety assessment must fuse numerical analysis, probabilistic assessment, and engineering judgement, based on experience.

AI concepts and technologies can assist engineers in safety management by providing additional components to the existing information system such as real-time interpretation systems linked to the data acquisition units and intelligent databases supporting the off-line analysis [2–4].

It is important to consider the *uncertainty* involved in any decision about technological systems; these problems may be regarded as cases of *incompleteness,* arising from the nature of the systems to be assessed, since there is no guarantee that all possible behaviors can be identified. This incompleteness is related to the distinction between *open-world* and *closed-world* assumptions. In civil engineering systems, the closed-world assumption that all possible behaviors can be identified and modeled is useful in only very restricted domains.

It is important to acknowledge that *unforeseen events* may occur. In the case of the dam safety, for example, often the worst problem at a dam may not be the dam itself but a lack of knowledge thereof [5]. The literature is rich in examples of failures resulting from unexpected combinations of causes, some of which had not been previously envisaged. The failure of the Malpasset dam was an exemplary case, where an unforeseen combination of the uplift pressure on the dam, due to the reservoir, and some geological structures in the foundation generated a collapse and subsequent flood, devastating the downstream valley and killing about 400 people.

The more a structure is investigated, the more likely that a potential or actual problem will be discovered.

In general, the need for monitoring systems increases with the degree of uncertainty associated with properties of materials and structures, and with the importance and potential risk of the system to be controlled. However, an abundance of data coming from monitoring systems, inspections, and tests raises its own problems in terms of evaluation, interpretation, and management. The first problem regards the availability and cost of the expertise required for the interpretation of the data. Data are only as effective as the designer's or owner's ability to interpret them and respond accordingly. The second problem arises from the volume of data, which can be too large to be easily handled.

Knowledge-based systems (KBS) may assist with such problems in the general context of decision support for safety management.

C. Technical Environments for Safety Management

Figure 1 shows a typical socio-technical environment for safety management, based on our experience on dam safety management: a data-acquisition system can get data from monitoring instruments installed on a dam; it communicates with an interpretation system which may provide on-line evaluation of the data, and with a local or remote archive for storing the data; these archives may be

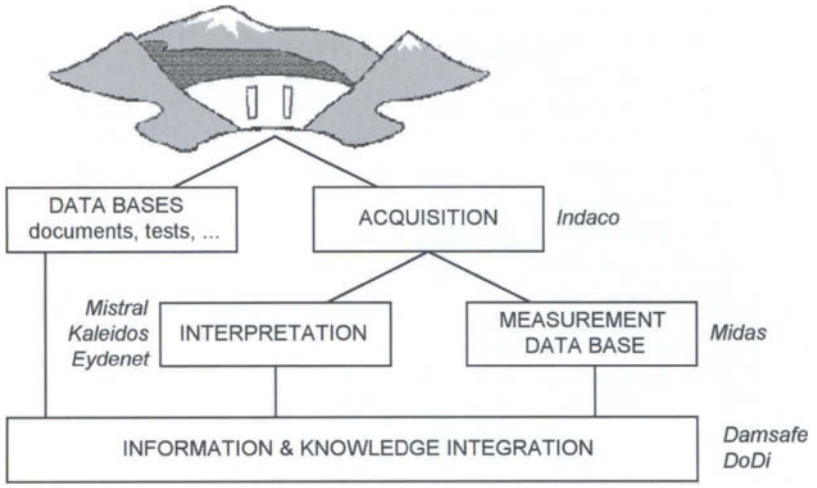


FIGURE 1 Data flow through a safety management system for dams.

accessed by several programs, which can be both conventional data processors and specialized *intelligent* agents; finally, data can be accessed by remote users via the Internet.

In the following, we shall describe some *real* examples of the systems and programs introduced above, and show how their cooperative action may help manage safety procedures. The objects to be discussed are the following:

1. INDACO: a data-acquisition software that gets data from monitoring instruments, validates and stores them locally, and provides communication facilities to transfer data;
2. Mistral, Kaleidos, Eydenet: a family of software tools for supporting data interpretation, based on artificial intelligence techniques;
3. Midas: a database management system for storing large data sets and processing them for off-line interpretation and tuning of interpretation models;
4. Damsafe, DoDi: client–server applications for providing remote users with data, documents, and intelligent support and for promoting cooperative work among users, who share data and knowledge via intranet/Internet.

The monitoring system (INDACO) collects data such as displacements of the dam, basin level, seepage losses, and uplift pressures and tests them against thresholds or values predicted by theoretical models (on-line checks). The resulting warnings and associated data are presented to the people working at the first level of the organization (e.g., the warden of the dam).

The data collected by the monitoring system are sent through a telemetry system to a central computer and are loaded, together with manually collected data, into a database (MIDAS), which allows the subsequent analysis and interpretation of the behavior of the structure (off-line check).

Midas has been operational since 1985 and is used by several organizations in Italy and abroad to manage the data of more than 200 dams. Midas runs in

UNIX, VMS, and DOS environments. INDACO, a PC-based system, has been installed at 35 sites for dam monitoring.

The coupling of these systems has limits, which belong to two different levels, corresponding to two different kinds of users' requirements for improvement, that we have faced by exploiting artificial intelligence techniques: local level (management of warnings) and central level (periodical safety evaluation or deep analysis on demand).

At the local level, monitoring systems currently available enable two kinds of checks on the values gathered by each instrument:

- comparison of the measured quantity and of its rate of change with preset threshold values;
- comparison of the measured quantity with the corresponding value computed by a reference model.

Therefore, these checks deal neither with more than one instrument at a time nor with more than one reading at a time for each instrument. In addition, any behavior (either of the structure or of the instruments) not consistent with the reference model generates a warning message. Because of the limited interpretation skills of the on-site personnel, false alarms cannot be identified and, hence, they require expert attention.

At this level, AI may contribute in collecting the expert knowledge related to data interpretation and delivering it through a system linked with the existing monitoring system. The system can evaluate the measurements and classify and filter the anomalies by using different types of knowledge (e.g., geometrical and physical relationships). It can take into account the whole set of measurements to identify the state of the dam and to explain it. This enables on-line performance of part of the expert interpretation, a performance that reduces the requests for expert intervention and the associated costs and delays, and increases the reliance upon the safety of the dam.

At the central level, the existing system contains quantitative data coming from the monitoring systems. Nevertheless, the safety evaluation requires also the availability of additional types of information (qualitative data coming from visual inspections, results of experimental tests, design drawings, old interpretations) and different types of knowledge (empirical relationships, numerical models, qualitative models), related to different areas of expertise (structural behavior, hydrology, geology).

As a consequence, a supporting system is needed to help people manage such heterogeneous data and the complexity of their evaluation. We have therefore exploited artificial intelligence techniques that can provide new ways of modeling the behaviors of the physical systems (e.g., a qualitative causal net of possible physical processes). This modeling approach is a useful way to integrate different types of knowledge by providing a global scenario to understand the physical system behavior. Moreover AI may be helpful in collecting and formalizing different types of knowledge from different experts for data interpretation and evaluation of the dam state.

To improve the capabilities of the information system in accordance with the above-stated requirements, we developed two systems using AI concepts and technologies.

The first system, Mistral, is linked to the existing monitoring system and provides on-line interpretation and explanation of the dam state. From Mistral, which was originally developed for managing dam safety, we have derived Kaleidos, for the safety assessment of monumental structures, and Eydenet, which is applied to the interpretation of monitoring data coming from landslides.

The second system, Damsafe, is essentially the evolution of MIDAS. It integrates MIDAS (database and graphical and computational tools) and adds new types of information, such as design records, photographs, design drawings, test and monitoring data, and qualitative assessments. Damsafe provides additional models of the dam system and tools to support the interpretation of data and the evaluation of possible scenarios. From Damsafe, a system designed before the *explosion* of the Internet, we have derived DoDi, which exploits Internet to distribute knowledge among a community of users.

II. DATA ACQUISITION AND COMMUNICATION

INDACO is a data-acquisition system for the acquisition of data for the monitoring of continuously evolving situations and the subsequent safety management procedures [6]. It was initially designed for applying it to structural engineering monitoring (dams, monuments, foundations), and then applied also to other fields of safety management, such as monitoring of alpine landslides, river basin flood forecasting, and hydrological and meteorological monitoring.

This acquisition system can get data from several different acquisition units through different data communication interfaces, connecting up to 1,000 remote stations and managing up to 10,000 data-acquisition channels. Moreover, the modular architecture of INDACO allows setting up multilayer master–slave configurations, made of several monitoring systems.

The acquisition process is followed by data validation and processing, and on-line check of data against thresholds. The first step is called *instrumental validation:* it consists of checking the correct operation of the acquisition unit, and verifying that the measurements are within the limits of the full scale of the instruments and of the physical range of the observed quantities.

Then data are checked against *tolerance bands:* the measurements must respect the maximum allowed shift compared to a reference function characteristic of the system. The reference function, and therefore the tolerance bands, may have a periodical or constant trend, according to the type of quantity that is checked. For instance, temperatures may have seasonal trends, modeled by a proper mathematical function; this function may be updated each time new measures are stored into the database.

Finally, INDACO checks the *rate of change* of the measures: given a time base depending on the kind of instrument that provides the measure, INDACO checks the variation of the measure against a set of thresholds. This check allows evaluating the reliability of the measurement, filtering those values that, although numerically correct, seem inconsistent with the physical reality of the measured quantity and therefore are probably altered by occassional noises.

Users can be warned of alarm situations via the display of INDACO, and may draw diagrams of the quantities acquired, as well as of their time trends (yearly, monthly, or daily).

INDACO runs on personal computers, and several secondary storage devices, from removable devices to LAN servers, can be connected to it to store data. Moreover, it can provide data via a standard telephone line, wide area network, or satellite or radio link. In such a way, the acquisition system can be integrated with other data processing systems, which exploit INDACO as a continuous data source.

INDACO has been installed on several dozens of sites, where it provides its support in several different situations. First of all, it is used as a real-time alarm system, for early detection of anomalous phenomena: as soon as an instrument reads an anomalous value, users are informed by the acquisition system and can react appropriately.

On the other hand, its archive is used to provide data for off-line analysis and support of expert interpretation of the physical processes going on. Eventually, the availability of communication facilities allows the coordinated management of distributed data from a remote control center: in such a way, data collected by instruments located on a wide area may be transferred to a unique *master station,* where they can be jointly used by safety managers.

III. ADDING INTELLIGENCE TO MONITORING

We have discussed above the problems arising from the limits of real-time acquisition systems, and we have introduced the requirements that led to the development of Mistral, a decision support system aimed at interpreting data coming from acquisition systems of dams. From that original system, we have derived similar tools for monuments (Kaleidos) and landslides (Eydenet).

In fact, the Mistral system is packaged in a way that enables us to deliver *solutions* for specific problems. A so-called solution for specific business needs is composed of:

- a context of application providing a set of requirements;
- a collection of software components that can be used as building blocks to be adapted and integrated into a software product;
- a technological environment (hardware and system software environment, common software architecture, standard communication mechanisms) that enables the integration of the existing components; and
- the ability to deliver adaptation and integration services based on a specific know-how both of software technology and engineering solutions.

The context of application is the management of the safety of structures. Data about structural behavior are collected through tests and visual inspections, while automatic instrumentation and data-acquisition systems are used for real-time monitoring. The interpretation of such data is not easy owing to different factors, such as the large amount of data, the uncertainty and incompleteness of information, the need for engineering judgement, knowledge of the

particular structure, experience of the behavior of structures in general, and a background of general engineering knowledge.

According to this context, Mistral-based solutions aim at helping safety managers, engineers, and authorities to deal with safety problems of structures in the following two ways:

- by filtering false alarms; and
- by supporting the automatic detection and early warning of dangerous situations.

Both aspects are valuable. The former is more related to the reduction of costs associated with human interpretation required even in the case of false alarms, while the latter deals with the improvement of the safety level of the structure.

From an organizational point of view, two situations for the use of such a kind of system may be interesting.

The first one is the interpretation of data coming from structures of particular relevance or criticality, such as a large structure or a structure that manifested some problem.

The second is the case where a significant amount of structures (e.g., a set of dams of a region) are operated by a central office that collects all the data and monitors the status.

Specific requirements are satisfied by adapting and integrating software components taken from an existing set. They need to be tailored in the sense both that not all the components may be of interest (e.g., a specific interpretation technique is chosen) and that the chosen components must be adapted (e.g., to the configuration of the existing sensors).

In the following we provide a list of the components we have already developed and are suitable for use for any new instance of Mistral:

- *Communication with a data-acquisition system.* This component manages the data communication from the data-acquisition system to Mistral; calls the monitoring system and receives the data gathered during the last acquisition (normal real-time procedure) or collected while Mistral was, for some reason, not active. It can work via serial connection or over a network. Moreover, Mistral and the acquisition system may be hosted by the same computer, provided that it can support multiprocessing.
- *Database of measurements and sensors and associated management functions.* Three kinds of database of measurements and interpretations are available: a static database of test cases, a dynamic database collecting all the data related to the monitoring system (measurements, evaluations, explanations), and an archive of past *reference* situations, used by case-based reasoning procedures (explained in the following).
- *Numerical preprocessing.* The numerical preprocessor checks the measurements and their rate of change against thresholds; moreover, it computes the expected values for (some of) the instruments by means of theoretical models and checks against thresholds the displacements of the measured values from the expected ones and their rate of change. Note that the whole set of thresholds used within Mistral has a different meaning from that used by INDACO when the two system are coupled: while the former are safety thresholds, individuated

by safety experts on the grounds of their knowledge of the structure and of past measurements, the latter are purely regarded as instrumental thresholds, used to verify the correct behavior of the instruments.

• *Empirical and model-based interpretation.* Essentially, the interpretation is a process of evidential reasoning, which transforms data states into states of the physical system and interprets them in terms of alarm states. The interpretation may be based on three types of knowledge [7]:

1. *Empirical knowledge.* States of sensors (derived from comparisons between measurements and thresholds) may be combined through sets of rules. The rules may be applied to groups of instruments of the same type providing a global index having the meaning of an anxiety status based on empirical past experiences. Rules may be also applied to sensors of different type belonging to the same part of the physical system (e.g., a dam block). This may suggest the existence of dangerous phenomena in that part of the physical system.

2. *Qualitative models.* Qualitative relationships between measured variables may be evaluated to provide evidences for physical processes (e.g., excessive rotation of a dam block)

3. *Quantitative models.* Quantitative models may be used, in cooperation with thresholds, to provide states of sensors (e.g., relations between cause and effect variables based on statistical processing of past data or deterministic modeling approaches).

• *Case-based interpretation.* Analogical reasoning is used to retrieve, given the qualitative description of the state of a structure, the closest-matching cases stored in a case base, which can help safety managers to interpret the current situation. Usually, situations stored in the reference database are those more deeply investigated by experts, because of some singularities, and are enriched with experts' comments; therefore, at run-time, the analogical reasoning makes it possible to understand whether the current situation is similar to a past one, and the comments of the latter can address the investigation of the current one. This makes it possible to record and manage special behaviors that cannot be adequately managed by explicit models.

Two different approaches were adopted for developing these tools. The first approach is based on numerical/symbolic processing: several metrics were defined to compute the analogy of a couple of cases. These metrics span from simple norms within the hyperspace defined by the state indexes, to more complex functions also taking into account the gravity of the cases, in a fashion similar to the law of gravitation. In such a way, instead of computing the distance between two situations, their reciprocal attraction is evaluated. The similarity between the situations is then defined by checking their attraction against an adequate threshold system. This approach led to the development of a tool (DéjàViewer) which encompasses these different metrics. The second approach is based on the application of neural networks to implement a sort of associative memory that makes it possible to compute the similarity of a couple of cases [8].

• *Empirical interpretation based on neural networks.* Within Mistral, deep knowledge is usually codified by numerical algorithms, and qualitative reasoning is implemented by symbolic processing; when heuristic knowledge is involved, the shallow knowledge about the structure and the instrumentation is managed through empirical rules based on the alarm state of single

instruments, taking into account their reliability and significance. Such rules are derived from the analysis of a set of exemplary cases, which make it possible to identify weights to be given to the parameters used by the rules.

This process, time consuming and boring for the experts, can be adequately dealt with neural nets. Therefore, neural nets for performing the empirical evaluation of data in order to achieve the same results as the symbolic processors previously used, but with reduced development and tuning effort, can be developed [9].

- *Explanation.* The result of the interpretation is the identification of the current state of the structure. From the trace of execution, using knowledge about the behavior of the monitored structure and the instruments, the explanation module generates natural language messages. They describe the current state of the structure and the deductions of the system.
- *Statistical processing.* Statistical analysis of data may be performed both to provide evidences for the evaluation process, in order to spot trends in data (the measure of a plumb line is constantly increasing), and for periodical synthesis of past evaluations (weekly reports).
- *Charting.* It is possible to select a situation from the database and show on the screen its graphic representation.
- *Reporting.* Reports may be extracted from the database, providing the required detail for various types of users (e.g., safety managers or local authorities).
- *GIS-based interaction.* A window of the system may interface a GIS, to exploit its cartographic facilities and offer a powerful representation tool for *navigating* through data and exploiting standard features of GISs, such as zooming functions to highlight subareas to be analyzed. This may be useful when the monitoring instruments are spread over a large area, as with environmental monitoring (we have exploited a GIS for monitoring landslides).
- *Man/machine interface.* This window-based interface draws on the screen graphical representations of the objects that have been assessed (e.g., the monument and its structural components) and displays them using a color scale based on the object's state, by mapping the indexes belonging to the scale normal–very high anomaly into colors ranging from green to red, using gray for representing missing information (for instance, malfunctioning instruments).

Figure 2 shows how Mistral gives the user information about its interpretation. Small squared lights atop the screen codify the state of the instruments. These squares lie on colored strips, whose colors represent the state of families of instruments of the same type, which are interpreted as the activation states of relevant phenomena, such as movements with reference to plumb lines, deformations with reference to extensometers, and seepage with reference to piezometers.

Rectangular lights in the lower part of the screen codify the processes' activation state, such as rotation or translation of a block. The colors of the dam blocks (right shoulder, block 8/10, etc.) and of the strip at the bottom of the screen (global state) summarize the state of the physical system's structural components and the state of the whole structure.

Interactors are available to get more refined information about the dam state, by focusing on interesting details. For instance, when a user presses the gray button "Block 1/2" in Fig. 2, the window shown in Fig. 3 appears on the

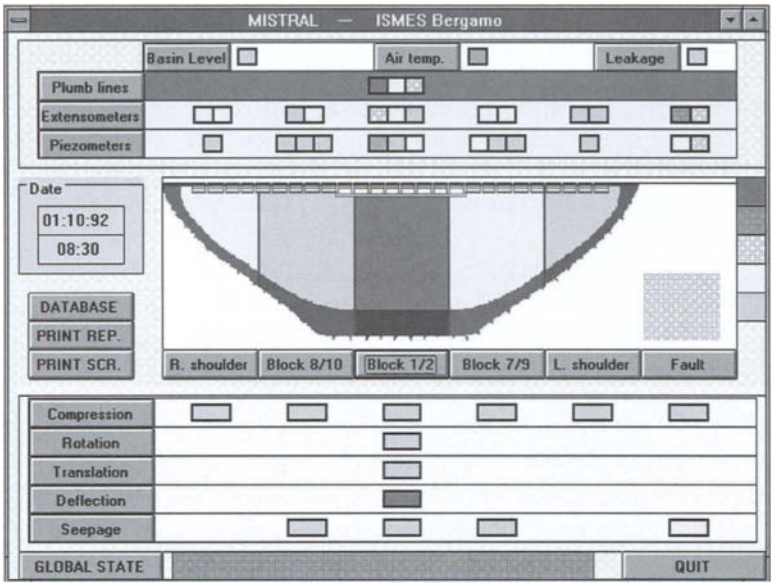


FIGURE 2 Mistral, first release: assessing the state of a dam (simulated data).

screen: this window presents the current readings recorded by the instruments on the block 1/2 (main section of the dam), a picture of the block, the current state of the instruments (on the right) and of the whole block (on the left), and an explanation of the current state.

Via the interface, the user can also activate functions, such as print screen, and access the internal data bases.

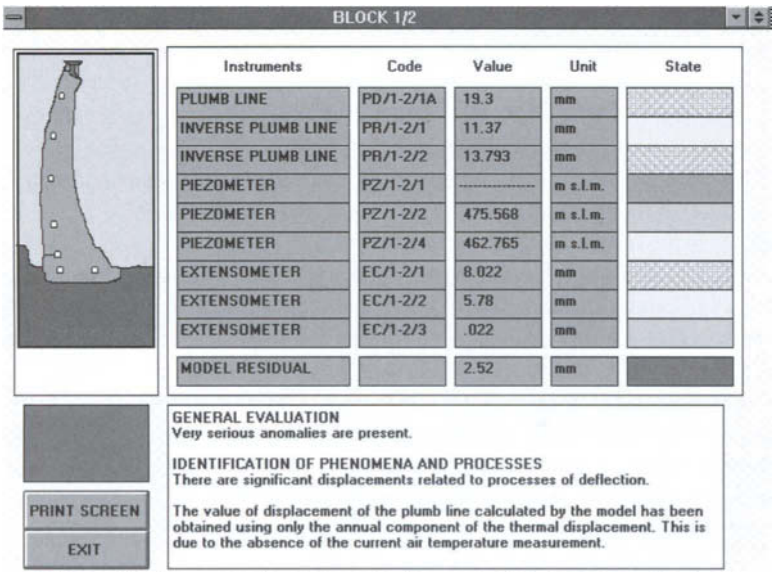


FIGURE 3 Mistral: state of a dam section (simulated data).

The existing set of components may be used as a basis for integrating them into a product only if they are based on a common technological and architectural environment. The reference technologies for our projects are PC Windows, relational DBMS (e.g., Access) interfaced through SQL and ODBC, GIS Mapinfo, communication mechanisms based on OLE, interpretation kernel based on Prolog, numerical processing in C, and interface in Visual Basic.

The choice of Prolog makes it possible to clearly separate data and procedures within the code; this feature makes it possible to quickly reconfigure the system, whenever the addition or removal of objects to be evaluated (e.g., new monitoring instruments) or functions (e.g., new evaluation rules) is required. Moreover, Prolog proved very useful for the development of rule-based evaluation submodules and for the generation of natural language explanations.

Using the standard interfaces such as OLE and ODBC, a reference architecture has been established where the components may be plugged in. Essentially the system is composed of three layers communicating through standard mechanisms: the man/machine interface, the layer of components providing types of processing of data, and the data layer.

A. Delivered Solutions

Below we list the solutions that have been delivered upto now. The list includes the name of the structure, the organization in charge, the year of installation, and the number of sensors interpreted by the system.

Site	Owner or manager	Operational from	No. of sensors
Ridracoli dam	Romagna Acque	1992	46
Pavia Cathedral and six towers	Lombardia Regional Authority	1994	120
Pieve di Cadore dam	ENEL	1996	54
Valtellina landslides	Lombardia Regional Authority	1996	250
Cancano dam	AEM	1997	68
Valgrosina dam	AEM	1998	96
Fusino dam	AEM	1998	12

The first version of Mistral has been installed since October 1992 on a PC linked to the monitoring system of the Ridracoli dam and is currently in use. After that, new applications were delivered for the Pieve di Cadore dam (operated by ENEL, the Italian National Electrical Company) and three dams operated by the AEM.

This case is interesting because, while in the previous cases the system is interpreting the data from a single dam and is installed at the warden house near the dam, in this last case the system is installed at a remote control site and is designed to progressively host the interpretation of data coming from many dams. Moreover, the same software is loaded on a portable computer used by inspection people and may be linked to the acquisition system on site to get data on the fly and provide interpretations.

The system obtained very rapid acceptance by the user. It reduced the effort required for the management of warnings, allowed better exploitation of the

power of the monitoring system, and most importantly, improved the quality of the safety management procedures. Users appreciated both the interpretation capabilities of the system and its man/machine interface, which highlights the system's state in a user-friendly and effective way.

At least two classes of users of the system exist:

- *dam managers and dam safety experts:* they use Mistral as a control panel, which shows the current state of the structure, and as a decision support system; the internal dynamic database helps the dam managers understand the evolution of the dam state, in order to start necessary actions (e.g., inspections, retrofitting); and
- *junior safety managers:* they use Mistral as a training tool, to understand past evaluations of the behaviors of a dam.

Another application was delivered for the management of the safety of the Cathedral of Pavia and of six towers in the same town.

On March 17, 1989 the Civic Tower of Pavia collapsed. After this event, the Italian Department of Civil Defence required the installation of an automatic monitoring system with the ability to automatically provide engineering interpretations, linked via radio to a control center, located at the University of Pavia.

The instrumentation installed on the Cathedral and on the towers makes it possible to acquire the most important measurements on each monument, such as the opening/closure of significant cracks, displacements, and stresses, and also cause variables, such as air temperature, solar radiation, and groundwater level (Fig. 4). In the case of anomalies the system calls by phone the local authority and ISMES. The system is installed in Pavia and has been operational since the beginning of 1994 [10].

Finally, the same approach has been applied to the interpretation of data coming from the monitoring of landslides. In the summer of 1987 a ruinous

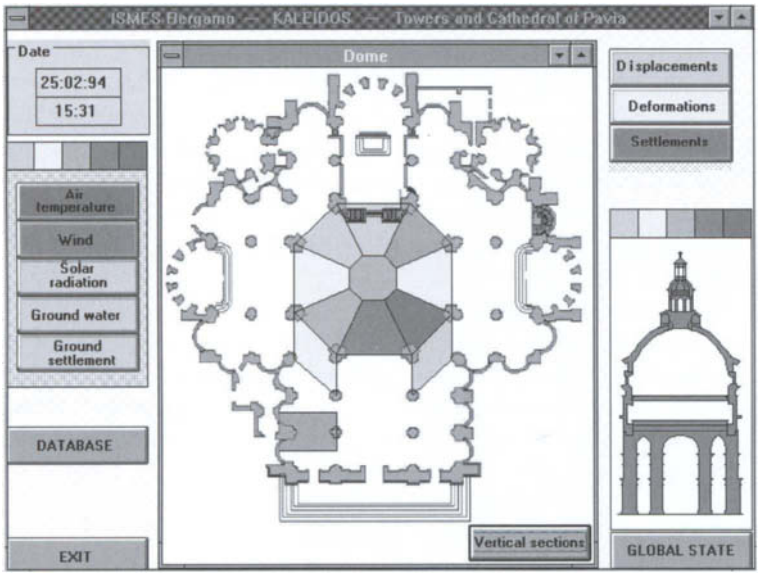


FIGURE 4 Kaleidos: the state of the dome in Pavia (simulated data).

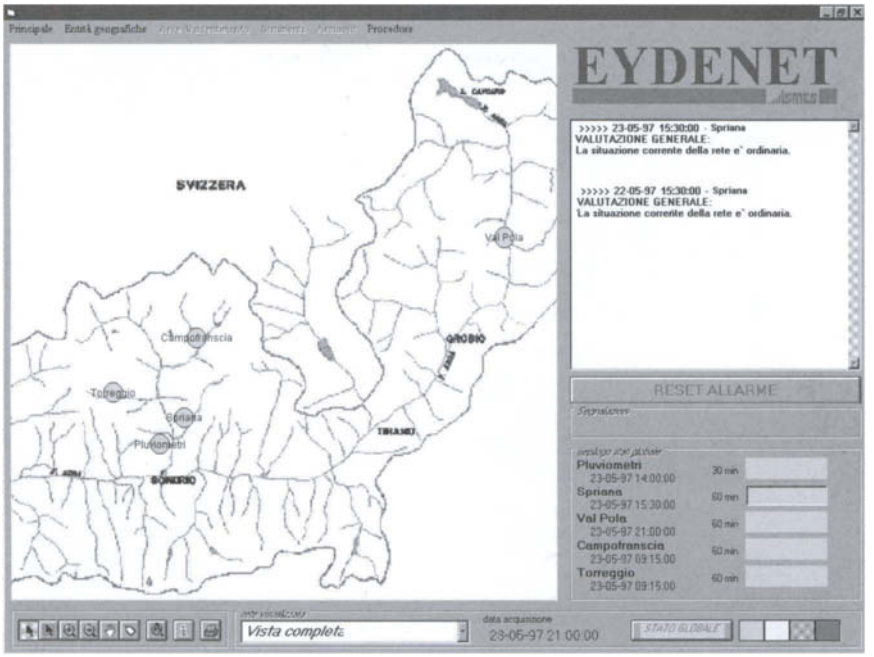


FIGURE 5 Eydenet: the window-based interface.

flood affected large areas of the Valtellina (Northern Italy) and caused many landslides. On July 28 a large mass of rock, estimated to be 34 million m^3, suddenly moved down toward the Adda Valley and destroyed several villages with casualties. As a consequence, the regional authorities and the Italian Department of Civil Protection appointed ISMES to develop a hydrogeological monitoring net (about 1000 sensors).

The net includes devices such as surface strain-gauges, topographical and geodetic benchmarks, inclinometers, settlement meters, cracking gauges, rain gauges, snow gauges, and thermometers (Fig. 5, Fig. 6, Fig. 7).

Currently, the data of the most significant instruments of the net (about 250) are processed by an instance of Mistral (called Eydenet), which supports real-time data interpretation and analysis of data concerning the stability of the slopes affected by the landslides. The system has been operational since October 1996 at the *Monitoring Centre for the Control of Valtellina,* set up by the Regione Lombardia at Mossini, near Sondrio; it is operated by a senior geologist and a team of engineers of the Regione Lombardia [11].

B. Quantitative Results: Report from a Specific Case

While planning the development of Mistral, we have decided to concentrate our efforts on the achievement of three main quality objectives:

1. ease of use of the system,
2. reliability, and
3. filtering capabilities.

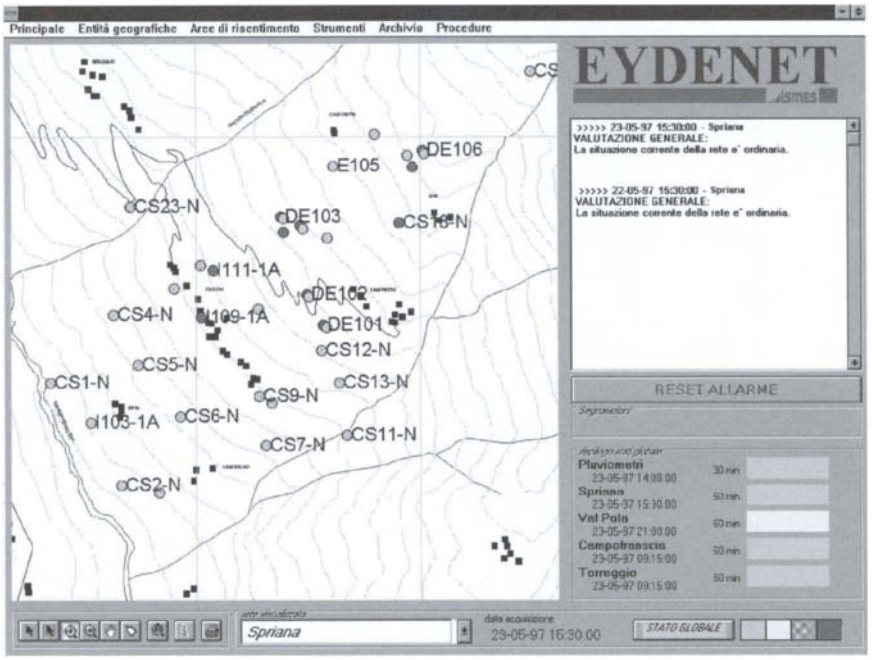


FIGURE 6 Eydenet: assessing the state of a specific area.

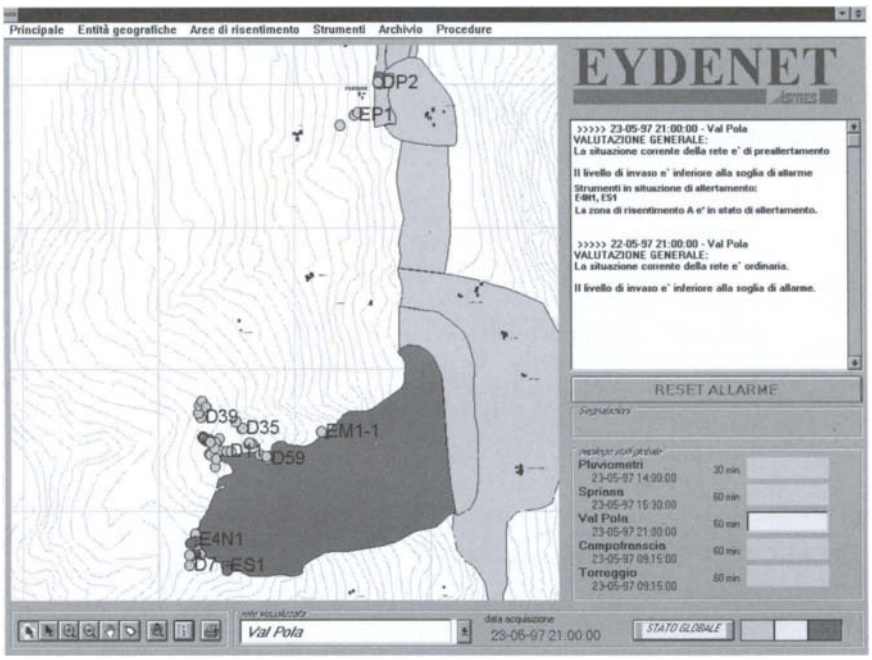


FIGURE 7 Hazard level of zones (simulated data).

With reference to the first aim, we believe that often very sophisticated AI prototypes fail and are not accepted by users because their functionalities are difficult to use. AI researchers tend to consider the *intelligence* of the system their main target, while we think that our target is the intelligence of the system's behavior, and this quality strongly depends, for interactive applications, on the *communication skills* of the system, that is, on its ease of use.

Therefore, we have sometime *sacrificed* clever functions that appeared too hard to be used or, even better, we have delivered different releases of the same application, whose complexity, both of reasoning and of use, was progressively higher, so that the users could get accustomed to the system in its simpler version, before facing more complex functionalities.

As an example, the system developed for the dam at Ridracoli was initially delivered in a version for a restricted number of instruments and with a limited set of interpretation procedures, mainly based on the validation procedures of the acquisition system, which were already well known by the users; we concentrated our attention on the man/machine interaction, and quickly got the system used by the safety managers. Subsequently, we delivered three other releases of the system, which incorporated more sophisticated numerical procedures, evaluation of trends, syntheses of the evaluation results, graphics, and interpretation algorithms on a larger set of instruments, and their improvements were easily accepted by users on the grounds of their previous experience with the system.

With reference to the second topic—that is, the reliability of the system—we are rather satisfied by Mistral's behavior: in six years of continuous operation, from the first installation at Ridracoli to the latest release at our fifth site (an improvement of Eydenet), we had to fix bugs or tune procedures no more than five times, and extraordinary maintenance interventions are required only when accidental crashes of the system, the acquisition devices, or the computer nets occur.

We believe that our development process, based on the ISO 9000 standard, is the main reason for such achievement, since it provides a very strong and clear reference framework for deriving correct code from specifications, and facilitates all those project reviews that highlight possible defects of the production process.

Eventually, the third, essential goal was related to the interpretation and filtering capabilities of the system.

In these years the different *incarnations* of Mistral have evaluated several millions of measurements; both users and our safety experts have analyzed offline the system's behavior, and their qualitative judgement is that Mistral works correctly.

More specifically, it never missed any significant situation, while it never created unjustified alarms; it filtered *correctly* several accidental warnings of the acquisition systems; moreover, it pointed out some situations that were significantly different from what was expected, since the physical structures under evaluation were *operated* in situations violating the assumptions of Mistral's models.

Let us have a look at some figures that describe an interesting sequence of events that were monitored by Mistral at one of our dams. First of all, let us

stress that *nothing dangerous* ever happened in this period, nor do the figures show any failure of the socio-technical apparatus that is in charge of the safety management: our main concern about these figures is related to the fact that Mistral, even when tested with situations that it is not trained to manage, can provide useful suggestions to safety managers.

We have analyzed a set of 684 acquisitions gathered in 1997 by some dozens of instruments, for a total of about 37,000 measurements. This instance of Mistral derives the global state from row measures through empirical and model-based reasoning, taking into account numerical and statistical processing; case-based reasoning is not applied, since we were not requested by our customer to incorporate this functionality. We have evaluated the global state of the structure according to the values shown in the following table:

Level	Global state
1	normal
2	normal—some local warnings
3	normal—some anomalies
4	anomalous—warning
5	very anomalous—alarm

With reference to those values, Mistral produced the following evaluations of the global state of the dam:

Level	No. of situations	% of situations
1	426	62
2	230	34
3	20	3
4	8	1
5	0	0

These figures are very different from those usually produced by Mistral, which are much more concentrated on the first level of the scale.

However, an off-line analysis of the data pointed out that the warnings were due to an extraordinary combination of events (higher level of the basin, abnormal air temperatures), which were outside the limits of Mistral's models. In fact, the result of Mistral's evaluations was, in those situations, a reminder for the dam managers of the unusual, delicate, but not severe operating conditions.

Finally, if we extract from the whole data set only the measurements gathered in the period when the dam was operated under *normal* conditions, that is, according to the limits we considered for tuning Mistral's models, we achieve the following results:

Level	No. of situations	% of situations
1	416	96
2	13	3
3	3	1
4	0	0
5	0	0

A deeper examination of the state indexes highlighted that Mistral classified as normal situations (Level 1) about 200 acquisitions that presented accidental

overthreshold values. According to our safety experts, such *filtering* was proper and correct; as a result, the dam managers were not requested to react to those situations, with a sensible reduction of management efforts.

On the other hand, when Mistral highlighted some unusual situations, the off-line analysis proved that its judgement was correct, and those situations, although not dangerous, were suitable to be carefully taken into consideration.

As a combined result of these behaviors, we have achieved that the dam managers are not distracted by false alarms, and can concentrate their attention and efforts on really significant situations. Moreover, the reduced number of *stimuli* avoids that safety managers discard alarms, since they are too much to be dealt with.

C. Impact on the Organization

We have shown some quantitative data to exemplify the results that may be achieved, adding such types of programs to an organization managing the safety of complex systems.

More generally we can draw the following considerations about the impact on the existing organization:

1. The diagnostic program must fit the existing information systems in terms of existing machines, flows of information, and human roles. A key way for the success is to understand the existing organization and how some requirement may be satisfied by adding a new AI-based component. The fundamental requirement for justifying the cost of the system is the filtering of false alarms. This performance may be quantified and used to support the installation with respect to the management. The advantage is relevant as the number of sensors to be interpreted increases. Obviously the availability of an automatic interpretation device increases the safety level, providing a continuous, distributed, and on-line expertise.

2. An interesting aspect is the collection and (partial) accurate modeling of a corpus of knowledge, which was formerly belonging to a limited number of people. Even in the mind of those experts, that knowledge was not organized in a clear conceptual structure suitable to be communicated. This knowledge is now an explicit property of the company and may be distributed and sold.

3. Finally, a last comment is related to the interaction between the user and the diagnostic program and the issue of responsibility. The system is used on a regular basis, and at the end, the decision to trust in the system or call the responsible people and the experts is in the hands of the users. From this point of view, the transparency of the system and the ability to justify decisions and support them with levels of detailed data are of key relevance. Nevertheless, we cannot forget that confidence is placed in a machine, that is, in its developers. This means that the interpretation system is a critical application, which must be specified, designed, built, and validated carefully. In fact, even if decisions and responsibility lie in the hands of human beings, the elements for decisions are in the cooperative system made by human beings and machines.

IV. A DATABASE FOR OFF-LINE MANAGEMENT OF SAFETY

Midas is a system for managing and processing the data that characterize the behavior of a structure monitored by on-site measurement networks. While on-line system like those presented in the previous chapter store restricted sets of data, necessary for their algorithms, Midas is intended as a historical database, where large sets of data gathered by acquisition systems can be transferred, stored, and then updated, retrieved, and processed quickly and with high reliability to produce, in the format required by supervisory agencies, all the graphic representations suitable for safety evaluation.

Moreover Midas makes it possible to interpret the behavior of the structure through theoretical forecasting models. Midas may manage chronological data collected from any type of structure. The numerical models must be created specifically for the particular type of structure involved.

Midas is organized in a modular structure and many users may access the system simultaneously. A number of structure databases may be managed concurrently.

Midas runs on PCs, Unix, and Vax/Vms workstations.

A. Functions and Data

The acquisition of data is the first step to know the structure's behavior. The measurements are provided by automatic data-acquisition systems that employ electronic apparatuses installed both on site (temperature sensors, electric strain gauges, automatic plumbline readers, etc.) and off site (minicomputers for data collection, recording, and transmission). This technique makes it possible to monitor the structure with more frequent readings.

The Midas system was created to accomplish the following purposes:

- independence of application programs and data;
- possibility of defining complex relations among data;
- no redundancy among data;
- data security;
- ease of communication with the database (the user can formulate links among the various quantities stored);
- possibility of developing new procedures; and
- possibility of centralized data control.

The structure of the database allows storage of the raw automatic or manual readings, which are organized in columns and defined as *source measurements*. They can then be processed automatically by means of formulas predefined by the user, to derive the so-called *computed values*.

The advantages of this method are that the original measurements are preserved unaltered (and available for further processing), while their automatic processing eliminates possible errors resulting from manual operations. A further advantage is that data storage is accomplished rapidly, since the measurements do not need any preliminary manual processing stage.

Preliminary data control (correlation, Fourier analysis, min/max values, smoothed theoretical values, etc.) highlights anomalous values (accidental

errors or systematic errors) due to incorrect processes or hardware malfunctioning.

Several different types of statistical models may be used within Midas to monitor all the effect quantities (displacement, seepage, settlement, etc.), provided that measured values are available. In fact, when adequately long chronological series of both cause quantities (basin level, temperature) and effect quantities are available, Midas can exploit regressive techniques to find correlations among these quantities, and then use them as a model to forecast values for the monitored quantities.

When the link between cause and effect is determined by structural analysis rather than on the basis of the structure's past history, quantities are controlled by a deterministic or a priori model. For instance, these models may be used to interpret and monitor the displacements or rotations of a structure and its foundations.

Eventually, hybrid models may be used as well, which mix statistical and deterministic components.

V. INTEGRATING DATABASES AND AI

We have claimed that, with reference to structural safety, quantitative data coming from the monitoring systems can be profitably integrated with additional types of information (qualitative data coming from visual inspections, results of experimental tests, design drawings, old interpretations) and different types of knowledge (empirical relationships, numerical models, qualitative models), related to different areas of expertise (structural behavior, hydrology, geology).

We have exploited artificial intelligence techniques to achieve such integration, building a first prototype system, called Damsafe; from Damsafe we have subsequently derived DoDi, the Dams Dossier, which has been tailored to the needs of a specific customer.

While the former system did emphasize the contribution of intelligent processing to safety management, the latter is based on a concept that we call *weak information system.*

In this section we shall discuss the concept of a weak information system and then we shall present some details about Damsafe and DoDi.

A. Weak Information Systems for Technical Data Management

The design, construction, and management of large structures and plants produce large amounts of technical information. This amount of information may be stored and distributed through different media: some of it may be paper based while other items may be hosted on different systems in digital form.

This information is used during the whole life cycle of the structure or plant from design to decommissioning. During the life cycle several classes of users need to access and modify the information (e.g., designers, maintenance people,

or safety managers), and require specific views of available information. Each view represents a specific way to access and integrate the data.

For instance, a safety manager of a set of dams may need to access the available information concerning a specific structure to evaluate its safety status (measurements, tests, past engineering evaluations, design documents).

On the other hand, a manager who needs to interface authorities must access the information from the point of view of the bureaucratic process. An information system able to manage such kind of problems must deal also with the following technical requirements:

- sharing the information (between local and central offices and among various partners);
- linking distributed sources of data;
- managing multimedia data; and
- providing technical solutions to deal with etherogeneous environments and supply common man/machine interfaces.

Another fundamental constraint is the development cost, considering that, for operational structures, the problem is to deal with information sources as they do evolve in time. To reduce the cost we also must take into account the following topics:

- reuse of existing data bases;
- reduction of the development from scratch; and
- arrangement for evolutionary paths for the development of the system.

1. Strong vs Weak Information Systems

The existing approaches for dealing with the above-mentioned requirements span from solutions where no real integration exists (the pieces of information are collected, if necessary in digital form, and some indexing mechanism is superimposed) to strong solutions where a unique database or product model is designed to act as a basis for integration.

We say that the second approach leads to *strong information systems,* and we argue that, between the *no integration approach* and the *strong integration* one, we may propose concepts and technology to develop types of information systems in the middle. They may exploit the advantages of a more knowledge-intensive integration than pure indexing, while avoiding the costs and the feasibility problems of the strong approach. We call this type of information system *weak information systems.*

In the following we resume the comparison between strong and weak approaches.

Strong information systems	Weak information systems
Common data model	Multiple specific integration models
Database-oriented architecture	Integration of multiple data sources through added integration layers
One-shot design	Growing process
Redesign of legacy systems	Integration of legacy systems

The architecture of a weak information system is composed of:

- a layer of information sources (databases, computational programs, subsystems); and
- an integration layer.

The integration layer is composed of the following sublayers:

A. abstraction layer (information models);
B. communication layer between models and humans (computer–human interface); and
C. communication layer between models and information sources.

The basic concept of the weak integration system is to add to the layer of the existing information a new layer of multiple partial models of products and processes. Each model may include some sort of specific knowledge and acts as a way to structure and access the information according to a particular user view.

For instance, as we will see in the description of the Dams Dossier, we may access the information concerning the management of a large set of dams in service through the following models:

- a unique object (or a hierarchy of objects linked with part-of relations) models a dam and makes it possible to link the related information (measurements, drawings, images, basic reference data, surveys, past safety evaluations, tests, design documents), which is spread across many geographically distributed databases. This type of model provides services to a technician assessing the safety status of the structure;
- a hierarchy of objects describes the whole set of activities that constitutes the management process of dams in service. Through this hierarchy it is possible to access part of the same information linked to the previous model as well as other information such as laws, standards, and company regulations.

The models act as specialized interfaces that provide help to access and structure the information so that a synthesis of information is not a new packing of it, but a set of links providing specific navigation paths.

2. Implementation through Internet Technology

The implementation of a weak information system may take advantage of the use of the Internet software technologies.

The main reasons for this choice are the following:

- the software components available through the Internet are based on a client–server architecture and are available on various platforms. This makes easy the design of distributed applications integrating different environments;
- the technology is largely based on *de facto* standards that support the stability of the applications, the interoperability, and the ability to evolve;
- the WWW-distributed hypertext technology is a useful tool for developing the integration layer. This allows managing in a straightforward way

hypermedia documents. Moreover standard interfacing mechanisms like CGI make it possible to integrate existing databases and applications;

- the hypertext technology by itself makes it possible to write simple product and process models. With the same technology it is possible to develop standard man/machine interfaces and to link models to existing information bases;
- if more complex models are required, technologies like Java allow the development of object-oriented models embedded within the distributed hypertext;
- finally, many low-cost software components may be exploited to reduce the development cost.

B. Damsafe

Damsafe is intended as a cooperative management tool, in which different types of information concerning a dam and different types of models of the dam system can be united to assist the engineer in carrying out the procedures of dam safety management Fig. 8.

The system enables hazard audits to be carried out on descriptions of the state and behavior of the dam coming from monitoring and from experts' judgement. Moreover, the system interfaces several external databases. The structure of the system is based on three main entities contained within an integration environment:

1. Models of the physical world, which describe both the present state and the desirable or undesirable states of the physical world; they are constructed using object-oriented modeling techniques;
2. Models of human reasoning (reasoning agents), which are models of reasoning about the problem domain, including identification of data features or mapping of data states into dam states;

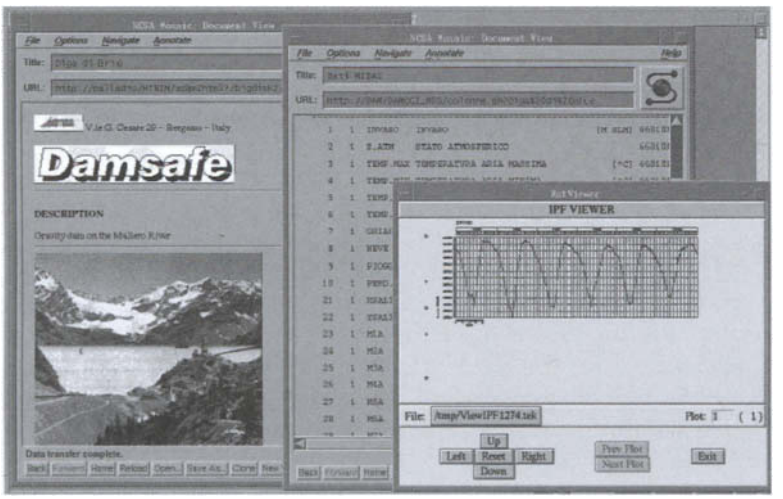


FIGURE 8 Damsafe: integrating databases over the Internet.

3. Communication mechanisms, which take the form of interfacing software components, that enable the user to cooperate with the system through an object-oriented man/machine interface.

The whole system can be used in two different ways:

- *As a diagnostic tool:* there is a sequence of operations of the reasoning agents that allows the translation of data into dam states;
- *As a knowledge integrator:* the system facilitates the integration of information about the dam. Drawings, maps and pictures of the dam form part of the information base.

Several databases are linked to the system: a database of past measurements of the dam, a database of laboratory and *in situ* tests, and an archive of documents and cadastral information. The system functions as an integration tool for different types of knowledge about the dam, such as theory, regulations, and expert knowledge. In this way the system can be seen as a virtual expert that reflects the knowledge of many different experts (civil engineers, hydrologists, geologists, etc.) interviewed during the knowledge-gathering phase.

The structure of Damsafe is based on the object-oriented approach. Different types of knowledge are integrated using a hierarchical model describing the main components of the system. The hierarchical structure includes two physical world models and three reasoning agents. The models make up the problem domain, while the reasoning agents contain the knowledge required to reason about the models. They perform a variety of tasks, the most important being that of relating the concepts in the data world to those in the dam world.

1. Data World and Dam World

The concepts that constitute the data world are those used by engineers in discussing and interpreting the data for dam safety. Some of these concepts are expressed quantitatively, that is, numerically; others are expressed qualitatively. Within this model are the features of data significant for identifying particular behaviors and states of the dam system.

The data world contains several objects; each object represents the data related to a single instrument of the monitoring system. These data are attributes of the object; they can be time series of instrument readings, as well as details of the type of variable represented. Features such as peaks, trends, steps and plateaux, identified in different types of time series, are recorded in this model.

Each object has methods to deal with it (in the object-oriented sense), which allow the user to access the knowledge linked to the object. In this way one can read the values of the attributes of the object, or show a time series on the screen. It is also possible to assign values to attributes; this allows the user to act directly on the data world, bypassing the filtering of the reasoning agents.

The dam world contains a model of the physical world of the dam and its environment, concepts describing the possible states of this world, and a set of concepts modeling its possible behaviors. The physical dam model describes the dam as a hierarchy of objects (e.g., dam body, foundation). These objects have attributes that taken as a set, describe the state of the dam. The attributes

can be quantitative (e.g., reservoir level), qualitative (e.g., concrete quality), or complex (e.g., underpressure profile).

The model of the behaviors of the dam contained within the dam world is represented through a causal net of processes and is described in the following.

2. Causal Net of Processes

The model of the behaviors of the dam and its environment is a set of processes connected in a causal network that models how the behaviors of the dam can interlink in a causal way, resulting in various scenarios as one process leads to another (Fig. 9 shows a fragment of the network).

The full net includes 90 different processes describing possible dam behaviors (e.g., translation, chemical degradation of foundation). We derived this network from published case studies of dam failures and accidents, and from discussions with experts on dam design and safety. We have included the

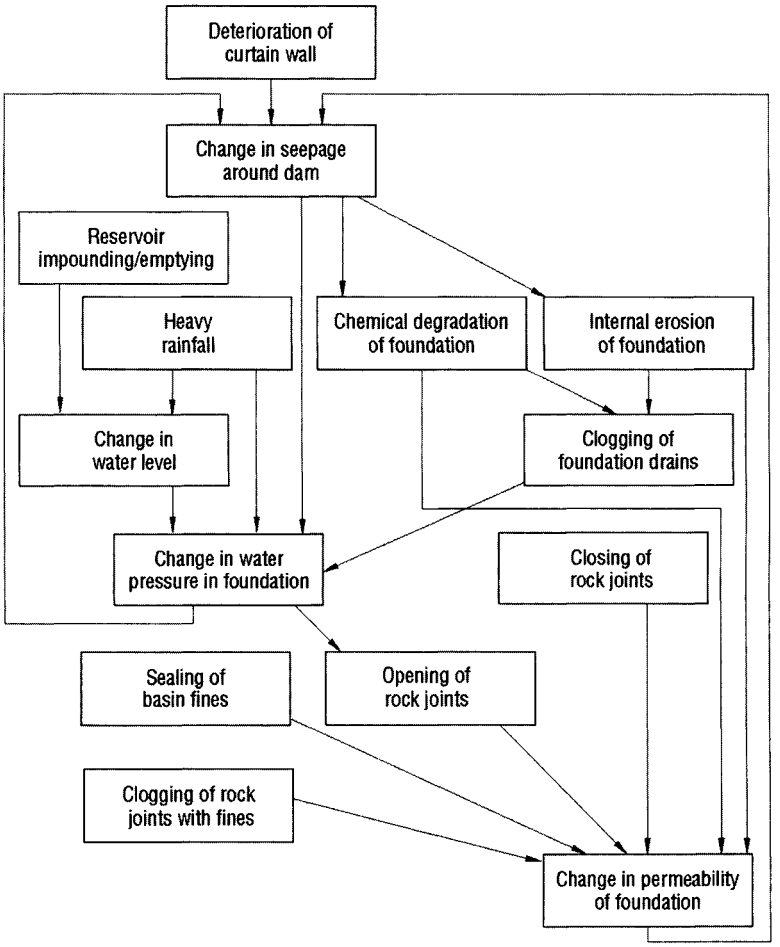


FIGURE 9 Damsafe: a section of the causal net of processes.

conditions under which one process can lead to another, and documented each of these processes along with descriptions of how evidence of these processes might be manifested in the monitoring data or in visual-inspection reports.

The network can be used in different ways:

- *As a database:* Each process has attributes that describe the process itself (e.g., start time, rate of change); among these attributes, the activation state expresses whether the process is within physiological thresholds or over them; both users and automatic reasoners may access and modify the attributes' values.
- *As a control panel of the system:* Each process is represented on the screen by a box that is highlighted whenever the system infers that the process is active. Therefore, the highlighted boxes give the user an immediate synthetic report on the dam's current state. Damsafe represents other attributes besides the activation state (for example, reversibility, speed) by colored areas of the box associated to the process.
- *As an inference tool:* Automatic reasoners can use the causal links to build event paths for simulating the system state's future evolution or identifying the possible causes of an active process. In the first case the net acts as an oriented graph, where a causal flow propagates through the net from the cause processes (e.g., rainfall, earthquake) to the effect processes (e.g., structural cracking) via activation rules. With these rules, Damsafe sets the state of each process on the basis of some conditions on its input (the links from its causes) and defines how each process influences its effects. In the second case, Damsafe applies consistency rules to chains of processes, using the activation state of some processes as evidence for the state of those processes that cannot have any direct evidence from the available data.
- *As a knowledge base:* Each process has hypertextual links to its written documentation that describes the process and its connections to other entities (processes and objects). Therefore, users can easily access the system's theoretical foundations through the user interface.

3. Reasoning Agents

Three reasoning agents have been developed. The first one (*extractor*) operates solely on the data world to manipulate data and extract features from the data sets. It uses the graphical interface to show to the user a time-series plot and to interactively identify features of the plot considered relevant to dam safety. They are defined by qualitative and quantitative attributes (e.g., spike length, start time) and stored within the data world. These attributes can also be accessed and manipulated through methods of the data world.

The second reasoning agent (*mapper*) performs the task of interpretation by identifying the possible behaviors of the dam in terms of a set of processes in the causal net, and the values of various attributes of the dam, based on evidence in the data.

This task is performed by firing production rules defined by experts, which link data values to dam states (Table 1). These links are defined by using a formal language designed to allow nonprogrammers to easily write and read rules (Table 2). When a rule is fired, the state of some dam world process is

TABLE 1 A Rule of the Mapper

```
RULE_5
    CONDITION ( Trend OF UnderPressure1TimesSeries
        OR
        Trend OF UnderPressure2Timeseries
        OR
        Trend OF UnderPressure3TimeSeries )
    ASSERT ( ChangeInWaterPressureInFoundation )
    SET ( Start_time OF ChangeInWaterPressureInFoundation TO
            MAX( Start_time OF Trend OF UnderPressure1TimeSeries,
                 Start_time OF Trend OF UnderPressure2TimeSeries,
                 Start_time OF Trend OF UnderPressure3TimeSeries )
        AND
        Finish_time OF ChangeInWaterPressureInFoundation TO
            MAX ( Finish_time OF Trend OF Underpressure1TimeSeries,
            Finish_time OF Trend OF Underpressure2TimeSeries,
            Finish_time OF Trend OF Underpressure3TimeSeries )
        AND
        Process_speed OF ChangeInWaterPressureInFoundation TO "slow" )
    MAP ( MAX ( Gradient OF Trend OF Underpressure1Timeseries,
            Gradient OF Trend OF Underpressure2Timeseries,
            Gradient OF Trend OF Underpressure3Timeseries )
        INTO
        RateOfChange OF ChangeInWaterPressureInFoundation )
```

declared active and some dam world attributes receive a value. The set of active processes linked in a causal chain is highlighted by the system and describes a scenario that demonstrates the evolution of the dam behavior.

The third reasoning agent (*enforcer*) acts on the dam world to extend the implications of the state identified by the mapper over the model of the dam and its environment, thus highlighting possible causal chains.

4. On-the-Fly Creation of Objects and Interface

A dam's entire information set might be inserted into the system by providing Damsafe with a description written in a special language, called ADAM.

TABLE 2 A Part of the Grammar of the Rule-Based Language Used by the Mapper

```
<ANiceRule> ::
            (CONDITION( <Condition> ),
            ASSERT( <ListOfDamProcesses> ),
            SET( <SetList> ),
            MAP( <MapList> ))
<Condition> ::
            <ExistentialCondition> | <RelationalCondition>
<ExistentialCondition> ::
            <ListOfFeatures>
<ListOfFeatures> ::
            <Feature> OF <DataObject> [ OR <ListOfFeatures> ]
```

We have designed ADAM so that users unfamiliar with programming languages can easily build their own data management environment [12].

Through ADAM descriptions users inform the system about the components of the dam currently under evaluation (for instance, blocks) and about the data sources to be linked (for instance, the database of monitoring measurements and its access code).

At run time, Damsafe interprets ADAM descriptions and sets up the necessary objects and links to databases, as well as a hypertextual interface based on HTML scripts (hypertext markup language). In this way, Damsafe can manage links to other databases or procedures (used as reasoning agents) on a net of computers via the hypertext transfer protocol (HTTP).

Furthermore, Damsafe translates ADAM descriptions of physical entities, processes of the causal net, and links among them into a hierarchy of C++ objects, which are then managed and accessed by the reasoning agents.

Damsafe was developed on workstations under UNIX using C++, the InterViews toolkit of the X-Window System, FORTRAN, and CGI scripts (Bourne shell). DAMSAFE is now integrated with the preexisting off-line system MIDAS, as well as with other databases of tests, inspections, and documents and is operational at ISMES to support a group of engineers and technicians who provide services related to periodic safety assessment of dams. Since its last release exploits the HTTP protocol and the HTML language for the interface, Damsafe can be run from any machine (PC, workstation, X-terminal) over our local network, by using a common Web browser.

C. DoDi, the Dams Dossier

Currently, we are exploiting the experience of Damsafe to create a new system that aims at providing users at ENEL, the Italian Electric Power Company, with access to databases and programs distributed over their geographic net concerning the whole set of dams managed by the company.

The organization that manages the problem is distributed over various geographical locations. The available information is distributed in many sites, and central databases collect specific types of data such as the data coming from the automatic and manual monitoring processes. The organization has in charge all the activities related to the structure, including safety management and production management.

The Dams Dossier (DoDi) has been designed as a WWW application, exploiting the intranet network of ENEL, to allow the user to easily access the information necessary to the dams management procedures.

The kernel of DoDi is hosted on a processing system configured as a Web server. The server accesses the remote databases through the TCP/IP protocol, the CGI interface, and if needed, local Web servers. Users can access the application through standard Web browsers.

The system links many existing databases hosted into several computers. The first release of the system integrates the following databases:

- measurements database (from monitoring systems or manual data acquisition);

- physical tests data base;
- basic reference data (for technical and administrative purposes); and
- document archives (safety reports, applicable laws, company procedures and rules).

Moreover the system may also access the real-time data produced by intelligent interpretation systems linked to real-time monitoring systems and provide real-time evaluations of the safety status of the structure.

The integration layer includes the man–machine interface written using HTML and a set of simple models, also written using HTML, which are visualized as icons in Fig. 10:

- *Single dam:* Through this model the user may select a specific dam and access the whole set of data, spread through various databases, related to the dam (Fig. 11). This model is mainly oriented to technical users aiming at evaluating the structural safety.
- *Set of dams:* The model makes it possible to navigate through data related to sets of dams and collects significant information related to a population of structures (e.g., for statistical purposes or economical evaluations).
- *Hierarchy of themes related to management procedures, laws, standards, company regulations:* This model allows a management-oriented user to access the information required to interface local and governmental authorities and to comply with the company regulations. The model may also link a specific subset of the available technical data.

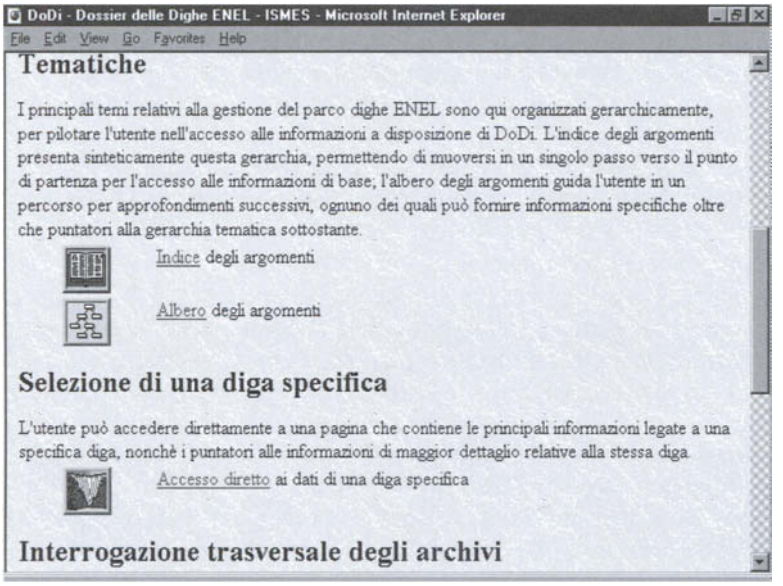


FIGURE 10 DoDi: accessing databases.

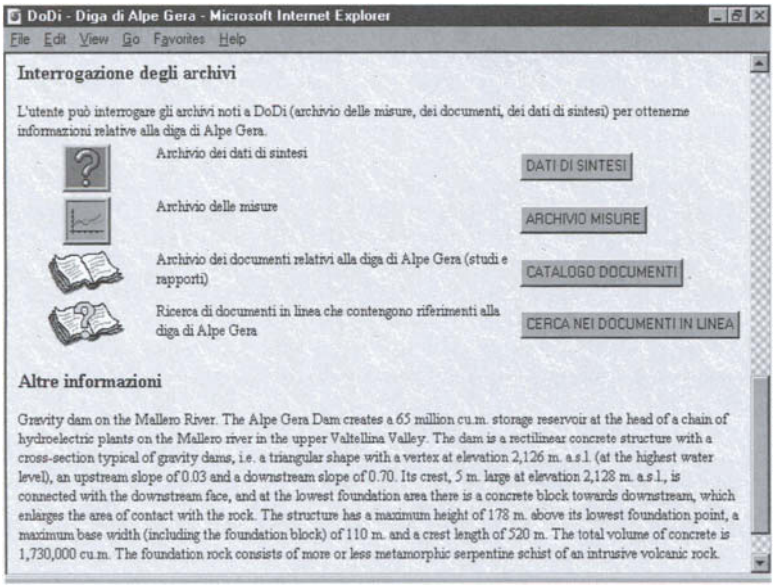


FIGURE 11 DoDi: accessing the whole set of data related to a specific dam.

Using the models the users may access multimedia information like texts, drawings, images, and data records. For the current implementation simple integration models have been used. If required, more complex object-oriented models may be added. For instance, a dam model may describe the dam as a hierarchy of objects (dam blocks, foundation, instrumentation system, sensors), each object including links to the available distributed information and methods to process it.

VI. CONCLUSIONS

We have seen how the integration of different information management technologies can help deal with safety problems. Artificial intelligence techniques can provide powerful tools for processing information stored in archives of safety-related data, such as measurements, documents, and test data. Internet technologies can be successfully coupled with these data processing tools to distribute data and knowledge among safety managers.

As a result of this process, we have shown a whole chain of software tools currently being used to face different aspects of the safety management procedures in different fields of engineering.

The main achievements of the joint application of these software techniques consist of the automatic support of those in charge of safety management, a performance that reduces the requests for expert intervention and the associated costs and delays, and increases the reliance upon the safety of the facility under examination.

REFERENCES

1. International Commission on Large Dams. *Lessons from Dams Incidents.* ICOLD, Paris, 1974.
2. Franck, B. M., and Krauthammer, T. *Preliminary safety and risk assessment for existing hydraulic structures - an expert system approach.* Report ST-87-05, University of Minnesota, Department of Civil and Mineral Engineering, Institute of Technology, 1987.
3. Grime, D. *et al. Hydro Rev.* **8**, 1988.
4. Comerford, J. B. *et al. Dam Engrg.* 3(4): 265–275, 1992.
5. National Research Council (U.S.), Committee on the Safety of Existing Dams. *Safety of Existing Dams.* National Academy Press, Washington, DC, 1983.
6. Anesa, F., *et al., Water Power Dam Construct.* **10**, 1981.
7. Salvaneschi, P. *et al. IEEE Expert* **11**(4): 24–34, 1996.
8. Lazzari, M. *et al.* Looking for analogies in structural safety management through connectionist associative memories. In *IEEE International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing (NICROSP '96)*, pp. 392–400. IEEE Computer Society, Los Alamitos, CA, 1996.
9. Brembilla, L. *et al.* Structural monitoring through neural nets. In *Second Workshop of the European Group for Structural Engineering Applications of Artificial Intelligence (EGSEAAI '95)*, Bergamo, Italy, 1995, pp. 91–92.
10. Lancini, S. *et al. Struct. Engrg. Internat.* 7(4): 288–291, 1997.
11. Lazzari, M., and Salvaneschi, P. *Internat. J. Natural Hazards* 20(2–3): 185–195, 1999.
12. Lazzari, M. *et al.* Integrating object-oriented models and WWW technology to manage the safety of engineering structures. In *Third Workshop of the European Group for Structural Engineering Applications of Artificial Intelligence (EGSEAAI '96)*, Glasgow, UK, 1996, pp. 93–96.

22

RELIABLE DATA FLOW IN NETWORK SYSTEMS IN THE EVENT OF FAILURES

WATARU KISHIMOTO

Department of Information and Image Sciences, Chiba University, Chiba 263-8522, Japan

Database and Data Communication Network Systems, Vol. 3

In a database and data communication network system, the network must not be allowed to fail or the entire system might suffer serious damage. We consider the design of a reliable data communication channel for reliable data flow in a network.

Network failures cannot be avoided entirely, so we need to design a system that will survive them. In this chapter, using a flow network model, we consider the design of a reliable communication channel with well-specified network survivability performance.

There are two approaches for constructing reliable communication channels, stochastic and deterministic. Here, we mainly consider the deterministic approach, in which the performance in the event of link or node failure is designed to exceed a predetermined level. We classify this approach as a survival method or a restoral method.

The survival method is to confirm the deterministic lower bound of the channel capacity in the event of link or node failure, while the restoral method is to reserve a restoral channel. With given network resources, we show methods for determining whether the required channels exist. Moreover, when such channels do not exist, the methods determine the locations of the bottlenecks so that we can improve the network resources efficiently. We consider these problems as reliable-flow problems in a network.

I. INTRODUCTION

A. Survivability

In a database and data communication network system, the network must not be allowed to fail or the entire system might suffer serious damage [1–4]. In this chapter we consider the design of a reliable data communication channel for reliable data flow in a network.

Network failures cannot be avoided entirely, so we need to design a system that will survive them. With given network resources, this chapter shows methods for determining whether the required channels exist. Moreover, when such channels do not exist, the methods determine the locations of the bottlenecks so that we can improve the network resources efficiently [19–21]. We consider these problems as a reliable flow problem in a network.

Usually, a network is designed to satisfy some specified performance objectives under normal conditions, without explicit consideration of network survivability. How such a network will perform in the event of a failure is difficult to predict. A distributed database system with two or more host computers has duplicate databases in different sites. When some data are changed in one of those computers, that data must be transmitted to the other computers to update them. This system requires reliable communication channels for accurate data exchange. If all the communication channels between the computers fail, the system cannot maintain identical versions of the data in them. For such a system, we need to specify a desirable survivability performance. Setting the objectives of survivability performance will enable us to ensure that, under given failure scenarios, network performance will not degrade below predetermined levels. The performance of the database system avoids the worst case, in which just one network failure causes a system outage. A system outage caused by a network failure is difficult to repair. With network equipment spread over a wide area, it takes a long time to locate the failure. Then repairing the failed equipment takes time too. A reliable communication channel with well-specified

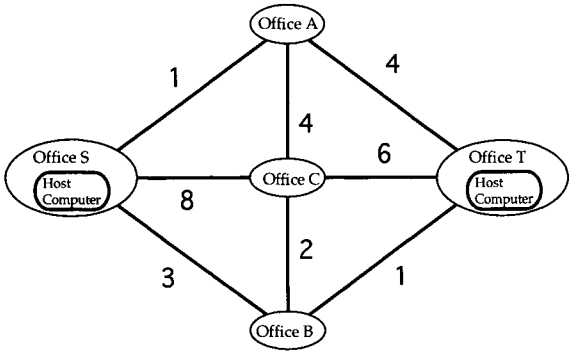


FIGURE 1 A simple example of a data network system.

survivability performance avoids this problem. In this chapter, we consider the design of such a reliable communication channel.

B. Simple Example

A company has a private communication network connecting its head office to its branches, and its branches to each other. The connection (rented from a public network) between any two adjacent offices is called a "link." A channel may comprise one or more links, perhaps a cable with the capacity required by the channel. Each office is a node that can connect channels between offices. For example, a communication channel between offices A and B goes through a sequence of two links between the two offices: one link is between A and another office C, and the other link is between C and B. A simple example of the problem this paper considers is as follows:

1. Example

The company designs a database system that has two host computers in separate offices S and T (Fig. 1) to ensure the "reliability of data and efficiency of distributed computing." Each host computer contains the same database. If data are changed in one of the computers, the new data must be transmitted to the other computer so that the latest version will be in both computers. This system requires communication channels large enough for the volume of data exchange. If all the communication channels between the two computers fail, the system cannot maintain identical versions of the data. Then the system needs a reliable backup communication channel.

C. Techniques for Making a Reliable Communication Channel

In the example of the previous section, if one link failure causes all the communication channels between the two computers to fail, the system is seriously fragile. If one link failure causes several communication channels to fail and severely degrades the system performance, the system is weak. Therefore, the communication channels between the two host computers should use

physically independent communication channels in the network using a disjoint set of communication links.

There are two approaches for constructing a reliable communication channel: one is stochastic and the other is deterministic.

In the stochastic approach, each link has a known probability of failure, or link capacities in a network are modeled as random variables. Using these links we construct some channels between two host computers. Then the total capacity of these channels is estimated stochastically. If this capacity exceeds the predetermined level (the desired average channel capacity), the design is complete. To improve the total channel capacity, we increase the capacity of some links. This improvement is continued until the desired channel capacity is exceeded. The aim of this approach is to increase the channel capacity so it will meet the estimated channel use. In this approach, estimating the stochastic channel capacity and determining the highly probable bottlenecks are both important, but both of these tasks are intractable. Rather than discuss this approach further, we refer the reader to Refs. [5–15].

In the deterministic approach, the performance in the event of link or node failure is designed to exceed a predetermined level. Here, we mainly consider this approach [25–31]. We classify this approach as a survival method or a restoral method. The survival method is to confirm the deterministic lower bound of the channel capacity in the event of link or node failure, while the restoral method is used to reserve a restoral channel. Details are given in the following sections.

D. Survival Method (Diversification): δ-reliable Channel

Through this survival method, we confirm the deterministic lower bound of the channel capacity in the event of link or node failure. For this aim, it is important to find out to what extent communication channels and their capacities can be damaged by link failures.

DEFINITION 1. A *δ-reliable channel* is a set of communication channels along a set of paths, such that a maximum of the fraction $n\delta(0 < n\delta \leq 1)$ of the total capacity of the channels is failed when n links fail.

The fraction $(1 - n\delta)$ of the total capacity of the channels survives when n links fail. The lower the value of δ, the better the δ-reliable channel.

Figure 2 shows an example of a 0.6-reliable channel between offices S and T. The total capacity of the channel is 5, so if one link with a capacity of 3 fails, the capacity 2, the fraction $0.4(= 1 - 0.6)$ of the channel, survives the link failure as shown in Fig. 3.

For a node failure, the surviving capacity of the channel cannot be specified because the δ-reliable channel is concerned only with link failures. This is the link version of the δ-reliable channel. A δ-reliable channel concerned with node failures could also be defined as the node version of a δ-reliable channel. The channel of Fig. 2 is also a 0.6-reliable channel of node version.

To improve its survivability, the network requires additional channel capacity. However, in a private communication network, the capacity of each communication link is limited by the available budget. The maximum channel

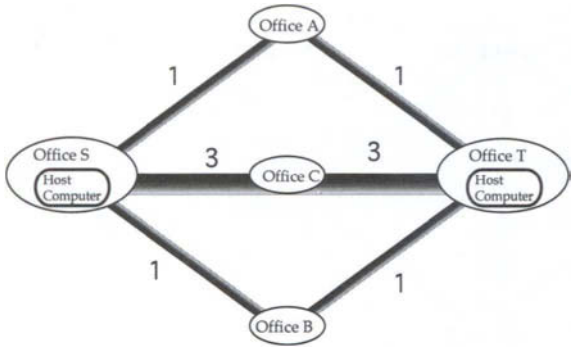


FIGURE 2 An example of a 0.6-reliable channel.

capacity of the δ-reliable channel and the bottlenecks of the channel are useful information for the design of the network.

E. Restoral Method (Reservation): *m*-Route Channel

By the restoral method, we reserve a dedicated channel for a channel or a set of channels. For this aim we may use a multiroute channel, that is, m ($m \geq 2$) physically separate communication paths between two offices.

DEFINITION 2. An *m-route channel* is a set of m communication channels with the same channel capacity, each of which traces along a set of paths between a specified pair of nodes, such that each pair of paths of the set is link-disjoint.

For example, we consider the case $m = 2$, a 2-route channel consisting of a pair of channels of the same capacity: one a working channel and the other a reserved channel. The concept is used as automatic protection switching [2]. Figure 4 shows an example of a 2-route channel between S and T. The channel consists of a pair of paths between S and T. These paths have the same capacity 2. Since this pair of paths goes through link-disjoint paths, if a link failure

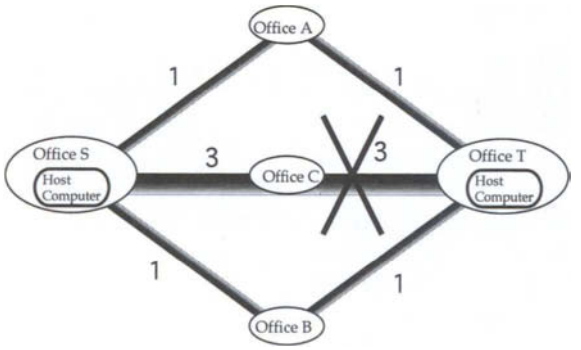


FIGURE 3 A link failure.

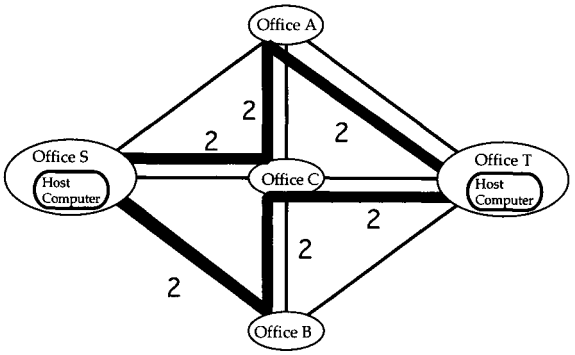


FIGURE 4 A 2-route channel between S and T.

occurs and stops the working channel, we can replace the failed channel with the reserved channel, which is not affected by the link failure.

In general, for an m-route channel, if we reserve one channel and use the other $m-1$ channels to work, a link failure can affect only one of the working channels, and we can replace the failed channel with the reserved channel having the same channel capacity.

Since the m-route channel of Definition 2 consists of link-disjoint paths, the definition is a link version of the m-route channel. As in the case of the link version of the δ-reliable flow, we can define a node version of the m-route channel, which consists of node-disjoint paths. The channel of Fig. 4 is not a 2-route channel of node version.

There are a variety of uses for m-route channels. For an essential communication channel, the m-route channel is used because $m-1$ channels are reserved and only the one channel is working. For more economical use, two channels are reserved, and $m-2$ channels use them to restore communication. If there are several failures, this economical method might not restore communication completely. It is important to find out to what extent communication channels can be constructed as m-route channels.

F. Flow Network: As a Model of a Communication Network

In this section we depict the modeling of a communication network system by a flow network.

A network can be modeled by a graph in which $c(e)$ is associated with each edge. Each link in the network with a specific capacity is modeled by an edge with the capacity, and each office is modeled by a vertex. Figure 5 shows a flow network corresponding to the network of Fig. 1. A communication channel between two offices corresponds to a flow along a single path between corresponding vertices in a network. A set of communication channels between two offices corresponds to a single commodity flow along a set of paths between two vertices. With this correspondence, a δ-reliable flow and an m-route flow are defined as a δ-reliable channel and an m-route channel, respectively.

For the design of a surviving network, the goal is to establish a δ-reliable channel or an m-route channel between two vertices according to the system requirement. The objective of this chapter is as follows: using a flow network

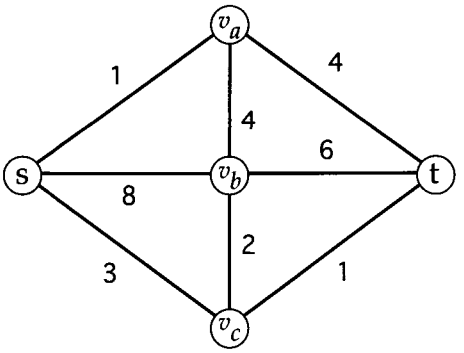


FIGURE 5 A flow network N_0 corresponding to the example data network system.

corresponding to the network resources available for the system, we will determine whether a δ-reliable flow or an m-route flow corresponding to the required channel exists. For this purpose we show some methods for evaluating the maximum δ-reliable flow and m-route flow. For the m-route flow we show the procedure of the synthesis of m-route flows, which are not easy to obtain and correspond to the assignment of the m-route channel to the data communication network. If such a flow does not exist, with these methods we can find the locations of the bottlenecks that limit the flow of the commodity. Finding the bottlenecks is important in network design, so we can improve the network resources efficiently.

G. Organization of This Chapter

Section II states the terms of a flow network, which is used as a model of a data communication network in this chapter. Sections III and IV are devoted to the survival method where δ-reliable flow corresponding to the δ-reliable channel is depicted; in Section III, the edge version of δ-reliable flow is discussed, and in Section IV, the vertex version of δ-reliable flow. In Section V, the restoral method is taken up and m-route flows of edge version and vertex version, which correspond to m-route channels, are described. Section VI is the summary of this chapter.

II. FLOWS IN A NETWORK

A. Preliminaries

Let $G = (V, E)$ be a graph with the vertex set V and the edge set E. Then $N = (G, c(*))$ is a network, where $c(*)$: $E \rightarrow \mathcal{R}_+$ denotes edge capacities of N. If F is a set of edges in N, $c(*)$ is expanded to give $c(F) = \sum_{e \in F} c(e)$, and we call $c(F)$ the capacity of F. When G is directed, N is directed; otherwise, N is undirected. We indicate an edge connecting from v_a toward v_b by (v_a, v_b). If N is directed, (v_a, v_b) is an ordered pair.

For each edge e of a directed network, e^r represents the edge that connects the same pair of vertices and is directed oppositely to e. We say e^r is a reverse edge of e and e is that of e^r. For an edge $e = (v_a, v_b)$, e^r is (v_b, v_a).

DEFINITION 3. Let $N_a = (G_a, c_a(*))$ and $N_b = (G_b, c_b(*))$ be two networks. If G_b is a subgraph of G_a and $c_b(e) \leq c_a(e)$ for each edge e of G_b, then N_b is a *subnetwork* of N_a.

DEFINITION 4. Let $N_1 = (G_1, c_1(*))$, $N_2 = (G_2, c_2(*)), \ldots, N_n = (G_n, c_n(*))$ be n networks. Let V_i, E_i be the vertex set and the edge set of G_i, and V, E be the vertex set and the edge set of G. The sum of $N_1, N_2, \ldots, N_n$ is $N = (G, c(*))$ such that

$$V = V_1 \cup V_2 \cup \ldots \cup V_n, \tag{1}$$

$$E = E_1 \cup E_2 \cup \ldots \cup E_n, \tag{2}$$

$$c(e) = c_1(e) + c_2(e) + \cdots + c_n(e) \text{ for all } e \in E, \tag{3}$$

$c_i(e) \equiv 0$ *for* $e \notin E_i$.

DEFINITION 5. The directed sum of $N_1, N_2, \ldots, N_n$ is $N' = (G, c'(*))$ obtained from N, the sum of those networks as follows: In N if there exist some pairs of edges with opposite directions to each other between two vertices, we perform the next operation. For each of the pairs, the smaller capacity of those edges is subtracted from the other, the greater capacity. Then the value is assigned to the edge with the greater capacity, and the edge with the smaller capacity is deleted.

DEFINITION 6. A network is λ *uniform* if each edge has capacity λ.

An undirected network N_0 of Fig. 5 corresponds to the example network of Fig. 1. When we treat an undirected network, we replace each of its edges with a pair of edges directed opposite to each other having the same edge capacity. With this transformation, we can treat any undirected network as a directed network. Therefore, a directed network N'_0 of Fig. 6 corresponds to both undirected flow network N_0 and the data network of Fig. 1.

From now on, N is assumed to be a directed network.

B. Cut and Mixed Cut

Let V_1 be a nonempty subset of V of network N, and V_2 be the complement of V_1. We denote by U^c the complement of a subset U of V.

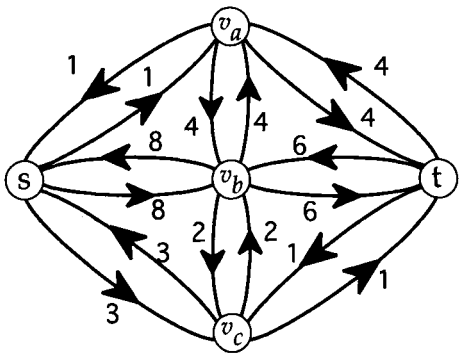


FIGURE 6 A flow network N_0 corresponding to the example data network system.

DEFINITION 7. In a directed network N, a cut $\langle V_1, V_1^c \rangle$ of N is a set of edges, each of which is directed away from a vertex in V_1 and directed toward a vertex in V_1^c.

A cut is an edge set, removal of which disconnects the network.

A set of edges incident onto a vertex v is $I(v)$, the incidence cut of v. In a directed network, a set of the edges directed away from v is $I_o(v)$, called the out-incidence cut of v, and a set of edges directed into v is $I_i(v)$, the in-incidence cut of v.

Let $s \in V_1$ and $t \in V_1^c$, then $\langle V_1, V_1^c \rangle$ is an s–t cut. The sum of all $c(e)$ of an edge set S is the capacity of S; the capacity of a cut $\langle V_1, V_1^c \rangle$ is simply represented by $c\langle V_1, V_1^c \rangle$ instead of $c(\langle V_1, V_1^c \rangle)$.

In Sections IV and V we consider the case in which the flow value that goes through each vertex in N is restricted. For this purpose we use the capacity of a vertex. As the capacity of an edge represents the maximum flow that can go through the edge, the capacity of a vertex should represent the maximum flow that can go through the vertex. For ordinary flows we need not restrict the capacity of a vertex, but for convenience we set the capacity of a vetex to a finite value. Since the maximum value that can go through a vetex is restricted by the capacities of the edges incident to the vertex. Unless otherwise stated, we set the capacity of a vertex as follows.

DEFINITION 8. The capacity of v in $N = (G, c(*))$ is

$$c(v) = \min\{c(I_i(v)), c(I_o(v))\}. \tag{4}$$

When the capacity of a vertex is defined as above, the vertex is the same as a vertex with infinite capacity in practice. From now on, if we say $N_b = (G_b, c_b(*))$ is a subnetwork of $N_a = (G_a, c_a(*))$, then besides Definition 3 the equation should be satisfied:

$$c_b(v) \leq c_a(v) \text{ for each vertex } v \text{ in } G_b. \tag{5}$$

DEFINITION 9 [25]. In a directed network $N = (G, c(*))$, let V_1 and V_2 be two nonempty proper subsets of the vertex set V of G such that $V_1 \cap V_2 = \emptyset$. The *mixed cut* is a set of all edges directed from a vertex in V_1 to a vertex in V_2, and all vertices in $V - V_1 - V_2$. We use the symbol $\langle V_1, V_2 \rangle$ for this mixed cut,

$$c\langle V_1, V_2 \rangle \equiv \sum_{e \in E^{(c)}} c(e) + \sum_{v \in V^{(c)}} c(v), \tag{6}$$

where E^c is the set of all edges in $\langle V_1, V_2 \rangle$, and $V^{(c)}$ is the set of all vertices in $\langle V_1, V_2 \rangle$.

If $V_1 \cup V_2 = V$, then the mixed cut $\langle V_1, V_2 \rangle$ coincides with an ordinary cut $\langle V_1, V_2 \rangle$. If $s \in V_1$ and $t \in V_2$, then $\langle V_1, V_2 \rangle$ is an s–t mixed cut. A cut and a mixed cut divide a network into two parts.

For a network $N_A = (G_A, C_A(*))$ of Fig. 7 let $V_1 = \{s, v_c\}$. Then $\langle V_1, V_1^c \rangle$ is a cut consisting of edges (s, v_a), (s, v_b), (v_c, v_b), (v_c, t), and $c\langle V_1, V_1^c \rangle = 12$. Let $V_2 = \{v_a, t\}$. Then $\langle V_1, V_2 \rangle$ is a mixed cut consisting of edges (s, v_a), (v_c, t), and a vertex v_b with $c\langle V_1, V_2 \rangle = 12$.

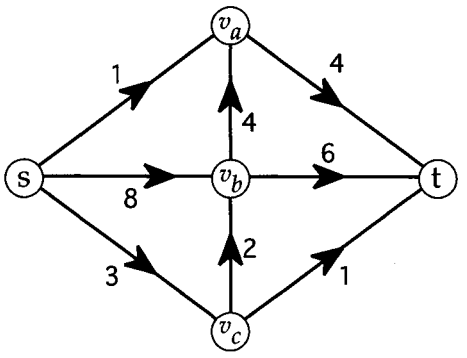


FIGURE 7 A flow network N_A.

C. Definition of Flow

A flow from s to t has two types of definitions; one is defined as a function $f(*)$ from the edge set of a network to nonnegative real numbers [32], and the other is Definition 11, which is used here.

DEFINITION 10 [25]. A directed s–t path-flow $P = (G_p, c_p(*))$ with a path-flow value of λ is a uniform network of value λ such that G_p is a directed path from s to t.

DEFINITION 11 [25]. A directed flow from s to t is the network $F = (G_f, c_f(*)) =$ "directed sum of directed s–t path-flows $P_1, P_2, \ldots, P_n$." The value of F is the sum of the values of those path-flows.

Let a path-flow F be a subnetwork of N; then F is a path-flow in N. There is no essential difference between Definition 11 and the ordinary definition using $f(*)$. If F is a flow from s to t, for each vertex v in the vertex set, excluding s and t, $c(v) = c(I_i(v)) = c(I_o(v))$.

In a flow the capacities of an edge represent the flow value at the edge and the capacities of a vertex represent the flow value at the vertex. The flow value at a vertex is the value of the flows that go through the vertex. Therefore, for a flow it is natural that the capacity of a vertex be defined as in Definition 8. For networks in which the capacity of a vertex is defined by Eq. (4), Eq. (5) is always satisfied. Therefore, for considering the flows defined above, we can ignore the capacities of vertices.

Network P_1 and P_2 of Figs. 8a and 8b are s–t path-flows in N_A of Fig. 7. Let F of Fig. 8c be a directed sum of P_1 and P_2. Then F is a flow with value 4 from s to t in N_A.

D. Maximum Flow

The next theorem is one of the basic theorems of network flow theory.

THEOREM 1 [32]. *In a network N, the "maximum flow from s to t" = the "minimum capacity of s–t cuts in N."*

The maximum flow in a network corresponds to the maximum channel capacity between two offices. There are many methods for obtaining the

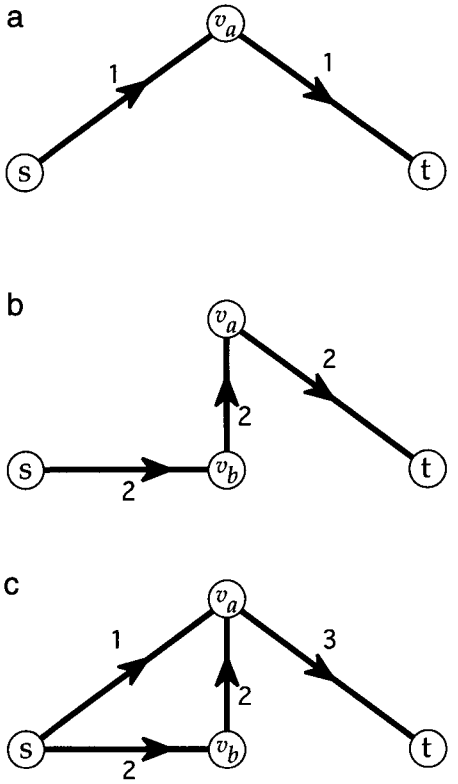


FIGURE 8 An example of flows in N_A. (a) P_1, (b) P_2, and (c) F.

maximum flow between a specified pair of vertices. To make this chapter self-contained, we show one of the most simple methods here. For a more efficient method refer to [24, 32–36].

DEFINITION 12. Let $N = (G, c(*))$ be a network and $F = (G_f, c_f(*))$ be a flow in N. For N and F a *residual network* is defined to be a network $N_R = (G_R, c_R(*))$ as follows. For each edge e in G,

- if $c(e) - c(e^r) - c_f(e) + c_f(e^r) > 0$, we set

$$c_R(e) = c(e) - c(e^r) - c_f(e) + c_f(e^r) \tag{7}$$

  and remove e^r, where e^r is a reverse edge of e,
- if $c(e) - c_f(e) + c_f(e^r) = 0$, we remove e and e^r, and
- if $c(e) - c(e^r) - c_f(e) + c_f(e^r) < 0$, we set

$$c_R(e^r) = -c(e) + c(e^r) + c_f(e) - c_f(e^r) \tag{8}$$

  and remove e.

ALGORITHM 1. For a given network N and a pair of vertices s and t, we find the maximum flow from s to t, and a minimum s–t cut.

1. Let $N_1 = N$, and F_1 be an empty network.

2. Iterate the next procedure for $k = 1, 2, \ldots$, until we cannot find a path from s to t in N_k.
   Procedure:
   (a) Using Algorithm 2 stated below, find a path π_k from s to t with the minimum number of edges in N_k.
   (b) Let f_k be the minimum edge capacity of π_k.
   (c) Let $P_k = (\pi_k, c_k(*))$ be a path flow from s to t, where $c_k(e) = f_k$ for each edge e in P_k.
   (d) Set a flow F_{k+1} as a directed sum of F_k and P_k.
   (e) Set N_{k+1} as a residual network for N and F_k.

   (end of procedure)
3. Let $V_1 = \{v \mid$ there is a path from s to v in $N_k\}$. Then cut $\langle V_s, V_s^c \rangle$ is a minimum s–t cut in N.

(end of algorithm)

ALGORITHM 2. For a given network N and a pair of vertices s and t, we find a path from s to t with the minimum number of edges.

1. For each vertex v in N we say $L(v)$ is a label of v. At first, all vertices are unlabeled.
2. Set $L(s) = 0$ (s is labeled to 0).
3. Iterate the next procedure for $j = 1, 2, \ldots$, until $L(t)$ is set to j (t is labeled to j).
   Procedure:
   For each vertex v such that $L(v) = j - 1$, if an edge from v to an unlabeled vertex x exists, we set $L(x) = j$ (x is labeled to j).

   (end of procedure)
4. Assume $L(t) = j$; that is, the minimum number of edges of a path from s to t in N is j. There is a vertex v_{j-1} that connects to t with $L(v_{j-1}) = j - 1$.
5. For each vertex v_m with $L(v_m) = m$ we can find a vertex v_{m-1} that connects to v_m with $L(v_{m-1}) = m - 1$.
6. Then we can get a path $s, v_1, \ldots, v_{j-1}, t$ with the minimum number of edges in N.

(end of algorithm)

EXAMPLE 1. We find the maximum flow from s to t in N_A of Fig. 7.

1. Let $N_1 = N$, and F_1 be an empty network.
2. For $k = 1$,
   (a) Using Algorithm 2, find a path π_1 from s to t with the minimum number of edges in N_1.
      i. Set $L(s) = 0$ (s is labeled to 0).
      ii. $j = 1$. For s ($L(s) = 0$), and unlabeled vertices v_a, v_b, and v_c, there are edges (s, v_a), (s, v_b), and (s, v_c). We set $L(v_a) = L(v_b) = L(v_c) = 1$.
      iii. $j = 2$. For v_a($L(v_a) = 1$), and unlabeled vertices t, there is an edge (v_a, t). We set $L(t) = 2$.

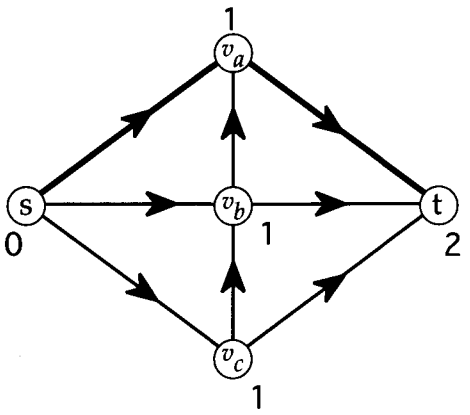


FIGURE 9 Labels of vertices in N_1.

iv. Then $L(t) = 2$. The minimum number of edges of a path from s to t in N is 2. There is a vertex v_a that connects to t with $L(v_a) = 1$.

v. For vertex v_a with $L(v_a) = 1$ we can find a vertex s that connects to v_a with $L(v_s) = 0$.

vi. Then we can get a path s, v_a, t with the minimum number of edges in N_1. Figure 9 shows this minimum s–t path with labels of vertices beside each vertex.

(b) Let $f_1 = 1$, the minimum edge capacity of π_1.

(c) Let $P_1 = (\pi_1, c_1(*))$ be a path flow from s to t, where $c_1(e) = f_1 = 1$ for each edge e in P_1.

(d) Set a flow F_2 as a directed sum of F_1 and P_1. Since F_1 is empty network, $F_2 = P_1$.

(e) Set N_2 as a residual network for N and F_2. Figure 10 shows N_2.

3. For $k = 2$, 3, 4, and 5, details are omitted.

4. Figure 11 shows F_6, and Fig. 12 shows residual network N_6 for N and F_6. Since there is no path from s to t in N_6, F_6 is a maximum flow from s to t in N, and the maximum flow value is 11. In N_6 for each of v_a, v_b, and v_c, there is a path from s. Let $V_a = \{s, v_a, v_b, v_c\}$. Then cut $\langle V_1, V_1^c \rangle$ is a minimum s–t cut in N.

(end of example)

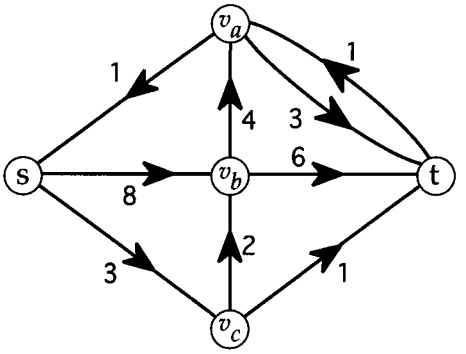


FIGURE 10 Network N_2.

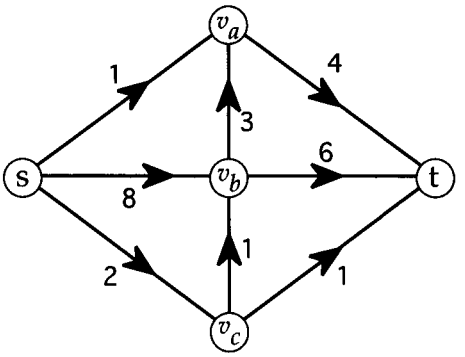


FIGURE 11 Flow F_6.

E. Cut-Flow

Until now we have discussed ordinary flows, in which the capacity of a cut can be defined as being the sum of the edges in the cut. Next we introduce a cut-flow for a cut in a network. Using this concept we can define the capacity of a cut for a flow with a specified reliability against edge or node failure.

DEFINITION 13 [25]. Let $\langle V_1, V_1^c \rangle$ be a cut of N. Let $F_c = (G_f, c_f(*))$ be a subnetwork of N such that "each edge of G_f" $\in \langle V_1, V_1^c \rangle$. Then F_c is a *cut-flow* of $\langle V_1, V_1^c \rangle$ and the value of $F_c \equiv c_f\langle V_1, V_1^c \rangle$.

Since $F_c = (G_f, c_f(*))$ is a subnetwork of N, then $c_f(e) \leq c(e)$ for each edge e in G_f. The "maximum value of cut-flows of $\langle V_1, V_1^c \rangle$" $= c\langle V_1, V_1^c \rangle$. A cut-flow of $\langle V_1, V_1^c \rangle$ is a flow between a pair of vertex sets, instead of between a single pair of vertices.

DEFINITION 14 [25]. Let $\langle V_1, V_2 \rangle$ be a mixed cut of N, and $F_c = (G_c, c_c(*))$ be a subnetwork of N such that each edge of G_c is an edge in $\langle V_1, V_2 \rangle$ and each vertex of G_c is a vertex in $\langle V_1, V_2 \rangle$ or an endpoint of an edge in $\langle V_1, V_2 \rangle$. Then F_c is a *cut-flow* of $\langle V_1, V_2 \rangle$ in N and its value is $c_c\langle V_1, V_2 \rangle$. For a cut-flow of a mixed cut, Eq. (4) of Definition 8 may not be satisfied.

The "maximum value of cut-flows of a mixed cut $\langle V_1, V_2 \rangle$" $= c\langle V_1, V_2 \rangle$.

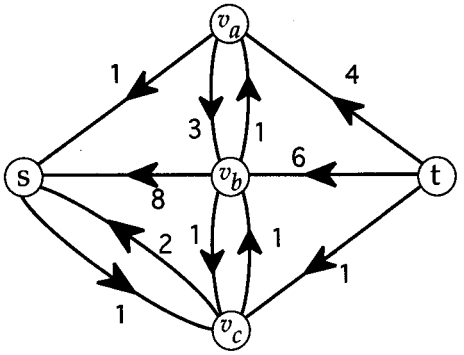


FIGURE 12 Network N_6.

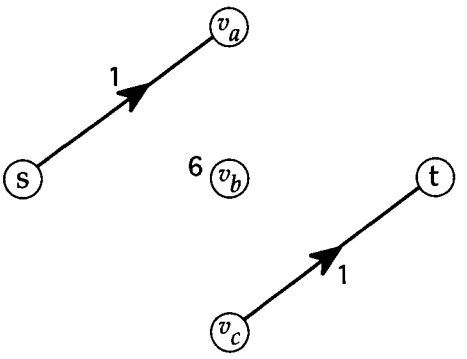


FIGURE 13 A cut-flow F_0.

Let $V_1 = \{s, v_c\}$, $V_2 = \{v_a, t\}$. A cut $\langle V_1, V_2 \rangle$ consists of edges (s, v_a) and (v_c, t) and a vertex v_b. Cut-flow $F_0 = (G_0, c_0(*))$ of Fig. 13 is an example of a cut-flow of a mixed cut $\langle V_1, V_2 \rangle$ in $N_A = (G_A, C_A(*))$ of Fig. 7. Since $c_A(v_b) = 10$, capacity $c_0(v_b)$ $(= 6)$ satisfies $c_0(v_b) \leq c_A(v_b)$. However, $c_0(v_b)$ does not satisfy Eq. (4) of Definition 8 in F_0.

A cut or a mixed cut divides a network into two parts. A cut-flow is a flow between those two parts, that is, a flow from one part of the network to the other part of the network. Obviously, a cut-corresponding part of a flow in a network, where capacities of edges and vertices are the same as those of the flow, is a cut-flow of the cut. As Theorem 1 (the max-flow min-cut theorem), the "minimum capacity of s–t cuts in N" is a bottleneck of the "maximum flow from s to t," which corresponds to the maximum channel capacity between two nodes. Therefore, in the succeeding sections we use cuts for specifying the bottleneck of a flow that corresponds to a channel for the survival method or restoral method.

F. Matching in a Bipartite Graph

For an undirected graph the degree of each vertex v is the number of edges incident to v. An undirected graph is said to be regular to degree n if the degree has the same value n at each of the graph's vertices. An n-factor of an undirected graph is a spanning regular subgraph of degree n.

DEFINITION 15 [25]. A 1-factor network of value μ of an undirected network N is a uniform subnetwork of value μ such that the graph of the subnetwork is a 1-factor of the graph of the network.

An undirected graph G is said to be bipartite if its node set V can be partitioned into two disjoint subsets X and Y($V = X \cup Y$, $X \cap Y = \emptyset$) such that each of its edges has one endpoint in X and the other in Y. We refer to the bipartite graph as $G = (X, Y; E)$, where E is the edge set of G. An undirected network is said to be bipartite if its graph is bipartite.

DEFINITION 16. For a bipartite graph $G = (X, Y; E)$ the subset E_0 of its edge set E is a matching of G if no pair of edges in E_0 has a common endpoint. Then the endpoint v of each edge in E_0 is said to be saturated.

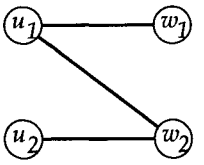


FIGURE 14 A cut-flow G_b.

A graph $G_b = (X_b, Y_b; E_b)$ of Fig. 14 is a simple example of a bipartite graph, where $X_b = \{u_1, u_2\}$ and $Y_b = \{w_1, w_2\}$. In G_b, edges (u_1, w_1) and (u_2, w_2) form a matching with the maximum number of edges.

There are many methods for finding a matching with the maximum number of edges in a bipartite graph. We show a method using an algorithm for obtaining the maximum flow in a network.

ALGORITHM 3. Let $G = (X, Y; E)$ be a bipartite graph. We will find a matching of G with the maximum number of edges.

1. For all edges (v, w) in G where $v \in X$ and $w \in Y$, assign a direction from v to w.
2. Add two vertices s and t to G. Connect s and each vertex v in X with a directed edge (s, v). Connect t and each vertex w in Y with a directed edge (w, t).
3. Assign all edges the edge capacity 1. Let the resulting network be N_m.
4. Find the maximum flow from s to t in N_m. From the integrity theorem of flow network theory, in such a network N_m with integer edge capacities, the maximum flow value is an integer, and there is a maximum flow in which the flow value at each edge is an integer. Using Algorithm 1 we can find such a maximum flow F_m in N_m.
5. In G the edges corresponding to the edges with flow value 1 in N_m form a maximum matching.

(end of algorithm)

Applying Algorithm 3 to bipartite graph G_0 of Fig. 14, we get network N_m of Fig. 15.

Then we can find the maximum flow F_m from s to t in N_m depicted in Fig. 16. From this maximum flow we can get a maximum matching consisting of (u_1, w_1), (u_2, w_2) in G_0.

DEFINITION 17. An undirected network N is even of value λ if each incidence cut of its vertices has the same capacity λ.

The next lemma is a variation of König's theorem [16–18] for a regular bipartite graph.

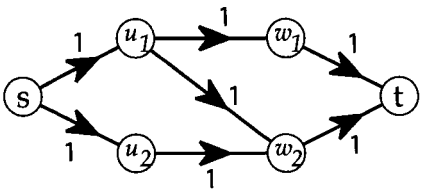


FIGURE 15 Network N_m.

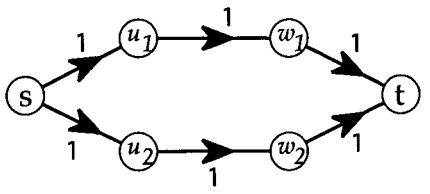


FIGURE 16 Flow F_m.

LEMMA 1 [25]. *If a bipartite network* $N = (G, c(*))$ $(G = (X, Y; E)(|X| = |Y|))$ *is even of value* λ, *N is the sum of a finite number of 1-factor networks that are obtained by applying Algorithm 4. The sum of values of those 1-factor networks is* λ.

ALGORITHM 4. For a given bipartite network $N = (G, c(*))$ $(G = (X, Y; E)$ $(|X| = |Y|))$, which is even of value λ, we will find 1-factor networks such that the sum of their values is λ.
(operation)

1. Find a matching M in which all vertices of X are saturated in G.
2. Obtain a 1-factor network by assigning the minimum of the edge capacities of M to all edges in M as edge capacities.
3. Subtract the 1-factor network from N.

(end of operation)
(end of algorithm)

EXAMPLE 2. A bipartite undirected network $N_B = (G_B, c_B(*))$ $(G_B = (X_B, Y_B; E_B))$ of Fig. 17 is even of value 3. In N_B, let $X_B = \{u_1, u_2, u_3\}$ and $Y_B = \{w_1, w_2, w_3\}$. Therefore, $|X_B| = |Y_B| = 3$. Using Algorithm 4 we will represent N_B as a sum of 1-factor networks.

1. Using Algorithm 3 we can find a maximum matching M_1 in G_B, which consists of edges (u_1, w_1), (u_2, w_2), (u_3, w_3).
2. In N_B the minimum edge capacity of M_1 is 2. We get a 1-factor network O_1 of Fig. 18a.
3. Subtracting O_1 from N_B we get N_{B0} of Fig. 18b. Since in this case N_{B0} is a 1-factor network, we set $O_2 = N_{B0}$. We represent N_B as the sum of O_1 and N_{B0}.

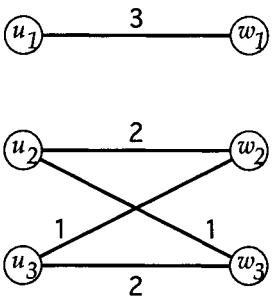


FIGURE 17 Undirected network N_B.

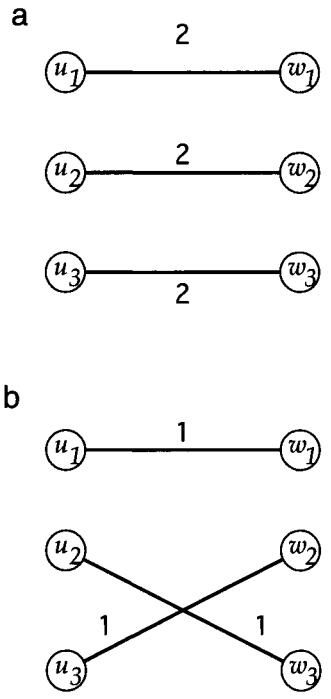


FIGURE 18 Networks (a) O_1 and (b) $N_{B0}(= O_2)$.

III. EDGE-δ-RELIABLE FLOW

Assumptions

1. The capacity of each edge is a positive real number.
2. The network is directed.
3. $0 < \delta \leq 1$.

Notation

$c_{e;\delta}\langle V_1, V_1^c\rangle$	δ-reliable capacity of $\langle V_1, V_1^c\rangle$
α	least integer upper bound of $(1/\delta)$ (liub$(1/\delta)$)
$\Psi(j)$	$[1-(j-1)\cdot\delta]^{-1}$

In this and the next sections we consider δ-reliable flows for the survival method. First, in this section, we consider the case of the edge version of δ-reliable flow.

A. Edge-δ-Reliable Flow

DEFINITION 18 (EDGE VERSION OF δ-RELIABLE FLOW). Let a flow $F = (G_F, c_F(*))$. If

$$c_F(e) \leq \delta f \qquad f \equiv \text{value of } F, \tag{9}$$

for each edge e of G_F, then F is an edge-δ-reliable flow.

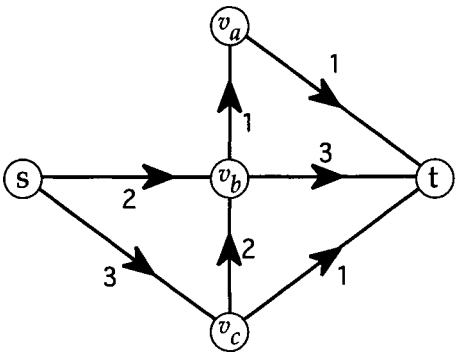


FIGURE 19 Edge-0.6-reliable flow F_e.

Network F_e of Fig. 19 is a flow with value 5 from s to t. Each "capacity of the edge in F_e" $\leq 3 = 0.6 \cdot 5$. Therefore, F_e is an edge-0.6-reliable flow.

A single edge-failure causes at most the fraction δ of an edge-δ-reliable flow to fail. Therefore, at least $(1 - \delta) \cdot$ (value of the flow) survives an edge failure. Even when n edges fail, at most $n \cdot \delta \cdot$ (value of an edge-δ-reliable flow) fails. The maximum value of an edge-δ-reliable flow between two vertices corresponds to the maximum communication capacity between two nodes by a δ-reliable channel.

B. δ-Reliable Capacity of a Cut

We define the capacity of a cut with respect to the δ-reliable flow.

DEFINITION 19 (CUT CAPACITY OF δ-RELIABLE FLOW). Let a cut-flow $F_c = (G_c, c_c(*))$ for $\langle V_1, V_1^c \rangle$ in N. If

$$c_c(e) \leq \delta \cdot f_c \tag{10}$$

for each edge e of G_c, then F_c is an edge-δ-reliable cut-flow of $\langle V_1, V_1^c \rangle$.

Let $V_1 = \{s, v_c\}$ in N_A of Fig. 7. A network F_2 of Fig. 20 is a cut-flow with value 5 of $\langle V_1, V_1^c \rangle$. Each "capacity of the edge in F_2" $\leq 3 = 0.6 \cdot 5$. Therefore, F_2 is an edge-0.6-reliable cut-flow of $\langle V_1, V_1^c \rangle$ in N_A.

DEFINITION 20. The edge-δ-reliable capacity of a cut is the maximum value of edge-δ-reliable cut-flows of the cut.

Since the edge-δ-reliable capacity is the value of a cut-flow, then $c_{e:\delta}\langle V_1, V_1^c \rangle \leq c\langle V_1, V_1^c \rangle$, where $c_{e:\delta}\langle V_1, V_1^c \rangle$ stands for the δ-reliable capacity of $\langle V_1, V_1^c \rangle$. Let F be an edge-δ-reliable flow from s to t in N. A subnetwork created with the edges of an s–t cut $\langle V_1, V_1^c \rangle$ of F is an edge-δ-reliable cut-flow of $\langle V_1, V_1^c \rangle$. Consequently, the "edge-δ-reliable flows from s to t" $\leq$ "edge-δ-reliable capacity of $\langle V_1, V_1^c \rangle$".

Using the δ-reliable capacity of a cut, we can specify the bottleneck for δ-reliable flow in a network.

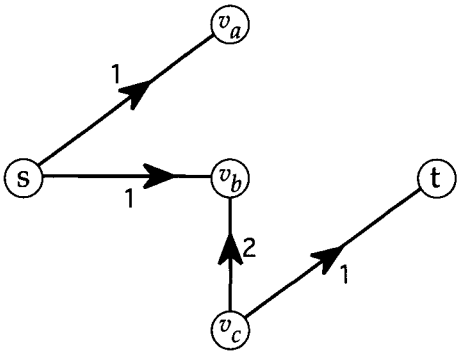


FIGURE 20 Edge-0.6-reliable cut-flow F_2.

Let the edges of cut $\langle V_1, V_1^c \rangle$ in $N = (G, c(*))$ be $e_1, e_2, \ldots, e_q$, such that

$$c\langle V_1, V_1^c \rangle = \sum_{j=1}^{q} c(e_j); \tag{11}$$

$$c(e_j) \geq c(e_{j+1}) \quad \text{for } 1 \leq j \leq q-1. \tag{12}$$

Let $F = (G', c'(*))$ be a network such that G' consists of edges of $\langle V_1, V_1^c \rangle$ in $N = (G, c(*))$. Then F is an edge-δ-reliable cut-flow of $\langle V_1, V_1^c \rangle$ if and only if

$$c'(e_j) \leq \min\{c(e_j), \delta\theta\}, \quad \text{for } j = 1, 2, \ldots, q. \tag{13}$$

A necessary and sufficient condition for a value θ to be the value of an edge-δ-reliable flow is

$$\theta \leq \sum_{k=1}^{q} \min\{c(e_k), \delta\theta\}. \tag{14}$$

The next method for obtaining the edge-δ-reliable capacity of a cut is described. Theorem 2 shows that Algorithm 5 is correct.

ALGORITHM 5 (EVALUATION OF EDGE-δ-RELIABLE CAPACITY OF A CUT).

1. The sequence θ_j, for $1 \leq j \leq \alpha$, is the θ sequence of $\langle V_1, V_1^c \rangle$ defined as

$$\theta_j = \Psi(j) \cdot \sum_{k=j}^{q} c(e_k) \tag{15}$$

$$= \Psi(j)\left\{ c\langle V_1, V_1^c \rangle - \sum_{k=1}^{j-1} c(e_k) \right\}. \tag{16}$$

2. τ is the minimum ℓ, for $1 \leq \ell \leq \alpha$, such that

$$c(e_\ell) \leq \delta \cdot \Psi(\ell) \cdot \sum_{k=\ell}^{q} c(e_k) = \delta \cdot \theta_\ell. \tag{17}$$

3. $\theta_\tau = c_{e:\delta}\langle V_1, V_1^c \rangle$, the edge-$\delta$-reliable capacity of $\langle V_1, V_1^c \rangle$.

(end of algorithm)

It can be shown that $\theta_\tau = 0$ if and only if $q < \alpha$. The $q < \alpha$ implies that a δ-reliable cut-flow requires that capacity be distributed over more edges than are in the cut, so the edge-δ-reliable cut-capacity must be zero. The next lemma shows a property of the θ sequence.

LEMMA 2. *Let $\tau (1 \leq \tau \leq \alpha)$ be the δ-index of cut $\langle V_1, V_1^c \rangle$.*

1. *For $\tau \leq j \leq \alpha$*

$$\theta_j \geq \theta_\tau, \tag{18}$$

2. *For $1 \leq j < \tau$,*

$$\theta_j > \theta_\tau. \tag{19}$$

Lemma 2 implies that the θ sequence is unimodal.

THEOREM 2. *θ_τ is the edge-δ-reliable capacity of $\langle V_1, V_1^c \rangle$.*

EXAMPLE 3. In N_A of Fig. 7, let $V_1 = \{s, v_c\}$. Using Algorithm 5, we will evaluate the edge-0.6-reliable capacity of $\langle V_1, V_1^c \rangle$.

1. Set $e_1 = (s, v_b)$, $e_2 = (v_c, v_b)$, $e_3 = (s, v_a)$, and $e_4 = (v_c, t)$. Then $c(e_1) = 8$, $c(e_2) = 2$, $c(e_3) = 1$, and $c(e_4) = 1$.
2. Since $1/\delta = 1.66\ldots$, set $\alpha = 2$, and θ sequence is calculated as follows:

$$\theta_1 = \Psi(1) \cdot \sum_{k=1}^{q} c(e_k) = 1 \cdot 12 = 12 > 8\ (= c(e_1)),$$

$$\theta_2 = \Psi(2) \cdot \sum_{k=2}^{q} c(e_k) = 2.5 \cdot 4 = 10 > 2\ (= c(e_2)),$$

3. Then

$$\delta\theta_1 = 0.6 \cdot 12 = 7.8 < 8 (= c(e_1)),$$

$$\delta\theta_2 = 0.6 \cdot 10 = 6 \geq 2 (= c(e_2)).$$

We get $\tau = 2$.

4. $c_{e:\delta}\langle V_1, V_1^c \rangle = \theta_2 = 10$.

C. Maximum Edge-δ-Reliable Flow

The next theorems show that we can specify the bottleneck for edge-δ-reliable flow by finding the cut with the lowest δ-reliable capacity.

THEOREM 3. *The maximum value of edge-δ-reliable flow from s to t is the minimum value of edge-δ-reliable capacities of s–t cuts.*

We show a method for evaluating the maximum value of edge-δ-reliable flows between a specified pair of vertices. The method consists of calculations of the maximum value of ordinary flows between those vertices.

ALGORITHM 6 (EVALUATION OF THE MAXIMUM VALUE OF EDGE-δ-RELIABLE FLOWS). Let s and t be the distinct vertices in a given network N, and $N_0 = N$. We will find the maximum value of edge-δ-reliable flows between s and t.

1. Evaluate the maximum value of the ordinary flows form s to t in N_0. Set μ_0 to be the value.

2. Set $\xi_1 = \mu_0$. Then repeat the following procedure for $k = 1, 2, \ldots$, in increasing order until the procedure stops.

(procedure)

Let N_k be the network obtained by decreasing the edge capacities greater than $\delta\xi_k$ in N to $\delta\xi_k$. Calculate the maximum value of the ordinary flows from s to t in N_k. Then set μ_k to be the value.

(a) If $\mu_k = \xi_k$, stop.
(b) If $\mu_k < \xi_k$, set

$$\xi_{k+1} = \frac{\mu_k - k\delta\xi_k}{1 - k\delta}. \tag{20}$$

(end of procedure)

When the procedure stops, μ_k is the maximum value of edge-δ-reliable flows from s to t. Then the minimum s–t cut in N_k is also the minimum s–t cut for edge-δ-reliable flows in N.

(end of algorithm)

THEOREM 4. *Algorithm 6 gives the maximum value of the edge-δ-reliable flow.*

EXAMPLE 4 (AN EXAMPLE OF OBTAINING THE EDGE-0.6-RELIABLE FLOW). Using Algorithm 6, we shall evaluate the maximum value of edge-0.6-reliable flows from s to t in the network N_A of Fig. 7.

(i) Let N_0 be N_A. Using Algorithm 1, evaluate the maximum value of the ordinary flows from s to t in N_0. Then $\mu_0 = 11$.

(ii) Set $\xi = 11$. We get N_1 of Fig. 21a by decreasing the capacities of edges in N_A so that each edge with a capacity larger than $6.6 (= \delta\xi_1 = 0.6 \cdot 11)$ in N_A has the capacity 6.6 in N_1. Evaluating the maximum value of the ordinary flows in N_1, we get $\mu_1 = 10.6$.

(iii) Since $\mu_1 < \xi_1$, set $\xi_2 = \frac{\mu_1 - 1 \cdot 0.6 \cdot \xi_1}{1 - 0.6} = 10$ and get N_2 of Fig. 21b by decreasing the capacities of edges in N_A so that each edge with a capacity larger than $6 (= \delta\xi_1 = 0.6 \cdot 10)$ in N_A has the capacity 6 in N_2. Evaluating the maximum value of the ordinary flows in N_2, we get $\mu_2 = 10$. Then $\mu_2 = \xi_2$, and the procedure stops.

The value $\mu_2 (= 10)$ is the maximum value of edge-0.6-reliable flows from vertex s to vertex t in N_A.

A flow F_3 of Fig. 22 is a maximum flow in N_2; therefore F_3 is a maximum edge-0.6-reliable flow from s to t in N_A. Since the minimum s–t cut in N_2 is a cut $\langle V_2, V_2^c \rangle$, where $V_2 = \{s\}$, $\langle V_2, V_2^c \rangle$ is also the minimum s–t cut for edge-0.6-reliable flows in N_A.

(end of example)

IV. VERTEX-δ-RELIABLE FLOW

In this section we describe the vertex version of δ-reliable flow.

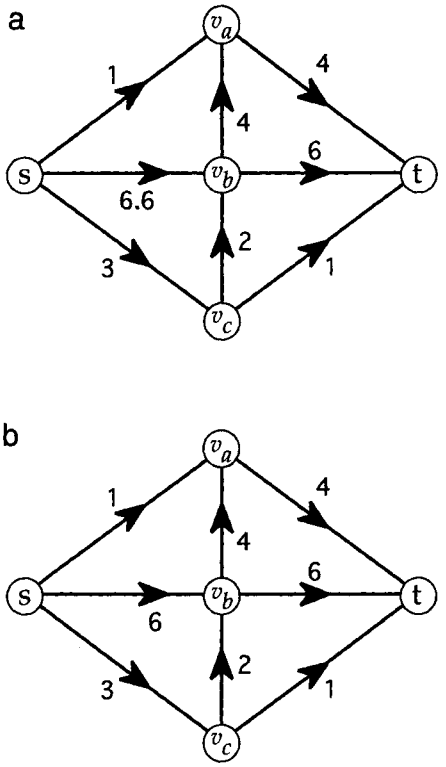


FIGURE 21 Networks (a) N_1 and (b) N_2.

A. Vertex-δ-Reliable Flow

DEFINITION 21. Let $F = (G_F, c_F(*))$ be a flow from a vertex s to a vertex t. If

$$c_F(v) \leq \delta f, \tag{21}$$

for each vertex v in F (except s and t), and

$$c_F(e) \leq \delta f, \tag{22}$$

for each edge e in F, F is the vertex-δ-reliable flow.

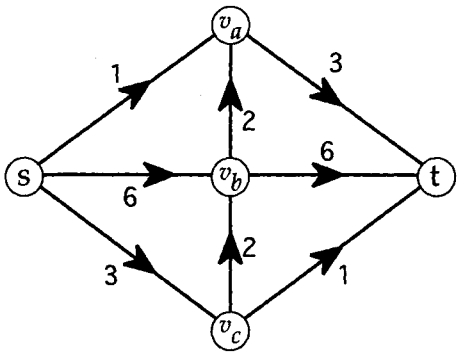


FIGURE 22 A flow F_3.

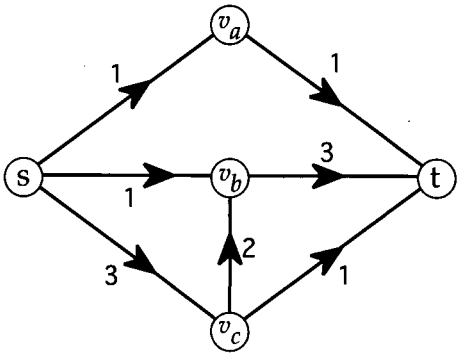


FIGURE 23 A flow F_4.

An edge failure or vertex failure causes at most the fraction δ of a vertex-δ-reliable flow to fail. Therefore, at least $(1-\delta)\cdot$(value of the flow) always survives the failure of an edge or vertex. The flow F_4 of Fig. 23 is an example of vertex-0.6-reliable flow. Each "capacity of the edge in F_4" $\leq 3(= 0.6 \cdot 5)$. The capacities of vertices v_a, v_b, v_c are 1, 3, 3. Each "capacity of the vertex in F_4 except s and t" ≤ 3. Therefore, F_4 is a vertex-0.6-reliable flow.

B. δ-Reliable Capacity of a Mixed Cut

Although the max-flow min-cut theorem of the edge-δ-reliable flow has been proved for ordinary cuts, the theorem for the vertex-δ-reliable flow is considered for mixed cuts.

DEFINITION 22. Let $F_c = (G_c, c_c(*))$ be a cut-flow of a mixed cut $\langle V_1, V_2\rangle$ in N. Let f_c be the value of F_c. If

$$c_c(e) \leq \delta \cdot f_c, \tag{23}$$

for each edge e; and if

$$c_c(v) \leq \delta \cdot f_c, \tag{24}$$

for each vertex v of $\langle V_1, V_2\rangle$, then F_c is a vertex-δ-reliable cut-flow of $\langle V_1, V_2\rangle$.

Let $V_1 = \{s, v_c\}$ and $V_1 = \{v_a, t\}$ in N_A of Fig. 7. A mixed cut $\langle V_1, V_2\rangle$ consists of edges (s, v_a), (v_c, t), and a vertex v_b. A network F_6 of Fig. 24 is a cut-flow with value 5 of $\langle V_1, V_2\rangle$. Each "capacity of the edge in F_6" $\leq 3(= 0.6 \cdot 5)$ and "the capacity of v_b in F_6" ≤ 3. Therefore, F_6 is an edge-0.6-reliable cut-flow of $\langle V_1, V_1^c\rangle$ in N_A.

DEFINITION 23. The vertex-δ-reliable capacity of a mixed cut is the maximum value of vertex-δ-reliable cut-flows of the mixed cut.

When a mixed cut $\langle V_1, V_2\rangle$ is an ordinary cut, then $c_{v:\delta}\langle V_1, V_2\rangle = c_{e:\delta}\langle V_1, V_2\rangle$, where $c_{v:\delta}\langle V_1, V_2\rangle$ stands for the vertex-δ-reliable capacity of $\langle V_1, V_2\rangle$. Let $F' = (G_f, c_f(*))$ be a subnetwork induced by the edges and the vertices of an s–t mixed cut $\langle V_1, V_2\rangle$ of a vertex-δ-reliable flow from s to t in

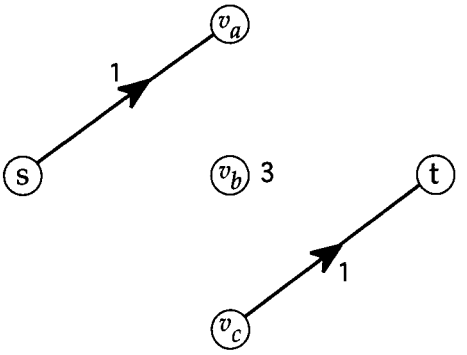


FIGURE 24 Vertex-0.6-reliable cut-flow F_6.

$N = (G, c(*))$. If $c_f = c$ for edges and vertices in F', then F' is a vertex-δ-reliable cut-flow of $\langle V_1, V_2 \rangle$. Consequently, the "maximum value of vertex-δ-reliable flows from s to t" $\leq$ "vertex-δ-reliable capacity of $\langle V_1, V_2 \rangle$".

The vertex-δ-reliable capacity of $\langle V_1, V_2 \rangle$ is evaluated in much the same way as the edge-δ-reliable capacity of a cut.

Let all edges of a mixed cut $\langle V_1, V_2 \rangle$ in $N = (G, c(*))$ be $e_1, e_2, \ldots, e_p$; let all vertices of that $\langle V_1, V_2 \rangle$ be $v_1, v_2, \ldots, v_q$. Then set $a_1, a_2, \ldots, a_{p+q}$ to be a series of the values of $c(e_1), c(e_2), \ldots, c(e_p)$ and $c(v_1), c(v_2), \ldots, c(v_q)$, where there is a one-to-one correspondence between $\{a_1, a_2, \ldots, a_{p+q}\}$ and $\{c(e_1), c(e_2), \ldots, c(e_p)\} \cup \{c(v_1), c(v_2), \ldots, c(v_q)\}$ such that

$$c\langle V_1, V_2 \rangle = \sum_{j=1}^{p+q} a_j, \tag{25}$$

$$a_j \geq a_{j+1} (1 \leq j \leq p+q-1). \tag{26}$$

ALGORITHM 7. We will evaluate the vertex-δ-reliable capacity of a mixed cut $\langle V_1, V_2 \rangle$.

1. The θ sequence of $\langle V_1, V_2 \rangle$ is

$$\theta_j^{(v)} = \Psi(j) \cdot \sum_{k=j}^{p+q} a_k$$

$$= \Psi(j) \left\{ c\langle V_1, V_2 \rangle - \sum_{k=1}^{j-1} a_k \right\}. \tag{27}$$

2. Define δ-index τ as the minimum of ℓ, $1 \leq \ell \leq \alpha$, such that

$$a_\ell \leq \delta \cdot \Psi(\ell) \cdot \sum_{k=\ell}^{p+q} a_k = \delta \cdot \theta_\ell^{(v)}. \tag{28}$$

3. Then $\theta_\tau^{(v)} = c_{v:\delta}\langle V_1, V_2 \rangle$, the vertex-$\delta$-reliable capacity of $\langle V_1, V_2 \rangle$.

(end of algorithm)

C. Maximum Vertex-δ-Reliable Flow

Using the next transformation, vertex-δ-reliable flows in a network are reduced to edge-δ-reliable flows in the obtained network.

TRANSFORMATION 1. For vertices s and t, we will transform a network $N = (G, c(*))$ to a network $N^{(v)}$ by replacing each vertex $v_j (v_j \neq s, t)$ in N to two vertices $v_j^{(0)}$ and $v_j^{(i)}$ as follows:

1. Delete v_j.
2. All edges that exit from v_j are transferred to exit from a new vertex $v_j^{(o)}$.
3. All edges that go into v_j are transferred to go into a new vertex $v_j^{(i)}$.
4. Connect $v_j^{(o)}$ and $v_j^{(i)}$ with the e_j that exits from $v_j^{(i)}$ and goes into $v_j^{(o)}$. Set $c(e_j) = c(v_j)$. The e_j is the bridge edge of v_j.

(end of transformation)

An edge-δ-reliable flow in $N^{(v)}$ becomes a vertex-δ-reliable flow in the original N by contracting each bridge edge between $v_j^{(o)}$ and $v_j^{(i)}$ to the vertex v_j. The "vertex-δ-reliable capacity of an s–t mixed cut $\langle V_1, V_2 \rangle$ in N" = "edge-δ-reliable capacity of the corresponding cut in $N^{(v)}$". Since in $N^{(v)}$ the max-flow min-cut theorem holds for the edge-δ-reliable flow, the max-flow min-cut theorem of vertex-δ-reliable flow can be obtained easily.

THEOREM 5. *The maximum value of vertex-δ-reliable flows from a vertex s to a vertex t in a network equals the minimum value of vertex-δ-reliable capacities of s–t mixed cuts.*

Since there is a one-to-one correspondence between a vertex-δ-reliable flow in N and an edge-δ-reliable flow in $N^{(v)}$, as long as they have the same value we get the next algorithm for obtaining the maximum vertex-δ-reliable flow in a network.

ALGORITHM 8 (EVALUATION OF THE MAXIMUM VALUE OF VERTEX-δ-RELIABLE FLOWS). Let s and t be the distinct vertices in a given network N. We will find the maximum value of edge-δ-reliable flows between s and t.

1. Using Transformation 1 for s and t, transform N to $N^{(v)}$.
2. Using Algorithm 6 find the maximum edge-δ-reliable flow F_e and the minimum s–t cut $\langle V_1, V_1^c \rangle$ for edge-δ-reliable flow in $N^{(v)}$.
3. Contracting each bridge edge between $v_j^{(i)}$ and $v_j^{(o)}$ to the vertex v_j in F_e we get the maximum vertex-δ-reliable flow F_v in N.
4. Contracting each bridge edge between $v_j^{(i)}$ and $v_j^{(o)}$ in $\langle V_1, V_1^c \rangle$ to the vertex v_j we transform $\langle V_1, V_1^c \rangle$ to a set of edges and vertices that is the minimum

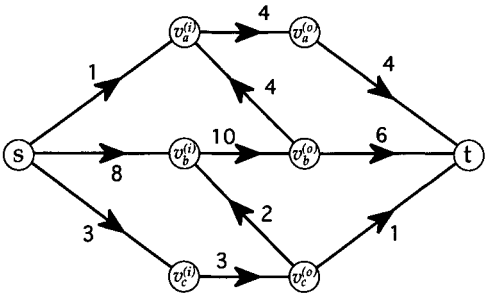


FIGURE 25 Transformed Network $N_A^{(v)}$.

s–t mixed cut for vertex-δ-reliable flow in N. Then we can select the vertex sets V_1 and V_2 that represent the above set of edges and vertices as $\langle V_1, V_2 \rangle$ in N.

(end of algorithm)

THEOREM 6. *Algorithm 8 gives the maximum value of the vertex-δ-reliable flow.*

EXAMPLE 5 (AN EXAMPLE OF OBTAINING THE VERTEX-0.6-RELIABLE FLOW). Using Algorithm 8, we shall evaluate the maximum value of edge-0.6-reliable flows from s to t in the network N_A of Fig. 7.

1. Using Transformation 1 for s and t, transform N_A to $N_A^{(v)}$ of Fig. 25.
2. Using Algorithm 6, we get the maximum edge-0.6-reliable flow F_e of Fig. 26 with value 5. Then we also find the minimum s–t cut $\langle V_1, V_1^c \rangle$ for edge-0.6-reliable flow in $N_A^{(v)}$, where $V_1 = \{s, v_b^{(i)}, v_c^{(i)}, v_c^{(o)}\}$.
3. Contracting each bridge edge between $v_j^{(o)}$ and $v_j^{(i)}$ to the vertex v_j in F_e and $\langle V_1, V_1^c \rangle$, we get the maximum vertex-δ-reliable flow F_v of Fig. 27 and the minimum s–t cut for vertex-δ-reliable flow $\langle V_1, V_2 \rangle$ in N_A, where $V_1 = \{s, v_c\}$, $V_2 = \{v_a, t\}$.

(end of example)

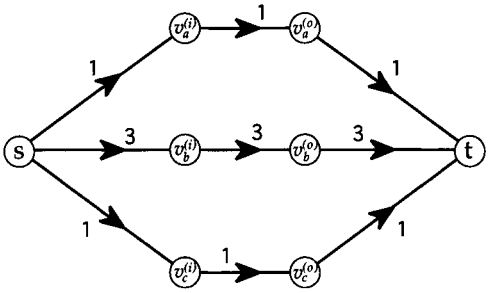


FIGURE 26 Maximum edge-0.6-reliable flow F_e.

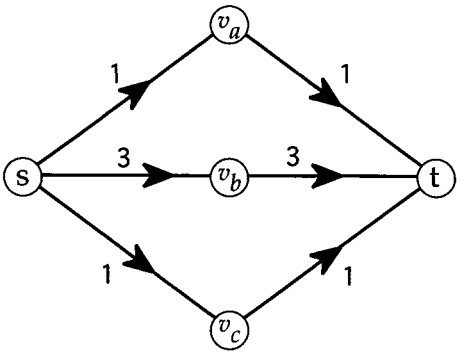


FIGURE 27 Maximum vertex-0.6-reliable flow F_v.

V. *m*-ROUTE FLOW

In this section we consider m-route flows for the restoral method. First, we consider the case of the edge version of m-route flow. Then we consider the case of the vertex version of m-route flow.

A. Edge-*m*-Route Flow

DEFINITION 24 [25]. An elementary edge-m-route flow from s to t is defined as a subnetwork of N, which is the sum of m edge-disjoint s–t path-flows, each of which has the same value μ. "Path flows $P_i = (G_i, c_i(*))$ and $P_j = (G_j, c_j(*))$ are edge-disjoint" means that G_i and G_j are edge-disjoint paths from s to t. The value of the elementary edge-m-route flow is defined as $m \cdot \mu$.

DEFINITION 25 [25]. An edge-m-route flow from s to t is a subnetwork of N such that the subnetwork is the sum of elementary edge-m-route flows from s to t. The sum of values of those elementary edge-m-route flows is called the value of the edge-m-route flow.

Both the networks of m_1 of Fig. 28a and m_2 of Fig. 28b are elementary edge-2-route flows from s to t. The network M_0 of Fig. 29 is the edge-2-route flow that is the sum of m_1 and m_2. Both values of m_1 and m_2 are 2. Therefore, the value of M_0 is 4.

The maximum value of edge-m-route flow between two vertices corresponds to the maximum capacity of m-route communication channels between two terminals.

DEFINITION 26 [25]. If $\delta = 1/m$ (m is a positive integer) we say an edge-δ-reliable flow is an edge-m-leveled flow, and the edge-δ-reliable capacity of a cut is the edge-m-route capacity of the cut.

Obviously, an edge-m-route flow is an edge-m-leveled flow.

Since an edge-m-leveled flow and the edge-m-route capacity of a cut are an edge-δ-reliable flow and the edge-δ-reliable capacity of a cut, where $\delta = 1/m$, the next theorem is easily followed.

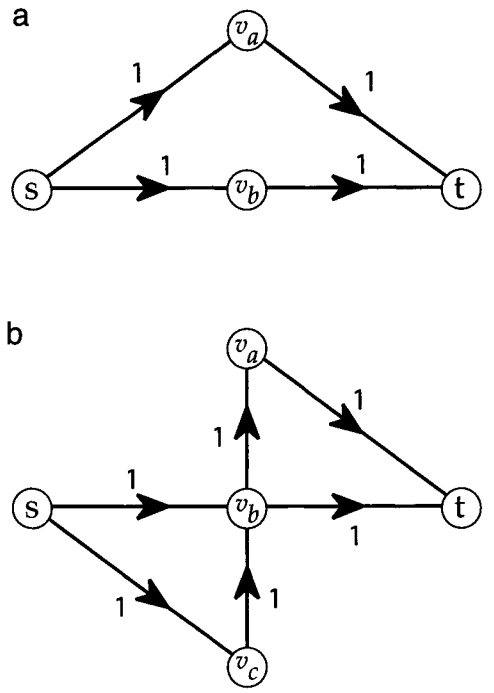


FIGURE 28 Elementary edge-2-route flows (a) m_1 and (b) m_2.

THEOREM 7 [25]. *The maximum value of edge-m-leveled flows from s to t is equal to the minimum value of edge-m-route capacities of s–t cuts.*

B. Maximum Edge-*m*-Route Flow

Edge-m-leveled flows and edge-m-route flows are related.

THEOREM 8 [25]. *An edge-m-leveled flow has a certain value, if and only if an edge-m-route flow has the same value.*

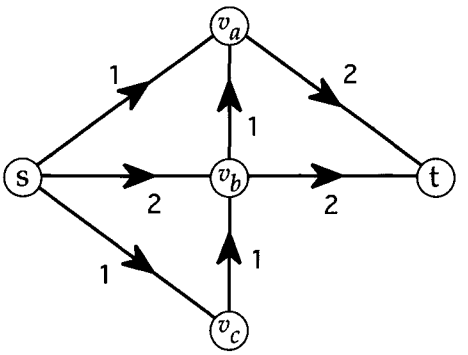


FIGURE 29 An edge-2-route flow M_0.

Theorem 8 means that if there is an edge-m-leveled flow with a certain value, then there must be an edge-m-route flow with the same value as a subnetwork of the edge-m-leveled flow.

The next theorem is a corollary of Theorems 7 and 8.

THEOREM 9 [25]. *In a network the maximum value of edge-m-route flows from s to t is equal to the minimum value of edge-m-route capacities of s–t cuts.*

On the basis of these theorems, Ref. [25] shows a method for finding the maximum edge-m-route flow between a given pair of vertices in a network. The method finds the maximum value of edge-m-route flows and the minimum cut by using a method for maximum edge-δ-reliable flow. Then it gives a maximum edge-m-route flow and represents the flow as the sum of elementary edge-m-route flows.

ALGORITHM 9 [22, 23, 25]. Let s and t be the distinct vertices in a given network N. We will find a maximum edge-m-route flow between s and t.

1. Using Algorithm 6 with $\delta = 1/m$, obtain the maximum edge-m-leveled flow between s and t, and find the minimum s–t cut for the edge-m-route flow. Let the maximum flow be $M_0 = (G_0, c_0(*))$ with the value λ.

2. Since Algorithm 6 uses Algorithm 1 for finding the maximum m-leveled flow and Algorithm 1 represents the flow as the directed sum of s–t path-flows, M_0 can be represented as the sum of s–t path flows $P_1, \ldots, P_q$, easily.

3. On the basis of $M_0 = (G_0, c_0(*))$, we define an undirected network $N_F = (G_F, c_F(*))$ [22]. Let the edge set of G_0 be $E_0 = \{e_1, e_2, \ldots, e_r\}$. The vertex set V_F of G_F is defined as

$$U = \{s, u_1, u_2, \ldots, u_r\}, \tag{29}$$

$$W = \{t, w_1, w_2, \ldots, w_r\}, \tag{30}$$

$$V_F = U \cup W, \tag{31}$$

where both u_i and w_i correspond to the edge e_i in M_0. The edge set E_F of G_F consists of edges between a vertex in U and a vertex in W. We write the capacity of edge (x, y) as $c_F(x, y)$. The capacity of each edge is defined as

$c_F(u_i, W_j)$ = the sum of values of the path-flows that pass through edge e_j just after passing through edge e_i among $P_1, \ldots, P_q$,

$c_F(s, W_j)$ = the sum of values of the path-flows that pass through edge e_j directly from vertex s among $P_1, \ldots, P_q$, and

$c_F(u_i, t)$ = the sum of values of the path-flows that end at vertex t just after passing through edge e_i among $P_1, \ldots, P_q$.

The edges assigned the capacity of 0 are removed.

4. (a) For each pair of vertices u_i and w_i in N_F, $c_F(I(w_i)) = c_F(I(u_i))$. Let the value be p_i, and then $p_i \leq \frac{\lambda}{m}$. Add edges (u_i, w_i) $(1 \leq i \leq r)$ with the capacity of $\frac{\lambda}{m} - p_i$ to N_F only if p_i is less than $\frac{\lambda}{m}$.

(b) Then replace vertex s by m vertices $\{s_1, s_2, \ldots, s_m\}$ and t by m vertices $\{t_1, t_2, \ldots, t_m\}$.

(c) Let the edges incident to the vertex s be $(s, w_{j_1}), (s, w_{j_2}), \ldots, (s, w_{j_x})$. We will divide those edges and connect them to $s_1, s_2, \ldots, s_m$ such that the capacity of the incidence cut of each s_i is equal to $\frac{\lambda}{m}$ as follows.

For vertex s_1 let d be the minimum integer such that

$$\sum_{i=1}^{d} c_F(s, w_{j_i}) \geq \frac{\lambda}{m}. \tag{32}$$

Then transform $(s, w_{j_1}), (s, w_{j_2}), \ldots, (s, w_{j_{d-1}})$, into $(s_1, w_{j_1}), (s_1, w_{j_2}) \ldots, (s_1, w_{j_{d-1}})$, and divide (s, w_{j_d}) into $(s_1, w_{j_d}), (s_2, w_{j_d})$. Set $c_F(s_1, w_{j_d})$ to be the value to hold the equality of Eq. (32) and

$$c_F(s_2, w_{j_d}) = \frac{\lambda}{m} - c_F(s_1, w_{j_d}). \tag{33}$$

For $s_2, s_3, \ldots, s_m$ proceed in the same way.

(d) The edges incident to the vertex t are divided in the same way.

(e) Let $N_U = (G_U, c_U(*))$ be the resulting undirected bipartite network, where $G_U = (X_U, Y_U; E_U)$.

5. Let

$$X_U = (U - \{a\}) \cup \{a_1, a_2, \ldots, a_m\}, \tag{34}$$

$$Y_U = (W - \{b\}) \cup \{b_1, b_2, \ldots, b_m\}. \tag{35}$$

Then N_U is bipartite and even of value $\frac{\lambda}{m}$. By using Algorithm 4, we can represent N_U as the sum of 1-factor networks $N^{(1)}, N^{(2)}, \ldots, N^{(h)}$. Then $\sum_{i=1}^{h} \nu_i = \frac{\lambda}{m}$ where ν_i is the value of $N^{(i)}$.

6. From the edge set $E^{(i)}$ of $N^{(i)}$, obtain the m disjoint edge sets $E_1^{(i)}$, $E_2^{(i)}, \ldots, E_m^{(i)}$, as follows.

(Operation to determine $E_j^{(i)}$)

As the initial condition, set $E_j^{(i)} = \emptyset$. First, add (s_j, w_{j_1}), the edge incident to s_j in $N^{(i)}$, to $E_j^{(i)}$. Then w_{j_1} is none of $t_1, t_2, \ldots, t_m$. Next, for $k = 1, 2, \ldots$, iterate the next procedure until $w_{j_{k+1}}$ is either of $t_1, \ldots, t_m$.

(procedure)

Noticing that both w_{j_k} and u_{j_k} correspond to e_{j_k}, add $(u_{j_k}, w_{j_{k+1}})$, the edge incident with u_{j_k} in $N^{(i)}$, to $E_j^{(i)}$.

(end of procedure)

(end of operation)

7. Let the edge set $E^{(i)'}$ be the union of $E_1^{(i)}, E_2^{(i)}, \ldots, E_m^{(i)}$. Then, in N let $N_\alpha^{(i)}$ be the subnetwork induced by the edges, of which each corresponds to a vertex w_j incident to an edge in $E^{(i)'}$. The capacity of e_j is equal to the value of $N^{(i)}$, that is, ν_i. Each edge set $E_j^{(i)} (j = 1, \ldots, m)$ is of the form

$$E_j^{(i)} = \{(s_j, w_{j_1}), (u_{j_1}, w_{j_2}), \ldots, (u_{j_{y-1}}, w_{j_y}), (u_{j_y}, t_{z_j})\}, \tag{36}$$

where $\bigcup_{j=1}^{m} \{t_{z_j}\} = \{t_1, t_2, \ldots, t_m\}$. Hence, each $E_j^{(i)}$ corresponds to a path-flow with the value ν_i from s to t in $N_\alpha^{(i)}$. Since $E_j^{(i)}$ are pair-wise edge-disjoint, $N_\alpha^{(i)}$ is an elementary edge-m-route flow with the value of $m \cdot \nu_i$ from s to t in N.

8. Let N_α be the sum of those elementary edge-m-route flows $N_\alpha^{(1)}, N_\alpha^{(2)}, \ldots, N_\alpha^{(h)}$. Consequently, N_α is an edge-m-route flow with the value λ from s to t in N.

(end of algorithm)

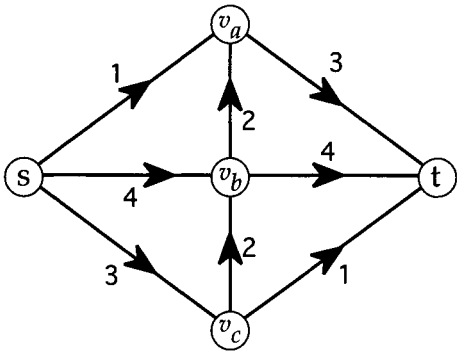


FIGURE 30 An edge-2-leveled flow M_0.

EXAMPLE 6 (AN EXAMPLE OF OBTAINING THE MAXIMUM EDGE-2-ROUTE FLOW). Using Algorithm 9, we shall find a maximum edge-2-route flow from s to t in the network N_A of fig. 7.

1. Using Algorithm 6 with $\delta = \frac{1}{2}$, obtain the maximum edge-m-leveled flow $M_0 = (G_0, c_0(*))$ with the value 8 of Fig. 30, and find the minimum s–t cut $\langle V_1, V_1^c \rangle$, where $V_1 = \{s\}$.

2. The maximum edge-m-route flow M_0 can be represented as the sum of s–t path flows P_1, P_2, P_3, and P_4 of Fig. 31.

3. On the basis of $N_A = (G_A, c_A(*))$, we define an undirected network $N_F = (G_F, c_F(*))$. Let the edges of G_A be as follows, $e_1 = (s, v_a)$, $e_2 = (s, v_b)$, $e_3 = (s, v_c)$, $e_4 = (v_b, v_a)$, $e_5 = (v_c, v_b)$, $e_6 = (v_a, t)$, $e_7 = (v_b, t)$, and $e_8 = (v_c, t)$.

The vertex set V_F of G_F is defined as

$$U = \{s, u_1, u_2, \ldots, u_8\}, \tag{37}$$

$$W = \{t, w_1, w_2, \ldots, w_8\}, \tag{38}$$

$$V_F = U \cup W. \tag{39}$$

Then the capacity of each edge in N_F is represented in Fig. 32.

4. (a) For each pair of vertices u_i and w_i in N_F, if $p_i < \frac{\lambda}{2}$ $(= 4)$, add edges $(u_i, w_i)(1 \leq i \leq 8)$ with the capacity of $4 - p_i$ to N_F.

(b) Then replace vertex s by 2 vertices $\{s_1, s_2\}$ and t by 2 vertices $\{t_1, t_2\}$.

(c) Divide those edges and connect them to s_1, s_2 such that the capacity of the incidence cut of s_1 and s_2 is equal to $\frac{\lambda}{2}$ $(= 4)$ each.

(d) The edges incident to the vertex t are divided in the same way.

(e) The resulting undirected bipartite network is $N_U = (G_U, c_U, (*))$ of Fig. 33, where $G_U = (X_U, Y_U; E_U)$.

5. Using Algorithm 4, N_U is represented as the sum of three 1-factor networks $N^{(1)}$ with value 1 of Fig. 34a, $N^{(2)}$ with value 1 of Fig. 34b, and $N^{(3)}$ with value 2 of Fig. 34c. All the edge capacities of Fig. 34a and those of Fig. 34b are 1, and those of Fig. 34c are 2.

6. From the edge-sets of $N^{(1)}$, $N^{(2)}$, and $N^{(3)}$, we get

$$E_1^{(1)} = \{(s_1, w_1), (u_1, w_6), (u_6, t_1)\}, \tag{40}$$

$$E_2^{(1)} = \{(s_2, w_2), (u_2, w_7), (u_7, t_2)\}, \tag{41}$$

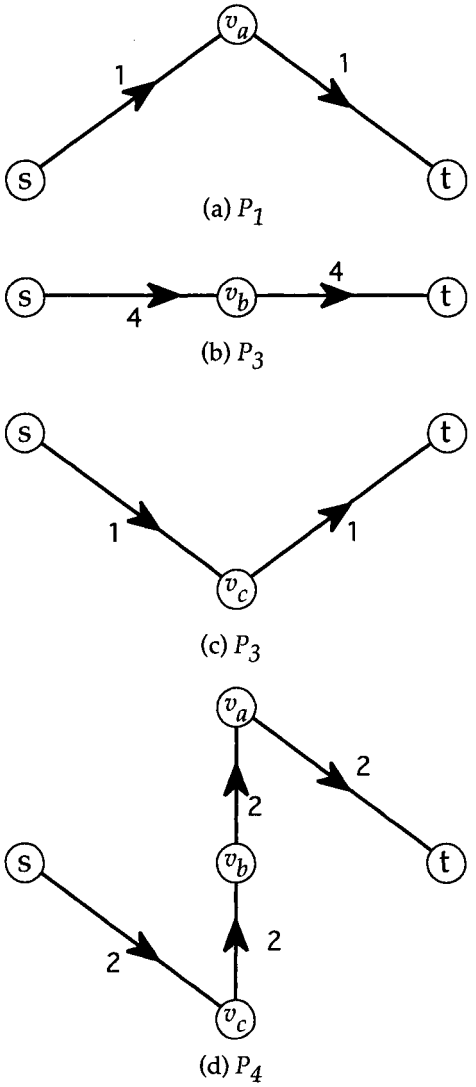


FIGURE 31 Path flows (a) P_1, (b) P_2, (c) P_3, and (d) P_4.

$$E_1^{(2)} = \{(s_1, w_2), (u_2, w_7), (u_7, t_1)\}, \tag{42}$$

$$E_2^{(2)} = \{(s_2, w_3), (u_3, w_8), (u_8, t_2)\}, \tag{43}$$

$$E_1^{(3)} = \{(s_1, w_2), (u_2, w_7), (u_7, t_2)\}, \tag{44}$$

$$E_2^{(3)} = \{(s_2, w_3), (u_3, w_5), (u_5, w_4), (u_4, w_6), (u_6, t_1)\}. \tag{45}$$

7. Let $E^{(1)'} = E_1^{(1)} \cup E_2^{(1)}$, $E^{(2)'} = E_1^{(2)} \cup E_2^{(2)}$, and $E^{(3)'} = E_1^{(3)} \cup E_2^{(3)}$. From three edge sets corresponding to $E^{(1)'}$, $E^{(2)'}$, and $E^{(3)'}$, we construct three elementary edge-2-route flows $N_\alpha^{(1)}$ of Fig. 35a, $N_\alpha^{(2)}$ of Fig. 35b, and $N_\alpha^{(3)}$ of Fig. 35c.

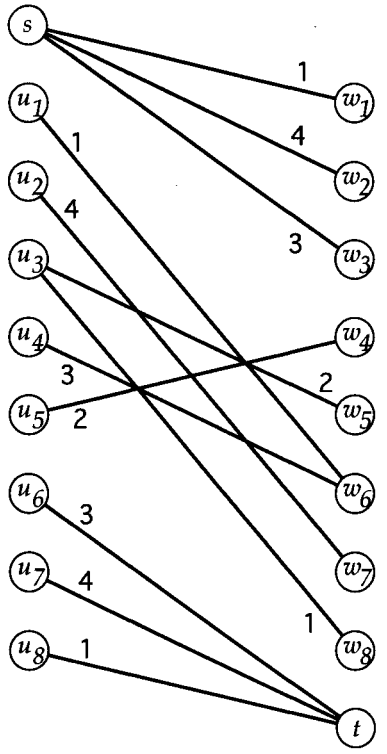


FIGURE 32 Undirected network N_F.

8. Let N_α be the sum of $N_\alpha^{(1)}$, $N_\alpha^{(2)}$, and $N_\alpha^{(3)}$. N_α (Fig. 36) is an edge-2-route flow with the value of 8 from s to t. (In this example N_α is the same as M_0.)

(end of example)

C. Vertex-m-Route Flow

We consider the case of the vertex version of m-route flow. From our viewpoint, the definitions for vertex-m-route flows are easy to follow.

DEFINITION 27 [25]. An elementary vertex-m-route flow from s to t is defined as a subnetwork of N, which is the sum of m internally disjoint s–t path-flows, each of which has the same value μ. "Path flows $P_i = (G_i, c_i(*))$ and $P_j = (G_j, c_j(*))$ are internally disjoint" means that G_i and G_j are internally disjoint paths from s to t. The value of the elementary vertex-m-route flow is defined as $m \cdot \mu$.

DEFINITION 28 [25]. A vertex-m-route flow from s to t is a subnetwork of N such that the subnetwork is the sum of elementary vertex-m-route flows from s to t. The sum of values of those elementary vertex-m-route flows is called the value of the vertex-m-route flow.

Both the networks of m_1 of Fig. 37a and m_2 of Fig. 37b are elementary vertex-2-route flows from s to t. The network M_1 of Fig. 38 is a vertex-2-route

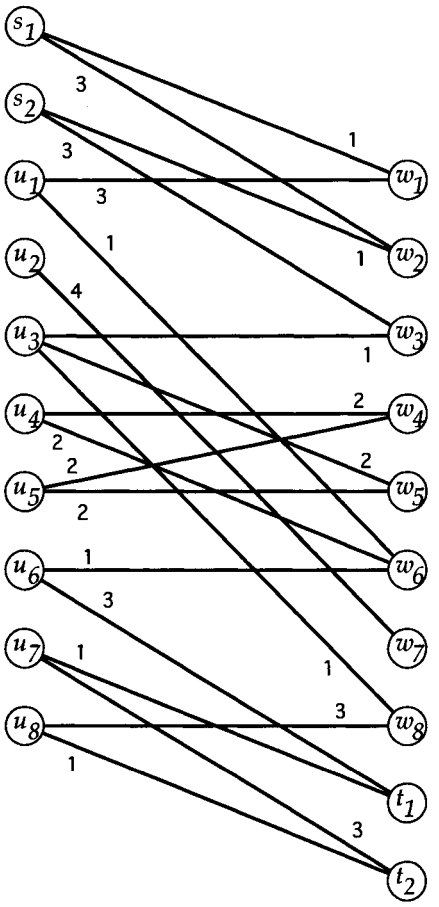


FIGURE 33 Undirected bipartite network N_U.

flow that is the sum of m_1 and m_2. Both values of m_1 and m_2 are 2. Therefore, the value of M_1 is 4.

DEFINITION 29 [25]. If $\delta = 1/m$ (m is a positive integer), we say a vertex-δ-reliable flow is a vertex-m-leveled flow, and the vertex-δ-reliable capacity of a cut is the vertex-m-route capacity of the cut.

Obviously, a vertex-m-route flow is a vertex-m-leveled flow. Like the theorem of edge-m-route flow, the next theorem is easy to follow.

THEOREM 10 [25]. *The maximum value of vertex-m-leveled flows from s to t is equal to the minimum value of vertex-m-route capacities of s–t cuts.*

D. Maximum Vertex-m-Route Flow

Vertex-m-leveled flows and vertex-m-route flows are related.

THEOREM 11 [25]. *A vertex-m-leveled flow has a certain value, if and only if a vertex-m-route flow has the same value.*

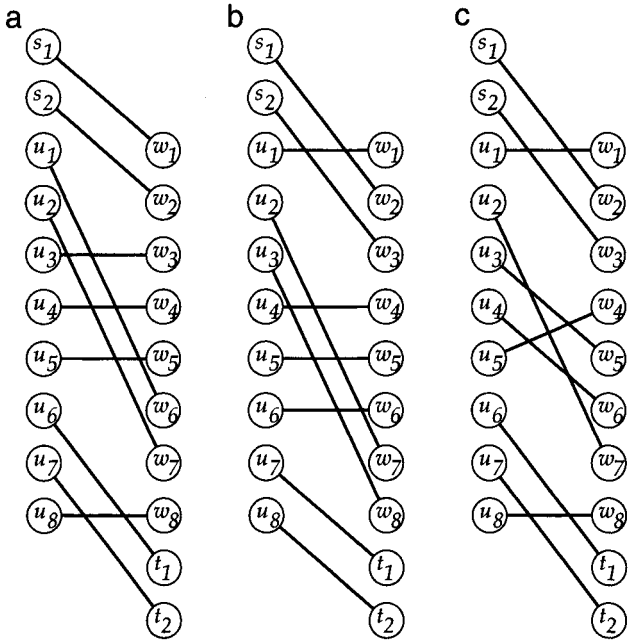


FIGURE 34 1-Factor networks (a) $N^{(1)}$, (b) $N^{(2)}$, and (c) $N^{(3)}$.

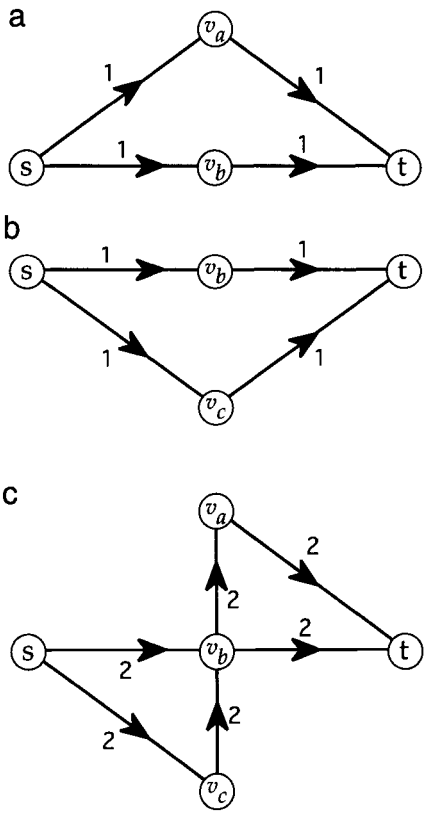


FIGURE 35 Elementary edge-2-route flows (a) $N_\alpha^{(1)}$, (b) $N_\alpha^{(2)}$, and (c) $N_\alpha^{(3)}$.

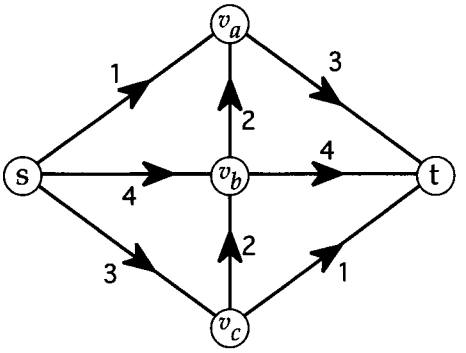


FIGURE 36 An edge-2-route flow N_α.

Theorem 11 means that if there is a vertex-m-leveled flow with a certain value, then there must be a vertex-m-route flow with the same value as a subnetwork of the vertex-m-leveled flow.

Then the next theorem holds.

THEOREM 12 [25]. *In a network the maximum value of vertex-m-route flows from s to t is equal to the minimum value of vertex-m-route capacities of s–t cuts.*

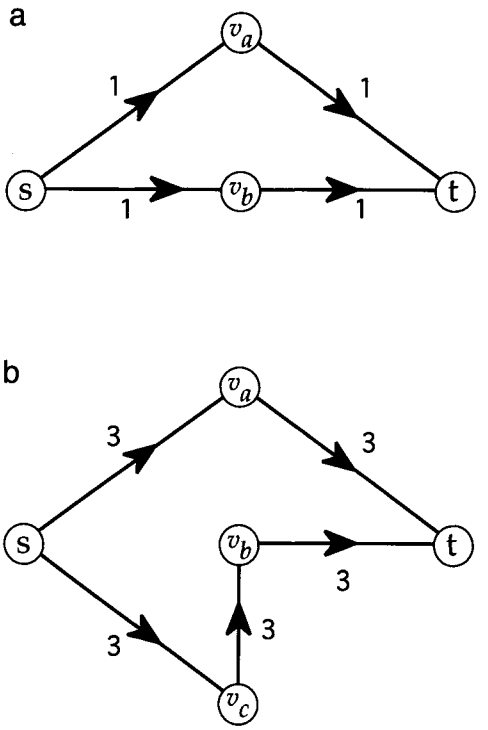


FIGURE 37 Elementary vertex-2-route flows (a) m_1 and (b) m_2.

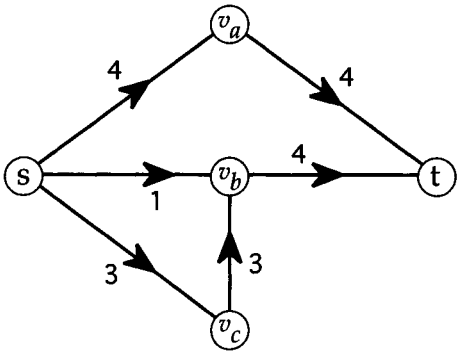


FIGURE 38 Vertex-2-route flow M_1.

Using Algorithm 9 with Transformation 1, we can get a method for finding the maximum vertex-m-route flow between a given pair of vertices in a network [25].

ALGORITHM 10. Let s and t be the distinct vertices in a given network N. We will find a maximum vertex-m-route flow between s and t.

1. Using Transformation 1 for s and t, transform N to $N^{(v)}$.
2. Using Algorithm 9 find the maximum edge-m-route flow F_e and the minimum s–t cut $\langle V_1, V_1^c \rangle$ for edge-m-route flow in $N^{(v)}$, and represent the edge-m-route flow as the sum of elementary edge-m-route flows, $m_1, m_2, \ldots, m_h$.
3. Contracting each bridge edge between $v_j^{(i)}$ and $v_j^{(o)}$ to the vertex v_j in F_e, and $m_1, m_2, \ldots, m_h$ we get the maximum vertex-m-route flow F_v, which is the sum of elementary vertex m-route flows $M_1, M_2, \ldots, M_h$ in N.
4. Contracting each bridge edge between $v_j^{(i)}$ and $v_j^{(o)}$ in $\langle V_1, V_1^c \rangle$ to the vertex v_j we transform $\langle V_1, V_1^c \rangle$ to a set of edges and vertices, which is the minimum s–t mixed cut for vertex-δ-reliable flow in N. Then we can select the vertex sets V_1 and V_2 that represent the above set of edges and vertices as $\langle V_1, V_2 \rangle$ in N.

(end of algorithm)

EXAMPLE 7 (AN EXAMPLE OF OBTAINING THE VERTEX-0.6-RELIABLE FLOW). Using Algorithm 10, we shall obtain the maximum vertex-2-route flow from s to t in the network N_A of Fig. 7.

1. Using Transformation 1 for s and t, transform N_A to $N_A^{(v)}$ of Fig. 25.
2. Using Algorithm 9 we find the maximum edge-2-route flow F_e of Fig. 39 with value 4, and the minimum s–t cut $\langle V_1, V_1^c \rangle$ for edge-2-route flow in $N_A^{(v)}$, where $V_1 = \{s, v_b^{(i)}, v_c^{(i)}, v_c^{(o)}\}$. The maximum 2-route flow F_e is represented as the sum of elementary edge-2-route flows, m_1 and m_2 of Fig. 40.
3. Contracting each bridge edge between $v_j^{(i)}$ and $v_j^{(o)}$ to the vertex v_j in F_e, m_1, and m_2, we get the maximum vertex-2-route flow F_v of Fig. 41, which is the sum of elementary vertex 2-route flows M_1 and M_2 of Fig. 42 in N_A.

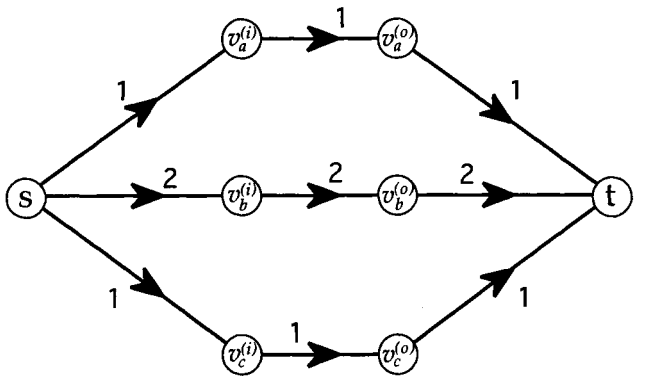


FIGURE 39 Maximum edge-2-leveled flow F_e.

4. Contracting each bridge edge between $v_j^{(i)}$ and $v_j^{(o)}$ in $\langle V_1, V_1^c \rangle$ to the vertex v_j we transform $\langle V_1, V_1^c \rangle$ into the minimum s–t mixed cut $\langle V_1, V_2 \rangle$ for vertex-δ-reliable flow in N_A, where $V_1 = \{s, v_c\}$, $V_2 = \{v_a, t\}$.

(end of example)

VI. SUMMARY

Network failures cannot be avoided entirely, so we need to design a system that will survive network failures. In this chapter, using a flow network model, we

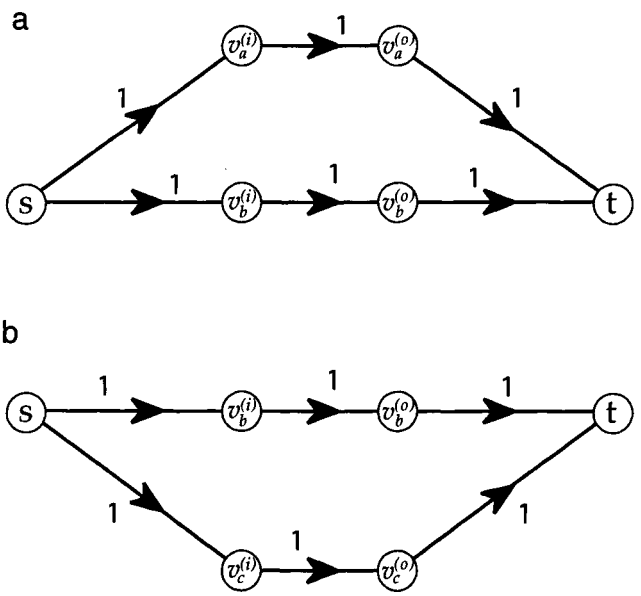


FIGURE 40 Elementary edge-2-route flows: (a) m_1 and (b) m_2.

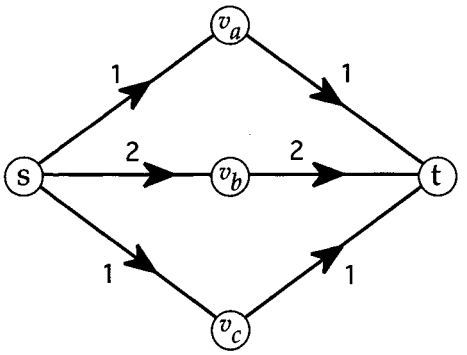


FIGURE 41 Maximum vertex-2-route flow F_v.

considered the design of a reliable communication channel with well-specified network survivability performance.

There are two approaches for constructing reliable communication channels: stochastic and deterministic. Here, we mainly considered the deterministic approach, in which the performance in the event of link or node failure is designed to exceed a predetermined level. We classified this approach as a survival method and a restoral method.

The survival method is for confirming the deterministic lower bound of the channel capacity in the event of link or node failure, while the restoral method is for reserving a restoral channel. For each of the two methods, we considered the problem of determining whether the required channels exist

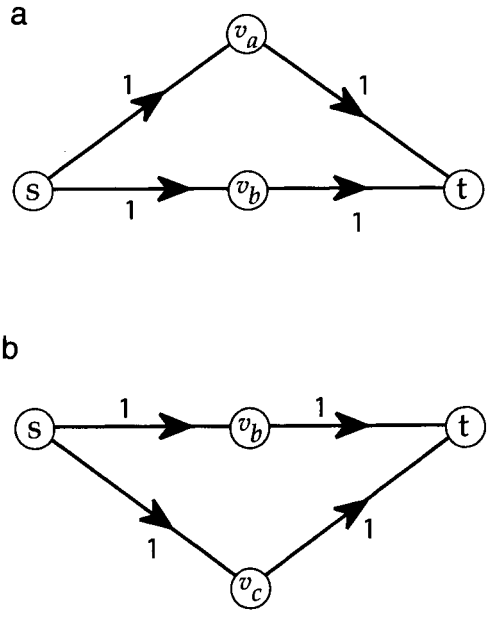


FIGURE 42 Elementary vertex-2-route flows: (a) M_1 and (b) M_2.

with given network resources. Moreover, when such channels do not exist, we determine the locations of the bottlenecks so that we can improve the network resources efficiently. We considered these problems as reliable-flow problems in a network.

REFERENCES

1. Zolfaghari, A., and Kaudel, F. J. Framework for network survivability performance. *IEEE J. Select. Areas Commun.* **12**: 46–51, 1994.
2. Wu, T. H. *Fiber Network Service Survivability*. Artech House, Norwood, MA, 1992.
3. Alevras, D., Grötschel, M., Jonas, P., Paul, U., and Wessäly, R. Survivable mobile phone network architectures: Models and solution methods. *IEEE Commun. Mag.* 88–93, Mar. 1998.
4. Wilson, M. R. The quantitative impact of survivable network architectures on service availability. *IEEE Commun. Mag.* 122–126, May 1998.
5. Evans, J. R. Maximum flow in probabilistic graphs—The discrete case. *Networks* **6**: 161–183, 1976.
6. Lee, S. H. Reliability evaluation of a flow network. *IEEE Trans. Reliability* **29**(1): 24–26, 1980.
7. Shogan, A. W. Modular decomposition and reliability computation in stochastic transportation networks having cutnodes. *Networks* **12**: 255–275, 1982.
8. Kulkarni, V. G., and Adlakha, V. G. Maximum flow in planar networks with exponentially distributed arc capacities. *Commun. Statistics–Stochastic Models* 1(3): 263–289, 1985.
9. Rueger, W. J. Reliability analysis of networks with capacity-constraints and failures at branches and nodes. *IEEE Trans. Reliability* 35(5): 523–528, Dec. 1986.
10. Alexopoulos C., and Fishman, G. S. Characterizing stochastic flow networks using the Monte Carlo method. *Networks* **21**: 775–798, 1991.
11. Alexopoulos, C., and Fishman, G. S. Sensitivity analysis in stochastic flow networks using the Monte Carlo method. *Networks* **23**: 605–621, 1993.
12. Alexopoulos, C. A note on state-space decomposition methods for analyzing stochastic flow networks. *IEEE Trans. Reliability* **44**(2): 354–357, June 1995.
13. Ball, M. O., Hagstrom, J. N., and Provan, J. S. Threshold reliability of networks with small failure sets. *Networks* **25**: 101–115, 1995.
14. Strayer, H. J., and Colbourn, C. J. Bounding flow-performance in probabilistic weighted networks. *IEEE Trans. Reliability* **46**(1): 3–10, 1997.
15. Chan, Y., Yim, E., and Marsh, A. Exact and approximate improvement to the throughout of a stochastic network. *IEEE Trans. Reliability* **46**(4): 473–486, 1997.
16. Biggs, N. L., Lloyd, E. K., and Wilson, R. J. *Graph Theory 1736–1936*. Clarendon Press, Oxford, 1976.
17. König, D. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Math. Ann.* **77**: 453–465, 1916.
18. König, D. *Theorie der Endlichen und Unendlichen Graphen*. Chelsea, New York, 1950.
19. McDonald, J. C. Public network integrity—Avoiding a crisis in trust. *IEEE J. Select. Areas Commun.* **12**: 5–12, 1994.
20. Dunn, D. A., Grover, W. D., and MacGregor, M. H. Comparison of k-shortest paths and maximum flow routing for network facility restoration. *IEEE J. Select. Areas Commun.* **12**: 88–99, 1994.
21. Doverspike, R. D., Morgan, J. A., and Leland, W. Network design sensitivity studies for use of digital cross-connect systems in survivable network architectures. *IEEE J. Select. Areas Commun.* **12**: 69–78, 1994.
22. Egawa, Y., Kaneko, A., and Matsumoto, M. A mixed version of Menger's theorem. *Combinatorica* **11**(1): 71–74, 1991.
23. Aggarwal, C. C., and Orlin, J. B. On multi-route maximum flows in Networks, Preprint, 1996.

24. Nagamochi, H., and Ibaraki, T. Linear-time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica* 7: 583–596, 1992.
25. Kishimoto, W., and Takeuchi, M. On m-route flows in a network. *IEICE Trans.* **J-76-A**: 1185–1200, 1993. [In Japanese.]
26. Kishimoto, W. A method for obtaining the maximum multi-route flows in a network. *Networks* 27: 279–291, 1996.
27. Kishimoto, W. Reliable flows against failures in a network. In *Proceedings of the 1994 IEICE Spring Conference,* Yokohama, A-9, April 1994. [In Japanese.]
28. Kishimoto, W. Reliable flows against failures in a network. In *Proc. of the 1994 IEICE Spring Conference,* A-11, April 1994. [In Japanese.]
29. Kishimoto, W. Reliable flow with failures in a network. *IEEE Trans. Reliability* **46**(3): 308–315, 1997.
30. Kishimoto, W., and Takeuchi, M. On two-route flows in an undirected network. IEICE Technical Report, CAS90–19, DSP90–23, 1990. [In Japanese.]
31. Kishimoto, W., Takeuchi, M., and Kishi, G. Two-route flows in an undirected flow network. *IEICE Trans.* **J-75-A**(11): 1699–1717, 1992. [In Japanese.]
32. Ford, L. R., Jr., and Fulkerson, D. R. *Flows in Networks.* Princeton Univ. Press, Princeton, NJ, 1962.
33. Korte, B., Lovász, L., Prömel, H. J., and Schrijver, A. *Paths, Flows, and VLSI-Layout.* Springer-Verlag, Berlin, 1990.
34. Chen, W. K. *Theory of Nets: Flows in Networks.* Wiley, New York, 1990.
35. Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1993.
36. Ozaki, H., and Shirakawa, I. *Theory of Graphs and Networks.* Corona, Tokyo, 1973. [In Japanese.]

23
TECHNIQUES IN MEDICAL SYSTEMS INTENSIVE CARE UNITS

BERNARDINO ARCAY
CARLOS DAFONTE
JOSÉ A. TABOADA*

*Department of Information and Communications Technologies, Universidade Da Coruña, Campus de Elviña, 15071 A Coruña, Spain; *Department of Electronics and Computer Science, Universidade de Santiago de Compostela, 15782, Santiago de Compostela (A Coruña), Spain*

I. GENERAL VISION ON ICUs AND INFORMATION

In recent years, advances in medicine and clinical engineering have provided physicians with more biomedical instruments for the follow-up of a patient's physiological condition. One of the clinical areas that has benefitted most from this evolution has been those units that are charged with the follow-up and treatment of critical patients, such as intensive care units (ICUs).

The recuperation process of a patient in these units can be seen as a transition from a high-risk condition, a critical situation, or a condition of physiological instability toward a correct objective condition of stability. This transition is carried out through a series of intermediate states and under continuous supervision: the acquisition, analysis, and validation of the information, the definition of alarm levels, and finally, the realization of therapeutical interventions.

Database and Data Communication Network Systems, Vol. 3

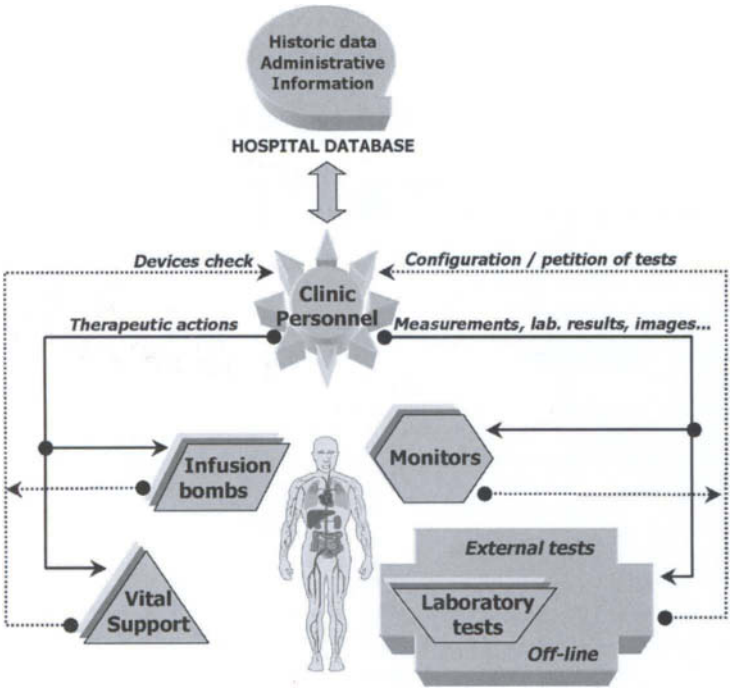


FIGURE 1 Scheme of relations in an intensive care unit.

As a result, a physician in an intensive care unit needs to handle a great amount of information, up to 30 variables for each patient, which is coming from various devices, up to 20 in number [22], that monitor the physiological variables of the patient, from the results of laboratory tests, or from parameters associated with vital support instrumentation such as mechanical ventilators or infusion pumps (Fig. 1).

In general, the forementioned tasks present a strongly dynamic and contextual character, since they largely depend on the patient's basic pathology and its evolution, the reaction to the medication, the existence of alarm states, or other medical complications. On the other hand, the planification and execution of an ICU personnel's work are very often conditioned by assistance requirements (supervision), temporal requirements (decision time), and documentation requirements (the making of reports) [16].

The design of monitoring and supervision systems in an ICU will have to take into account all these aspects. The requirements therefore are the following:

(i) Integrate the biomedical monitoring instrumentation and facilitate the acquisition of data from other sources, such as the laboratory analysis.
(ii) Store the information and facilitate its recuperation, two processes for which databases are indispensable since the design of the system largely depends on them.
(iii) Dispose of ergonomical interfaces that allow the integration and exploitation of the data, the study of historic data, and the management and configuration of the instrumentation.

(iv) Provide a support that allows the configuration and management of the alarm levels associated with the patient's instrumentation.

At the present time, information systems in ICUs tend to dispose of network integrated units that allow the centralization and storage of the information (by means of databases), the definition and implementation of standards (in acquisition, storage, or visualization), the fusion of data, the design of ergonomical interfaces, the implementation of an efficient alarm management, or the study of the tendency and evolution of a patient's physiopathological states.

Considering this network architecture, an ICU's system can be naturally integrated into a hospital information system (HIS) with all the advantages that this implies [9,46,47].

The work of the physicians as well as the infirmary staff in ICUs is considerably facilitated by the integrated management of information, which allows a clear improvement in assistance time, assistance quality, decision taking, and ulterior study and exploitation of the data.

The next step in this ICU structure is a system for intelligent monitoring [32]. Artificial intelligence techniques are very appropriate for the design of this kind of system [32], considering the characteristics of the process: complex systems without global behavior models; a strong temporal dependence; an analysis heavily determined by the physiological context and the evolution of the patient; performances often based on the experience of the physician; the need to adapt and focus the information acquisition; an important symbolic component in the information; and a strong relationship between the variables.

We can therefore extend the requirements of the monitoring system and make it as follows:

(i) Facilitate a data acquisition and classification that considers the dynamic and contextual character of the process.
(ii) Dispose of an adaptive monitoring so as to manage the computing and temporal resources according to the needs of the monitoring, but always prioritize the tasks for time.
(iii) Allow an intelligent management of the alarms.
(iv) Facilitate the definition of knowledge-based modules for interpreting the data and suggesting adequate therapies.

In order to meet the forementioned requirements, the development of intelligent monitoring systems in general and for ICUs in particular must imply the integration of various techniques. The computing structure of the system is indeed a hybrid structure that must integrate algorithmic and instrumental techniques based on knowledge, databases, etc.

II. INTELLIGENT DATA MANAGEMENT IN ICU

In the previous section we explained the requirements for the development of an intelligent ICU monitoring system. This section will now analyze a few fundamental design aspects, leaving for the next section the structure of the knowledge bases and the databases.

A. Information Acquisition. Network Structure

The integration of devices proper to an ICU has several approaches from the viewpoint of network design. On the one hand, researchers have proposed ad hoc solutions that allow the connection of a set of medical devices of one specific trademark. It is a method that leads quickly to the implementation of this philosophy and has frequently been used as a strategy for the development of high-level applications in these clinical environments. Unfortunately, the enormous variability of the connection hardware and software between various medical devices, even if they come from the same manufacturer, represents very often an important obstacle to designers who must maintain and update the applications.

A second and more flexible solution consists of incorporating the standards that come from the industries. Corporations, however, often do not consider the particularities of a ICU environment, which requires a series of requirements nonexistent in other environments. Let us mention some of these requirements, keeping in mind the large variety and amount of generated information and aiming at the smallest requirement of a user for routine tasks.

Apart from the characteristics common to any kind of data network, we must consider a structure that is flexible enough to:

- Allow the connection of medical devices that differ in trademark and version, thereby facilitating expansion and updating of the system:
- Allow a hot plug-in so as to modify the monitoring of a patient through the incorporation or elimination of devices.
- Reconfigurate automatically the network after the connection or disconnection of devices. This "plug and play" capacity implies that the system is able to relate automatically each apparatus with the bed or the patient it has been connected to, without asking any kind of special intervention from the user.
- Allow the configuration of the distinct devices in the ICU by offering a uniform interface for all of them.
- Make the local network connect with other hospital networks, thus connecting their databases, in order to facilitate the future incorporation of information coming from other services such as administrative, laboratory, or image data.
- Automatic recovery in case of failure: errors should indeed have the smallest impact possible on the final behaviour of the system. An example of how this can be achieved is the local storage of information in case of a communication disconnection.
- Indicate a deliberate disconnection in order to avoid generating alarms associated with the malfunctioning of devices or communications.

The third and final initiative for the design of centralized systems proposes the creation of a specific LAN (local area network) standard to perform in an ICU environment (Fig. 2). After two members of AMIA (Gardner and Shabot) launched the idea in 1982, a special committee of the IEEE was created in 1984 to that very purpose. The project was called IEEE P1073 Medical Information Bus (MIB) [14] and included the definition of the seven layers of the OSI (Open System Intercommunication) model of the ISO (International Standard Organization).

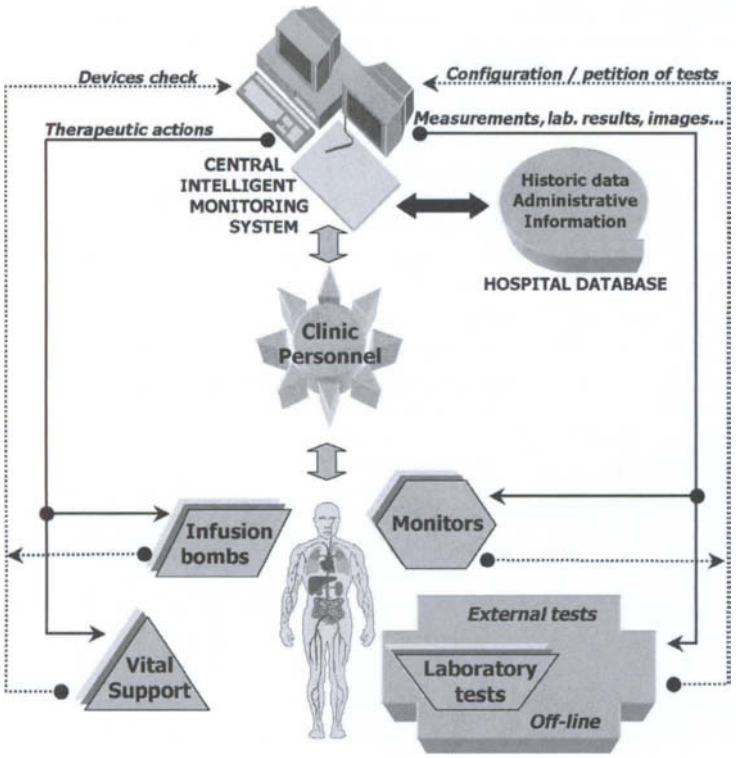


FIGURE 2 Scheme of relations in an intensive care unit using monitoring systems.

The standards proposed for the lower levels and described in IEEE documents 1073.3.1 and 1073.4.1 were approved in 1994 and entirely adopted by the ANSI (American National Standard Institute). It is these documents that are being used as a basis for the development of the European CEN TC251 standard, which other countries, such as Canada and Australia, are also about to adopt.

The higher levels are charged with the communication of the data, content, format, and syntax of the message. Since there is such a great variety and quantity of information, these levels present the largest number of problems for reaching consensus on what defines a standard. While this book is being published, researchers are currently working on a version that will be submitted to the development teams for a vote.

Although at these levels the standard proposed has its own language for communications, the MDDL (Medical Data Device Language), it also acknowledges the contributions of other organizations and teams dedicated to the development of medical standards; the final version will therefore accept the use of HL7.

Founded in 1987, the HL7 aims at the development of standards for the electronic exchange of clinical information that, apart from being clinical, is also administrative and economic. Its name refers to the fact that it describes only the higher levels of the OSI model, i.e., the application level (level 7).

TABLE I Standards for Communication between Medical Devices

Health Level-7	For exchanging data among clinical systems
X12N	For exchanging insurance, eligibility, and managed care information
DICOM	For exchanging clinical images
ASTM	For exchanging messages about clinical observations, medical logic, and electrophysiologic signals. There are other standards about healthcare identifiers, system functionality, etc.
NCPDP	For exchanging data processing standards of the pharmacy services sector of the health care industry
IEEE	For standards on medical device information and a general informatics "framework"
Clinical Context Object Workgroup	For standards coordinating multiple applications within a workstation. Applications that participate will link together on the patient, encounter, or other data objects, so that if one switches they all switch together.

Note: For more information: http://grouper.ieee.org/groups/mib/archives/related.html.

We must remember that, although the standard we describe was developed by the IEEE, the urgent need for a uniformalized communication between medical devices has pushed each country or community to develop its own standards in this field. There have also been other proposals for standards that uniformalize the format or the exchanges of all sorts of medical information. Table 1 describes a few of these standards.

B. Real-Time Information Management

The process of monitoring and continuous supervision associated with an ICU implies one very important restriction: the need to work with real-time information. It is a time aspect that affects all the levels of an information system and, consequently, all the associated processes. We must therefore see the time variable as another part of the system and integrate it into the various tasks: acquisition, network management, classification, storage in the database, exploitation, design of the inferential processes, presentation, etc.

As for the temporal problem, a few of the characteristics of these systems could be enumerated as follows [23]:

- *Nonmonotonicity.* Conclusions valid in the present may become invalid at a later moment, so we must provide mechanisms that maintain the coherence in the knowledge base and in the databases.
- *Continuous operation.* These systems must remain active over large periods of time. In order to guarantee the robustness of the system, we will have to make full use of the memory and install mechanisms of recuperation after errors.
- *Reaction capacity in the case of asynchronous events.* The system needs to be able to detain its reasoning in order to assimilate the consequences of events such as alarms, and will do this through an occasional modification of the scheduling proposed.

- *Uncertain or invalid data.* The direct interaction with the environment will inevitably provoke noises during the functioning of the instrumental sensors. These noises must be detected and interpreted accurately.
- *High performance.* The system must dispose of enough performance velocity so as to respond to the temporal restrictions imposed by the execution of the monitoring tasks.
- *Temporal reasoning.* This kind of system must be able to reason with the time component of the information, since the analyses will frequently be based on events that are previous to the actual ones or will have to be correlated temporally.
- *Focalization of attention.* There must exist mechanisms that focus the attention of the system on resolving the most urgent problems, diverting toward them the use of resourses and optimizing in this way their own use.
- *Guaranteed answering time.* The expert system must be able to guarantee a useful answer before a preset lapse of time.

The first differential problem that surges during the development of these systems is the need to represent time as another element of analysis. Taking into account the existence of time will have enormous consequences for both the representation of the knowledge and the way we reason with this knowledge. It is not sufficient to mention that a patient shows a certain symptom; we must also indicate when the symptom first appeared, when it dissapeared, and whether it persists. This way we make it possible to compare the information within its temporal context.

In order to determine a temporal content with which we can reason, we must assign to each fact one or various temporal parameters, for instance, the initial date and its duration. Mac Dermott [26] and Allen [2] both proposed temporal representations as respectively instants and segments. Tsang [48] has proposed that the two representations are actually equivalent: this would naturally mean that any type of temporal relationship can be expressed in function of the other, and vice versa.

The temporal context of medical information influences the result of the reasoning, since the same symptoms can lead to the conclusion of different pathologies, depending on whether these symptoms present themselves simultaneously or in some other time sequence. During a process of continuous monitoring, it also allows us to discard information that, due to its limited validity (latency) period, has lost its meaning. Finally, we can also arrange the measurements in time and in this way analyze the patient's evolution.

The reasoning a system uses to reach a conclusion can be seen as the evolution between a series of partial knowledge states. The purpose of any type of expert system is to find a way to relate the initial state with a final state or conclusion. Except for the cases of simple problems, the condition diagram can rapidly extend itself in what is called a combinational explosion [43], which makes it impossible to go through the whole diagram in search of the optimal solution. As a result, we must apply certain cutting criteria in the tree of reasoning so as to eliminate those branches or segments that turn out to be less promising. Such algorithms must guarantee the existence of at least one solution within the selected subtree [7,33,37,52].

This introduction of a time variable in the reasoning of the system complicates not only the size of the tree, but also the validity of the conclusions. Therefore, in order to treat time as one more variable of the problem, specific reasoning strategies were developed: the temporal reasoning in critical time, the interruptible reasoning, and most of all strategies to maintain the truth associated with the nonmonotonicity of the reasoning, a primordial factor in those systems that consider the time element.

Time is not the only parameter that influences the reasoning; the fact that we work in a medical environment with imprecise information makes us treat this reasoning matter with great uncertainty [50]. Finally, the possibility of working with partial models or with heuristic decision rules puts before us the choice of either the use of reasoning based on "deep" knowledge or that based on "shallow" knowledge [32].

1. Temporal Reasoning

Temporal reasoning [27] is the capacity of the system to establish temporal relationships between facts. The apparent simplicity of this problem, however, should not mislead us: implementing this kind of reasoning usually provokes considerable problems of computational complexity.

Real-time reasoning systems often use very simple models to resolve temporal problems that are only valid in the context that defines them, a method which leads to applications that are very efficient but rarely extendible to other domains. In fields such as medicine, where the temporal expressiveness is absolutely basic for numerous facets of medical activity (diagnosis, monitoring, follow-up, prognosis, and therapy selection), researchers have tried over the past few years to develop theoretical general models that are independent parts of the system and defined as temporal reasoners [35].

Temporal reasoners do not reason about the temporal component of the domain knowledge, but about time itself: their purpose is to understand temporal expressions such as before, during, and after, which refer to temporal labels and can appear as absolute dates (11/3/67), relative dates ("in the third week of the treatment or during the second month of pregnancy"), or pseudo-dates ("during the first consultation").

The general time reasoner must be able to interpret all these circumstances in their temporal context so as to arrange temporally the various events, detect temporal inconsistencies between them, and later on, be able to answer questions, such as "did event X take place before, after, or during event Y?", which is the kind of question asked by the user as well as the reasoner of the domain.

The last factor can complicate the problem of temporal reasoning even more: it consists in allowing approximative time expressions like "last month" or "last week". Approximative reasoning deals with these kind of factors by applying special tools that avoid the propagation of unprecise data that are impossible to relate [27].

On the other hand, we also need another kind of temporal reasoning: a type that can be called temporal reasoning in the domain and does not determine relative temporal relationships between events but interprets domain information in its temporal context. As an example, two heart rate data of 60 and 100

pulsations can be perfectly valid if they have been taken with a time interval of 15 min, but they must be considered incorrect if the measurements were made with an interval of 5 s.

Approaching this problem in such a generic way as described above would consume important computational resources that cannot be disposable in real-time execution systems. Besides, these systems often contain necessary mechanisms for the automatic incorporation of information and in this way simplify the problem and allow a less rigurous focus on the temporal problem:

(1) Date stamps are numeric values in the shape of segments or instants. This simplifies enormously the analysis of the precedency, since it will be reduced to an algebraic analysis.

(2) Coherence of the temporal information is guaranteed, since the facts or data are automatically introduced by the domain reasoner and hence do not need to be verified.

It is for this reason that the majority of these systems turn toward ad hoc solutions for solving the temporal problem, mixing deliberately the time reasoning and the reasoning in the domain time.

2. Reasoning in Critical Time

All real-time reasoning systems face a certain limit time, a moment from which their solutions start to lose their value. This limit can be either hard or soft [44]: hard in cases like the triggering of an alarm after the damage that it was supposed to avoid has been done; soft when a conclusion about, for instance, the general condition of a patient gradually loses its initial efficiency even though it has been valid for a certain time. For those real-time systems that have hard limits it is fundamental to obtain a solution before reaching the limit time. They can do this by triggering, for instance, the alarm while showing the motive, even though it is impossible to specify why the alarm went off or how it could be corrected.

The largest problem with expert systems is that, due to the complexity of the space of states with which they work, they do not allow us to calculate how long it will take them to find an answer. There exist, however, several solutions for guaranteeing that this answer is found within the disposable time [15].

One possibility is progressive reasoning, which implants the knowledge in levels: the first levels are focused on finding a fast solution, the subsequent levels on refining it. In this way, the system starts with a coarse solution, refines it through the execution of successive levels, and stops the process when it reaches its limit time.

Another solution presupposes that we know exactly how much the system disposes to come to a conclusion (the sampling time of the information, for instance), which means that we are able to lead the reasoning in various directions to reach more or less elaborated solutions. This method is equivalent to realizing more or less drastic cutting mechanisms of the states tree [34]: if the disposable time is larger, the cutting process will be smoother and the states tree can be examined more deeply.

3. Interruptible Reasoning

Another option to be taken into account during real-time reasoning is the possibility of detaining a reasoning line for a cause that is external to the reasoning itself. There are two kinds of interruptible reasoning [15].

The cancelable reasoning abandons the reasoning in course because it has lost its meaning, for instance, because the values of the initial facts have been altered. If this occurs, the easiest solution would be to restart the reasoning with new information, but due to temporal restrictions we may be obliged to try and take advantage of the work that has already been done. In that case it is possible to propagate the alterations through the reasoning graph, or else, if the initial conditions can be considered equal, we may assume that the obtained answers are qualitatively correct.

The suspendible reasoning allows us to detain the actual reasoning line when more urgent tasks appear, and lets us continue that line from the point where we left it once those tasks have been taken care of. This mechanism naturally obliges us to maintain the environment of the interrupted reasoning line and may therefore be expensive in terms of time and memory.

On the one hand, the cancelable reasoning is most indicated for applications in intelligent control systems, in which a conclusion based on expired data can lead to incorrect control actions. The suspendible reasoning, on the other hand, is most useful in all those processes that execute tasks with different priorities, processes in which the high-priority tasks normally associated with alarms will detain or even impede the execution of lower-priority tasks.

4. Nonmonotonous Reasoning. Maintaining of the Truth

An expert system based on facts invariable with time can use the conclusions it has obtained in a given moment as part of the database for a subsequent reasoning cycle. In that case we call it a monotonous reasoning, and it is identical to those reasonings whose conclusions stay valid over time.

Quite the opposite is a system whose basis varies in time and which must therefore continuously revise its conclusions to decide whether they are still correct. For this reason these systems incorporate in their facts base mechanisms for maintaining the truth [12], which serves to eliminate those data and conclusions that have stopped being valid. These mechanisms can go through the states tree, modifying the elements that depend on each new information, or they can change their values to "unknown" so that they must be recalculated whenever they are needed.

5. Reflexive Reasoning

A computational system is called reflexive when the system forms part of its own domain [42]; a reflexive system is therefore a system that has a certain representation of itself. A set of structures that refers to its own system is called "own-representation" of the system.

The reflexivity of a system does not demand that system to be based on knowledge; it rather uses as a start the information that refers to itself (reflexive architecture). Introducing the representation of the system as a part of its knowledge base allows us to sophisticate its control, since now decisions can

be taken about what tasks to realize according to the resources of the system (CPU occupation, disposable memory, etc.). In this way the pruning algorithm, selected during the reasoning in critical time, would have to consider not only the available time but also the occupation level of the system.

A particular case of reflexive systems are the introspective systems. In these systems, the motive for reflection is to determine "what the system knows about what it knows." An example of this behavior is the default reasoning, in which part of the information being analyzed corresponds to generic values or parameters that are only substituted by real data when these are available. In order to realize such a replacement, the system obviously needs to be conscious of the existence of such data.

6. Reasoning with Uncertainty

Rather than resolve the problem of working in real-time, the reasoning with uncertainty allows us to work with complex problems, or with incomplete or imprecise information. The MYCIN, for example, one of the first expert systems in medicine, made it possible to diagnose bacterial diseases while simultaneously taking into account its own impreciseness when coming to conclusions.

There are various methods for working with uncertainty. As we have seen in the case of time-related systems, these methods imply the introduction of the uncertainty's representation in the knowledge base and the capacity of the system to propagate the uncertainty and reason with it:

- *Classical probabilistic logic.* Basically, it supposes the application of the Bayesian probability that generally leads to complex calculations in the propagation processes.
- *Certainty factors.* This method was used in the MYCIN and implies that each fact and rule of the knowledge base has a determined reliability, although only those rules that possess a certain degree of certainty in their conditional part are used in the analysis.
- *Groups and fuzzy logic.* This is an intent to formalize the reasoning with uncertainty in which, to a certain degree, each element belongs to each possible classification set.

7. Superficial Reasoning and Profound Reasoning

Even though we might speak in terms of superficial and profound reasoning, it would be more exact to say that a system reasons on the basis of superficial or profound knowledge. We will therefore use the term "knowledge" instead of "reasoning."

In the case of superficial knowledge, we find descriptions of prototypical situations that are resolved thanks to the experience of an expert without ever mentioning the causal relationship between the initial situation and the choice of a determined solution. We can therefore say that the decision of the system is not necessarily correct, since it is based upon the assumption that the situation described by the variables corresponds with a situation described by the expert—which is not always true. If an error occurs, all we can do is introduce new rules that distinguish between the two possibilities by defining more carefully the variables that characterize each situation. In this way, and after

various cycles of testing and modifying, these kind of systems can reach a high level of competence.

Nevertheless, the negative side is that those possibilities that have not been considered will irrevocably lead all the systems to incorrect decisions that must be corrected. As a result, the behavior of the systems designed accordingly does not slowly degrade itself as the situations move beyond their knowledge domain; quite on the contrary, each possibility not foreseen in the design of the system provokes the appearance of incorrect answers. Another drawback of these kinds of systems is their explanation capacity, since it only takes into account a trace of the executed rules. This trace merely describes the evolution of the system through the states tree and does not necessarily coincide with a rational argumentation.

This type of knowledge description has been very frequent in so-called first generation expert systems, because they make it possible to generate rapidly efficient prototypes. Now researchers are working on what they call second generation expert systems, which are systems based on profound or causal knowledge [8] and provided with information about the interaction between the elements that compose the domain, for instance, the causal links between pathologies and their symptoms [38]. Ideally, the profound knowledge presupposes a theory capable of explaining everything about a concrete domain; this explanation then serves as the basis for all actions.

The main qualities of a system with profound knowledge are the following:

- *More robustness.* As the rules of these kinds of systems are based on a causal model, we will need to purify the model whenever it is unable to explain a particular case. This correction will not only concern the error that originated the change, it may also cover other errors that initially had not been considered.
- *Easier maintaining of the knowledge base.* The methodical representation of the knowledge in this method implies a clearer representation and as a result facilitates the comprehension and alteration of the knowledge base.
- *Reusability possibility.* The same qualities that facilitate the update allow knowledge transfer between systems that work in the same domain.
- *Improvement of the explanatory possibilities.* The availability of profound knowledge allows us to generate causal explanations of the behavior of the system and its actions.

This model is obviously not without inconveniences. The main drawback is that in many cases, and especially in the field of medicine, we do not know of any model that can be implemented. This situation has justified the use of heuristic expert systems, i.e., based on superficial knowledge, for solving problems of implementation. Apart from that, applications with this kind of knowledge are much harder to develop and need more time and efforts.

C. Integration of the Intelligence

Up to this point we have treated the problem of centralized information acquisition in intensive care environments. We have discussed ways of treating the complex problem of introducing time as a reasoning element and the capacity of knowledge-based systems to treat complex problems using incomplete or

unprecise information. In this section we shall focus on the tasks associated with the management of a patient in an intensive care unit and on the systems that simplify those tasks, revising the temporal requirements and some solutions proposed by various authors.

Despite the advantages of starting from centralized information systems, many knowledge-based systems developed in parallel to centralization strategies propose ad hoc solutions. For instance, one of the first expert systems based on the automatic acquisition of information from a laboratory of pulmonary functions was PUFF [1], developed in 1979 at Stanford University and therefore prior to the 1982 proposal for the creation of a Medical Information Bus (MIB). Although this system was not conceived for monitoring, other systems such as SIMON [11] or Guardian [17,24], which are considerably inspired by this development, add other capacities to the acquisition module. One of these capacities is the incorporation of a strategy that allows the module to acquire the correct information volume so as to avoid unnecessary information loops in the high-level modules. SIMON further adds to this module the capacity of data abstraction and the possibility of comparing acquired information with that foreseen by existing models at superior levels. These comparisons then help detect artifacts and alarms or avoid unnecessary information fluxes between the system levels.

As we have seen in the SIMON project, these systems facilitate the introduction of knowledge at the stages of error detection and alarm triggering. The alarms especially are a basic aspect in monitoring systems, since their correct management allows a fast team performance in crisis situations and thus determines in many aspects the quality of the assistance.

Knowledge-based systems avoid the typical limitations of classical treshold alarms such as:

(1) Excessive time consumption due to extensive treshold configurations,
(2) A lack of dynamic contextualization of the triggering criteria, and
(3) The repeated triggering of alarms caused by noises, artifacts, or small oscillations around the treshold, a phenomenon that undermines the confidence of clinical personnel on monitoring teams.

The introduction of the AI makes it possible to predict a patient's evolution through the use of partial models or approximate reasonings. These predictions then allow us to generate prealarm states and give us more time to avoid crisis situations or permanent damage to the patient. They also offer the possibility of prioritizing alarms [5], highlighting those that are the cause of a risk situation rather than those resulting from it; finally, they provide explanatory capacity and hereby simplify the interpretation tasks for the physician.

Another task of the systems improving the decision-making is planification. Often, the therapies provided to these patients are rather complex. However, systems like ONCOCIN [41] or ATTENDING [28] have proved their capacity to schedule therapies and associated tests. During this task the temporal requirements can be relatively soft, which is why it has been possible to develop these systems far beyond the hard requirements that determine a real-time monitoring. When we consider the real-time monitoring processes linked to the tasks that help the decision-making, we must take into account those resources that

are disposable in the system at that moment (network congestion, states of the acquisition devices, capacity of the processors, etc.), analyze them contextually, and consequently define the planning of the tasks that must be executed, always giving priority to some and slowing down or eliminating others (Guardian).

Here, as in any information system, the generation and presentation of reports is a fundamental aspect. Using the technique of the expert systems is not necessary, but it allows us to present the information in an interpreted way, generating documents that can be assimilated faster by the clinicians and facilitating the development of systems that actively help the physician to make decisions. A pioneer in both tasks has been the HELP [36] system.

Once we dispose of the acquired information, it is possible to analyze it with expert systems specifically developed for the pathology or pathologies that must be treated. Most systems have been developed at this stage, many times ignoring the temporal problem and focusing on the knowledge engineering tasks (MYCIN [40], DXplain [6], Internist-1 [29,30], etc.).

Since the different knowledge modules that we have been analyzing do not constitute isolated entities, but all work toward the same purpose and on the basis of the same data, there must be communication between them. To this effect researchers have developed strategies like the "blackboard" used in Guardian or distibuted architectures like that proposed in SIMON. In any case it is important to mention that one and the same task may present both soft and hard temporal requirements according to its particular purpose. This occurs, for instance, with the selection of a therapy, which will be a slow process if it concerns the recuperation of a pathology, but a very fast process if we pretend to correct an alarm. The SEDUCI system [46,47] deals with these criteria by proposing a development of distributed expert systems. This means that it incorporates intelligent mechanisms in the hardware elements close to the patient (reasonings that could be called "reflexive," with less answering time) as well as in the central system, conceived for diagnosis and selection of a therapy.

III. KNOWLEDGE BASE AND DATABASE INTEGRATION

In the field of intelligent monitoring, one of the major current areas of interest is the incorporation of databases as an integrated part of the overall monitoring system. For systems such as our rule-based expert system for intelligent monitoring of patients in ICUs [4,11], the advantages of adding a database to the expert system [3] include:

1. Facilitation of the use of filed data by the rules of the expert system (filed data can be retrieved by means of highly efficient database (DB) mechanisms rather than by much slower, more cumbersome, and more restrictive methods available in expert system programming languages).
2. The possibility of having a larger working memory for the expert system. In our system this is very important, since we must work with a large number of physiological parameters (from 20 to 30), as well as the resultant data from digital signal analysis (ECG, EEG, Capnography, etc.), for each patient at any one moment.

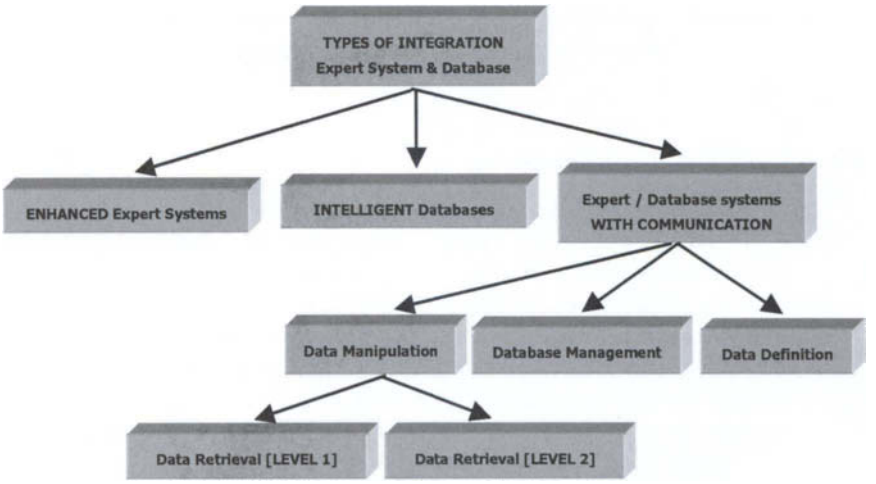


FIGURE 3 Integration types of databases and expert systems.

3. Facilitation of permanent filing of case data. Using this information it is possible to analyze, a posteriori, the patient's progress by studying the cause of alarms, instabilities, knowledge base debugging, etc.

4. The possibility of using expert system rules to implement automatic intelligent screening of data prior to permanent filing so as to ensure the consistency of the filed data. Incidentally, this fourth objective is also of interest from the viewpoint of DB theory.

There are different ways to integrate both systems (Fig. 3). They can be classified into three categories [3] according to their degrees of coupling and the allocation of control:

1. *Enhanced expert system.* Extended data management facilities are incorporated into the expert systems.

2. *Intelligent/deductive databases.* In this type of system, deductive components are embedded into database management.

3. *Expert/database systems with communication.* Database management and expert systems are independent with some form of communication between them. Also, the communication can be classified into three classes, data definition, database maintenance and administration, and data manipulation.

It may be logical to believe that the implementation of a specific database for a concrete working environment of the expert system (first type of integration) is the best solution. In our working environment, however, we are dealing with an information system that is already implanted in the hospital (HIS) and that will incorporate a relational database manager with a large quantity of historical data and specifically developed applications. Therefore, the most practical and realistic solution for integration in this kind of system and the use of information from disposable historic data is communication with this already existing manager. We also must keep in mind the development level of the commercial database managers, which have reached a flexibility, robustness,

security, and velocity that could only be improved by investing heavily in design and implementation and are therefore out of reach of most expert system developers.

A. Intelligent/Deductive Databases

In this type of system, the deductive components have been incorporated into the database manager system. These deductive components can improve the functionality and efficiency of the database by allowing it to maintain, among others, restrictions of integrity and optimization of queries. Besides, during the incorporation of deductive components into the system, the knowledge rules that interact with the stored data can be implemented in the same environment without using an external tool: this allows typical database applications (data entry, report generation, etc) and the intelligent processing to realize tasks together.

We can define a database as a triple [31] composed of the following elements:

(1) *A theory.* It consists of a finite set of axioms and meta-rules that describe how one operates with the particular theory.
(2) *Integrity constraints.* These must be satisfied by a database that must be consistent with the theory.
(3) *A set of constants.* A finite set over which the theory is defined.

The way in which this definition tends to be implemented allows us to maintain it using a syntax similar to that of natural language, which means that the integrity restrictions of the database can be expressed with knowledge rules. These database systems also seek to improve the mechanisms for manipulating the information they contain. Instead of using a very strict language to consult, modify, or eliminate data, they pretend to facilitate an interaction in near natural language by means of integrating expert systems. This kind of interaction would avoid the use of complex semantic constructions that now hinder the control of these data managers. Some of the capacities provide the expert systems with the following:

1. They make it possible to design front-ends, even in natural language, to interact with the database in a less complex manner.
2. They trigger the periodical or event-driven maintenance tasks by means of a simple syntax, close to that of natural language.
3. They facilitate the expression of complicated queries, like transitive and recursive ones.

Most of the relational managers that are being sold at the moment incorporate more or less sophisticated rule systems charged with triggering tasks (triggers).

B. Expert/Database Systems with Communication

Up to this moment this has been the most frequently used type of integration in the expert systems domain. In this case the expert systems and the database

management systems are independent elements with a certain communication mechanism between them. Since they have been developed without an ex profeso structure that allows them to communicate with each other, the complexity of the interaction between them will normally be larger than in the previous cases. There is, however, one advantage: the communication interface is precisely the only aspect that we will have to develop.

This communication can be classified in three groups [25]:

1. *Data definition communication.* In this type of implementation the expert system can determine the structure of the database or modify it at any given moment if necessary.
2. *Database management communication.* In this case the expert system can execute typical database management tasks such as the realization of backups and recovery, or storage analysis.
3. *Data manipulation communication.* The expert system sends data manipulation commands to database management systems to update, insert, delete, and retrieve data. The expert system can realize analyses on the basis of information it recuperated from the database. Using only this type of communication, and with an appropriated implementation, we will have a system equivalent to an intelligent/deductive database.

A knowledge rule can be divided in two parts: an **If-Part** (condition) and a **Then-Part** (action). Based on that information, we can distinguish two types of retrieval within this third communication group:

- *Simple data retrieval.* The facts are recuperated directly from the database. This retrieval requires an information charge operation before the operation of the expert system, which means that direct recuperation of data from the database through the If-Part of the rules is impossible.
- *Full data retrieval.* In this case information from the database is recuperated through the If-Part of the rules using a queries language supported by the management system of the database. This means that it is not necessary to charge the information stored in the database before the operation of the expert system.

It is also possible to use this integration method between an expert system and an intelligent/deductive database so as to obtain two communicating systems whose "intelligence" will not directly relate to each other, due to the absence of a common database or a common inferences motor. Both systems will be dedicated to two totally different tasks without affecting the global functioning of the system.

If we intent to extend the knowledge base of the expert system to a database, one of our primordial purposes is to establish the most natural and transparent communication possible between the two systems. Ideally, the implanted communication mechanism would allow us to have access to any of the two storages in a way that is transparent even to the knowledge engineer. Our first task, therefore, is to structure the knowledge in a similar way in both the knowledge base of the expert system and the database. In case we use a relational database, the most common one in real environments, we face a serious problem: most of the development languages and tools of expert systems (OPS/83/R2, Nexpert

Object, G2, etc.) use a scheme of objects or pseudo-objects for information storage in the database. This means that we will have two systems (database and factsbase) that use a different storage scheme. One of the most interesting ways to deal with this problem was developed by Gio Wiederhold in the PENGUIN project [51]. He presented a study of the management of objects through a storage structure in a relational database, outlining the following advantages in each of these information organization systems:

1. Relational technology provides a basis for reliable sharing of large bodies of information.
2. Object approaches provide conceptual locality of information to focus users' attention.

Decision–support systems (as in the case of ICUs) require the combined use of database and intelligent application. In that work he presents a schedule that allows the object-oriented data structures to be derived from a shared relational database. It is based on the similarities between the concepts of object and view (virtual chart generated on the basis of a consult) and states that both are intented to provide a better level of abstraction to the user. The objects are tailored to adapt themselves to the needs of workstation applications, while maintaining strict consistency with the underlying shared database model (relational model).

An important factor to be studied in a similar statement is the effect that the use of virtual charts (views) has on the speed of accesses to the database where "consults on consults" are concerned. This will largely depend on the managements of these views by the database manager.

C. Temporal Databases

A DBMS stores information in a perfectly defined format. In the case of a relational database this format consists of relational tables. Each table contains tuples that are composed of attributes. In the case of an object-oriented DBMS the information on entities is stored by means of objects. The difference between data that are stored in a temporal database and those stored in a nontemporal database is that the temporal information attached to the data expresses when a datum was stored and what its validity is. Conventional databases consider the data stored in it to be valid at time instant now; they do not keep track of past or future database states. By attaching a time period to the data, it becomes possible to store different database states.

The first step to obtain a temporal database is to timestamp the data, which allows us to distinguish between different states in the database. A very often applied approximation is to timestamp entities with time periods. In the context of relational databases, which is of interest to us, tuples are timestamped. In order to decide which period we store in these timestamps, we must consider two different notions of time that are relevant for temporal databases [45]:

- *Validity time.* This timestamp indicates the period during which a fact is true in the real world.
- *Transaction time.* The period during which a fact is stored in the database.

Identifier	Name	Reason	Enter	Exit
23.332.222	Anton Serrapio	Craneoencephalic Traumatism	7/1/1998	8/2/1998
34.333.212	Merita Tuiriz	Dilated cardionmiophaty	1/5/1999	1/10/1999
23.332.222	Anton Serrapio	Post-surgery	1/19/1999	void
34.865.345X	Valeria Bruins	Valvular surgery	1/18/1999	1/19/1999

FIGURE 4 Example of a temporal database table.

These time periods do not have to be the same for a single fact. In the database we have, for instance, information on patients that were hospitalized the year before the actual one. The valid time of these facts is somewhere between January 1 and December 31; the transaction time, however, starts when we insert the facts into the database.

Suppose we want to store information on the patients of an intensive care unit. The database table shown in Fig. 4 stores the history of patients with respect to the real world. The attributes Enter and Exit actually represent a time interval closed at its lower bound and open at its upper bound. The patient Anton Serrapio, for instance, was treated in the hospital from 7/1/1998 until 8/2/1998 for a Craneoencephalic Traumatism and later on he entered again the ICU on 1/19/1999; the "void" value in the exit column indicates that this patient is still being treated and that the data are actual.

A temporal DBMS should support:

- A temporal data definition language,
- A temporal data manipulation language,
- A temporal query language, and
- Temporal constraints.

A possibility in the case of relational databases is the SQL3 specification. This change proposal specifies the addition of tables with valid-time support into SQL/temporal, and explains how to use these facilities to migrate smoothly from a conventional relational system to a temporal system. The proposal describes the language additions necessary to add valid-time support to SQL3. It formally defines the semantics of the query language by providing a denotational semantics mapping to well-defined algebraic expressions.

This kind of database has multiple applications, among which is medical information management systems. Nevertheless, commercial database management systems such as Oracle, Informix, etc., are, at this moment, nontemporal DBMS since they do not support the management of temporal data. These DBMS are usually the ones we find in hospitals and ICUs, which means that we must use a type *date* supplied in a nontemporal DBMS and build temporal support into applications (or implement an abstract data type for time); the management of timestamps must take place inside the monitoring applications and expert systems involved, using techniques similar to those described in the previous section.

IV. A REAL IMPLEMENTATION

In this chapter we present an intelligent monitoring system for ICUs (supported by the XUGA10502B97 Research Project), focusing especially on the aspects related to the integration of knowledge bases and databases. This multiplatform distributed monitoring system allows for an indefinite number of expert systems for medical reasoning to be run at the user's request in a workstation, to which the bedside PCs are connected via the Ethernet and TCP/IP. These expert systems perform high-level analyses designed to help the clinician decide how to manage the patient and what monitoring facilities should be provided. Although timing considerations are not totally foreign to these expert systems, time is not as critical as in the monitoring; the real-time requirements are "soft" [44], whereas low-level monitoring data must be processed and passed on while still fresh. The central workstation also performs signal analysis tasks referred to it by overloaded bedside PCs, and manages multitasking communication with an Informix database set up in another PC. All data acquired by the bedside PCs are stored in this database, which can be accessed by all currently active expert systems. Data are in principle acquired by the bedside PCs largely via a bedside MIB [14], however, the bedside PCs can also acquire data through other channels if necessary.

These experts systems are written in OPS/83 [13], which has gained wide acceptance for real-time monitoring on account of its fast rule evaluation and its flexibility as regards to the design of inference strategies. OPS/83 allows READ instructions on the right-hand (executive) side of rules, but not on the left-hand side, and in the absence of any circumventing mechanism it is therefore impossible to use rules with conditions referring to data that must be read from an external source. At first sight, it might seem that one way of getting around this obstacle would be to provide rules that, on the basis of the current state of working memory, predict whether external data will be needed and, if they will be, load them; however, prediction of the course of the inference process is not always possible. A more down-to-earth approach is to precede each rule for which external data are needed with an auxiliary rule whose right-hand side loads the data. This latter option is feasible, but hardly elegant; what is really needed, with a view to both flexibility and efficiency, is a mechanism allowing external data to be used on the left-hand sides of rules in much the same way as internal data are used.

Here, we describe our solution to the problem. Specifically, we have written a set of C routines implementing functions that can be used on either the left- or right-hand side of OPS/83 rules to communicate with the relational database (RDB) management system Informix [19,20], which was chosen on account of its portability, capacity, and innovative features. The new functions (a) allow data stored externally in an Informix database to be requested via dynamic SQL queries and used directly on the left-hand sides of OPS/83 rules; and b) allow data acquired or generated by the expert system to be stored directly in the external database. In addition, we have provided a set of rules that screen data for reliability before filing, thus protecting the consistency of the data stored in the database.

A. Description of the Monitoring System

Figure 5 shows the general scheme of the system. At the lowest level, communication between the monitoring instruments and a bedside PC takes place largely via an MIB. This standard for medical data acquisition specifies that each monitoring device send its data to the bedside communications controller (BCC, the bedside PCs of our system). Furthermore, our system also allows direct communication between the bedside PC and the monitoring device via a conventional data acquisition system (DAS) (MIB does not have analog capacity) and can also be used to obtain information if the MIB suffers some kind of failure.

At the center of our system is the multitasking workstation. Communication between the workstation, the bedside PCs, and the PC housing the Informix database is established via the Ethernet and TCP/IP. The workstation acts as a remote server, supplying the bedside PC with extra signal analysis and data processing capacity, as a user interface and controller sending instructions to the bedside PC, and as the location of the high-level data analysis processes performed by the medical reasoning systems. It is envisaged that in an intensive care unit the bedside PC would essentially present the patient's current state and emit warnings to aid immediate attention, whereas the workstation would be located in a control room attached to the ward to assist medical staff to analyze the patient's case.

The monitoring expert system based in the bedside PCs has been written in OPS/83 and C. The conflict resolution strategy employed by the monitoring system (i.e., the way it decides which rule to follow when the conditional parts of more than one rule are satisfied) is an adaptation of the means–ends analysis

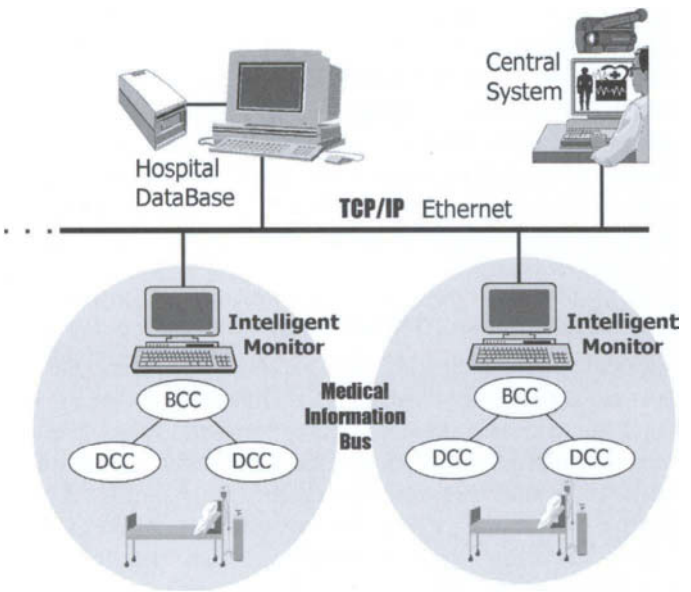


FIGURE 5 Monitoring system scheme.

(MEA) approach that OPS/83 inherited from its predecessors; the MEA strategy drives the identification and satisfaction of a succession of intermediate goals on the path to satisfying a global objective [13,18]. The inference engine of our monitoring expert exhibits three main modifications of this strategy. Firstly, because of the very nature of monitoring, in normal circumstances the whole system runs in a loop rather than terminating upon satisfying a global objective. Secondly, for many rules the element defining the logical path of reasoning is the element representing current real time, which is updated once per second from the inference engine itself immediately before redetermining the set of rule instantiations (an instantiation is a rule together with a particular set of existing memory elements satisfying its conditions); in this way, rules mediating the acquisition of data via the MIB, the DAS, or the Ethernet at specified sampling times will be executed immediately the time condition is satisfied as long as the rest of their conditions are also satisfied. Thirdly, the inference engine includes a memory analysis procedure which temporarily suspends MEA-type reasoning when memory threatens to become saturated; when this happens, all the rules whose conditions are currently satisfied are executed so that information elements made obsolete or superfluous by their execution can be destroyed, and this element destruction is implemented in the inference engine itself rather than by rules.

Communication between the bedside PCs, as BCCs, and the DCCs of the monitoring devices is governed by a protocol using MDDL, the Medical Device Data Language, defined for this purpose in IEEE standard P1073.

The workstation runs under UNIX, which allows simultaneous multitasking operation of the central user interface, the medical reasoning systems, analysis tasks referred to by the monitoring experts, communication with the Informix database, and a central control module managing the data exchange between various tasks.

The selection of an Informix database running under DOS is made because it is a system with a reasonable cost and it will allow us to do a simple migration to a UNIX platform; it can also function as an autonomous system.

Communication between the expert systems and the Informix database is managed on a client–server basis by a module organizing data requests by priority before passing them in the form of SQL queries to the database; problems associated with the use of external data requests on the left-hand sides of OPS/83 rules have been circumvented by the use of C routines to pass the SQL queries.

1. Data. Its Acquisition and Time-Dependent Aspects

Our system distinguishes between *medical* variables, such as heart rate or intracranial pressure, and *measured* variables, which are the measurements made by specific instruments; at any given time, the value assigned to a given medical variable (a semiquantitative label) will be obtained from the most reliable of the measured variable values to which it corresponds. Measured variable values are stored in memory elements of type Mdata. These elements are liable to be destroyed not until they have overrun their validity period; however, their removal will only actually be effected if the system is pressed for memory.

When ample memory is available the normal garbage collection routine allows retention of at least one already-processed Mdata element for each measured variable.

From that point of view, a fundamental aspect is the scheduling of these tasks: to situate in time the acquisition processes, i.e., determine their frequency, while keeping in mind the latency of each variable, its tendency, the process state, and the possible existence of artifacts.

These acquisition and classification rules treat the time variable as one of their LHS elements. Time is represented in segments [2], and the granularity is measured in seconds.

In order to establish the taking of actions in a time context, we compare the system's "real_time" variable with the "own_time" variable of the variable or process under analysis [46,47]. As a result, we obtain the following elements in the working memory of the rules system: Treal, the actual time stamp; TmVi, the measurement time of the Vi variable (and consequently of all the actions related to Vi); T_period, the variable's latency; and TpVi, the associated process time.

The measurement of a variable takes place in a time lapse that starts when TmVi>Treal. Furthermore, we are certain not to lose the execution of the process [46,47], since Treal is updated in the rule election mechanism before triggering each rule. The times associated with a variable or process are dynamic and contextual, which means that TmVi increases if the TpVi of a variable is superior to the available segment and the measurement task does not have priority. In this way we adapt tasks according to their latencies and priorities. A problem that may arise is the temporal overlap of two tasks: the execution of a certain task may cause another task to remain outside of its work interval and as a result causality is lost. This kind of problem depends mainly on the temporal requirements of the system [23]. It is solved with the help of the structure of the rules and the design of the inferences motor, by establishing priorities in the execution of rules via the incorporation of the rules' own weight factors. On the other hand, we must remember that in this kind of process the latencies are considerable. To minimize hard time problems, our inference engine of this module has been written in such a way as to (a) give data acquisition priority over any other task when there is enough memory to store the data, and (b) give priority to data reduction (processing) when RAM threatens to run short.

In this form, Mdata and equivalent information elements are liable to be removed from memory as soon as the time specified in their T_destroy field has passed. Furthermore, the high-level general data acquisition module can decide, on the basis of the relative urgencies of different data needs, to overrun by up to 5% the normal data acquisition interval for any given medical variable (the time specified in the T_period field of its descriptor element). This strategy reduces the information transmission in the bedside subnet.

Signals are acquired (Fig. 6) and analyzed by the DAS only if the system is relatively unoccupied. Since the time taken for signal acquisition is known, this task is only initiated if it is completed before the presumed start time of any other task. However, even if it is found to be compatible with scheduled data acquisition from other sources, the initiation of signal acquisition is subject to

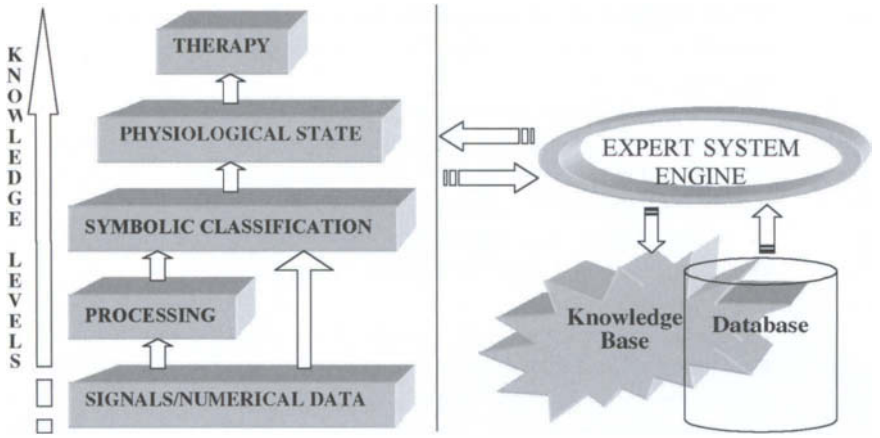


FIGURE 6 Data adquisition in SEDUCI.

other conditions designed to reduce to a minimum the possibility that it might block the reception of alarm signals; i.e., the values and trends of the patient's variables must all be normal.

Once a signal has been acquired, the DAS will send the acquired signal to the workstation for analysis if it decides it has no time to analyze it itself in the bedside PC. Otherwise, the calculation of derived variables takes up little computer time, and they are recalculated whenever any of the variables on which they depend have a new value.

After they have been acquired by the PC, numerical data are analyzed to detect artifacts and trends, and the values of the variables are then translated into semiquantitative labels in accordance with tabulated range boundaries (Fig. 7) subject to dynamic redefinition as a function of the patient's state [39]. These semiquantitative labels are divided into two classes: one set (the "H–N–L classifiers") simply notes whether the value of the variable is high, normal, or low, while the second set (the "severity classifiers") indicates the degree to which nonnormal values deviate from normality (slightly, moderately, severely, or extremely). For each medical variable, these assignments are performed using a table of thresholds specifying the numerical ranges corresponding to the various combinations of H–N–L and severity. For each patient, the Informix database stores information on all changes in thresholds, the doctor or expert system that made them, and why they were made.

Once the data have been classified in this way they are scrutinized by the expert system with a view to detecting states indicative of danger for the patient and incompatibilities indicative of instrument problems. The classified variable values are also used for evaluation of the patient's clinical subsystems and overall condition [49]. The results of trend analysis (static and dynamic parts) [21] are stored in memory elements of type **trend**, which have fields for the name of the variable in question and for a semiquantitative label indicating the overall trend (increasing, decreasing, or stable). The semiquantitative states and trends of variables and systems are used to screen for instrument errors or states of risk to the patient. This allows the detection of risk states not detected

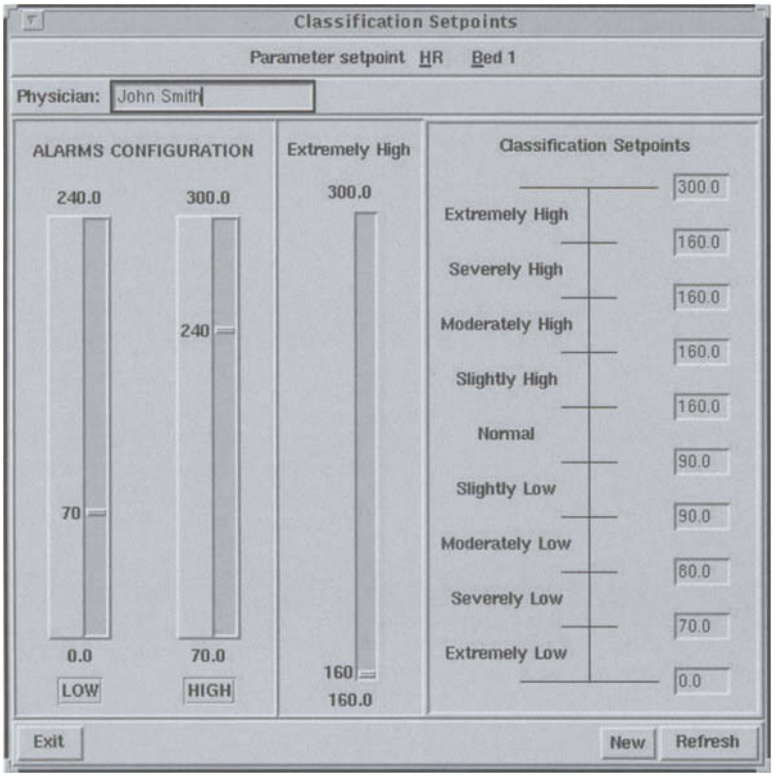


FIGURE 7 Manual configuration of threshold classification.

by conventional systems that only analyze numerical data. An example is the following rule for generating an hypovolemic shock alarm (for simplicity, only the natural language synopsis is shown):

IF severe Hypotension and diuresis=oligury
and CVP is low
THEN generate alarm: hypovolemic shock.
suggest: administer liquids until CVP>5

The generation of an alarm, at either the numerical or the semiquantitative stage, immediately affects the process of data acquisition. Relevant variables are acquired with increased frequency, and in the case of a warning of instrument error, signal analysis may be specifically requested as an alternative source for the questionable measured variable.

The screen of the PC monitoring system is shown in Fig. 8.

2. The Workstation-Based System

The central SUN workstation, which runs under UNIX, supports the medical reasoning expert systems, has graphics capacity for simultaneous reproduction of six bedside PC displays, mediates access to the Informix database, and performs any signal analyses that the bedside PCs have no time

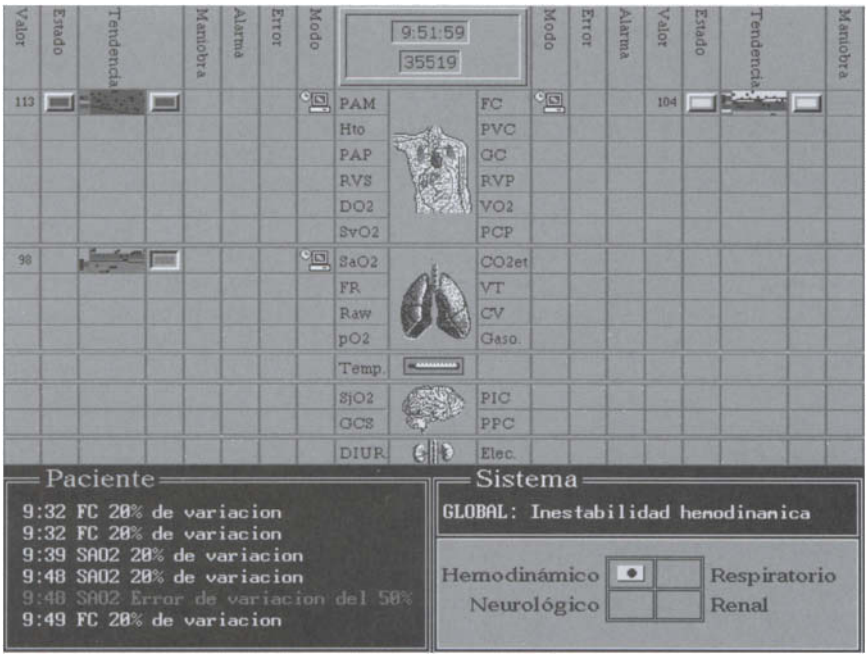


FIGURE 8 Screen of PC monitoring system.

for. All these activities are coordinated by a central management module. The central manager provides a wide spectrum of information distribution modes, ranging from the broadcasting of information to all active processes to the channeling of information from one specific process to another. In particular, in between these extremes, it allows information to be sent to all the processes dealing with a given patient that are capable of receiving information of the type in question; for example, warning messages generated by a bedside PC can be sent to all the expert systems currently analyzing the situation of the corresponding patient.

B. Database Structure

An interface, running in the database PC, mediates communication with the database and allows SQL queries to be passed to the database from any expert system (whether it be running in the workstation or at the bedside). The formulation of SQL queries by the user is facilitated by the user interface. In order to facilitate the use of externally stored data by the expert system, the tables of the external database have, in general, been defined in such a way as to echo the structure of the expert system's data elements; that is, the structure of the Informix database reproduces that of the set of expert systems. It contains tables recording, for each patient and at any given time, the monitoring devices in operation, the variables being monitored and their acquisition frequency, the current alarm and semiquantitative labeling thresholds, and so on (Fig. 9). One detail to be taken into account is that the entity "Connection" results from

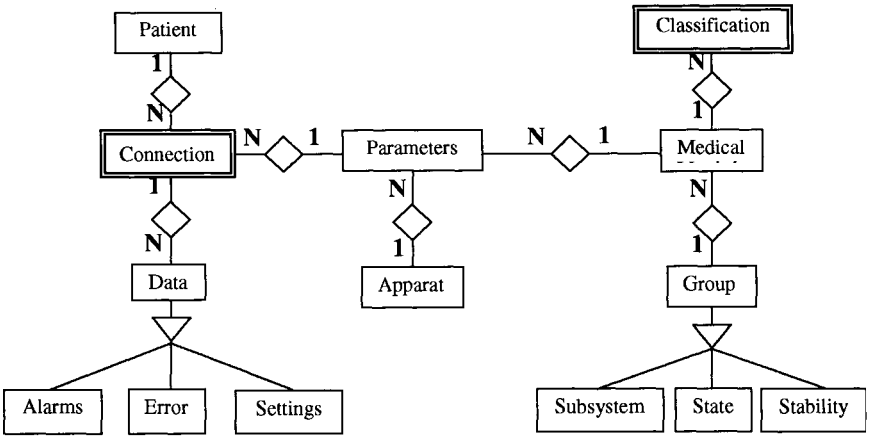


FIGURE 9 Database structure.

the relationship between "Patient" and the "Parameters" obtained from the patient.

All the information stored in the database contains an entry for the date and the hour of the analysis; this will allow us to maintain temporal restrictions in the expert system when we use in the reasoning information belonging to the working memory and to the database, including temporary information from both.

Since information such as radiographies, tomographies, magnetic resonances, and signals (ECG, EEG, capnography, etc.), in digital form, cannot be physically stored in the database (because the PC version of the Informix does not allow storage of binary data), a nominal indexation structure of data files has been created, thus making it possible to recover from the database the name of the file with the information that interests us at a given moment. The names of the files have been codified using the patients identification number, the date, the hour, and the type of information to avoid the possibility of files existing with the same names. In order to facilitate access to the database and to reduce network traffic, the image files are stored, in compressed format, in the workstation and not in the database server.

The resulting parameters of digital signal processing (ECG, EEG, capnography, etc.) are stored in files to analyze trends and stabilities; the instantaneous real values remain in the working memory, too, i.e., the HR at the moment, extracted from the ECG signal.

Another very important information stored in the database is all the data refering to the maneuvers that have been applied to the patient, the drugs that have been provided at each moment, commentaries and diagnosis given by the doctor, etc. All this allows us to observe, a posteriori, the patient's evolution and to analyze the reaction to the different treatments.

All the above information is available to all the medical reasoning expert systems running in the workstation, and a series of special functions makes access to this information totally transparent for anyone creating a new one. As was mentioned earlier, access to the database from the left-hand sides of expert

system rules (which would normally be prevented by the OPS/83 compiler) has been made possible by means of special C routines; there is no risk of inconsistency, because the data obtained from the database are all historical data not subject to change, and the expert systems will not modify the information they obtain in this way.

The database is endowed with the necessary multitasking capacity by the central manager, which receives all requests for access, orders them in accordance with the relative priorities of the requesting processes and the time elapsed since their reception, and passes them one by one to the database as soon as the latter is free. The central manager does not remain inactive while the database is processing a query; it attends to other business until a top-priority message from the database signals that the current query has been processed, whereupon the central manager channels the result of the query to the querying process and sends the next pending query to the database. This communication system between the database server and the central system follows the classic client–server scheme.

Also, a module inside the central system is in charge of executing, when it is not busy, the queries most required by the expert systems (trends of the variables, stabilities, etc, in general, information with a high latency time); then, when a query is required, the result can be found in the central system and it is not necesary to access the PC database.

C. Communication Mechanism. Database–Expert System

The OPS/83–Informix interface consists of a set of C functions (using embedded SQL [19,20]), allowing data to be downloaded to and retrieved from the RDB via a common block of memory shared by the two programs [10]. These functions fall into three groups (Fig. 10), corresponding to the three stages involved in data retrieval via the interface:

Group 1. Three functions for passing parameter values from the expert system to a parameter vector in the common block (one function for alphanumeric strings (BDsym), one for integers (BDint), and one for reals (BDreal));

Group 2. Two functions (readBD and otherBD) for passing to the database, via the common block, dynamic SQL commands and queries that use parameters previously introduced into the parameter vector (which they reset upon termination) and deposit their results, if any, in one of three results vectors likewise located in the common block; and

Group 3. Three functions (readBBDs, readBBDi, and readBBDr) for reading results from results vectors (three results vectors are provided so as to have room for results to remain in memory for possible rereading without the need for reexecution of the SQL query that produced them).

These C routines have been linked to the expert system by manipulating the assembler source code produced by the OPS/83 compiler, as follows. In the OPS/83 code for the expert system are included dummy functions with the same names and argument requirements as the C functions; once this code has been converted to assembler source code by the compiler, external calls to the C routines are plugged into the dummy functions. It is these external calls that

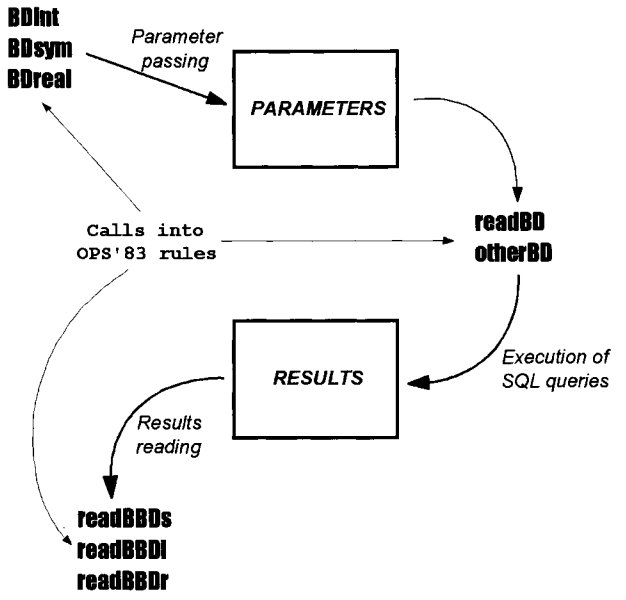


FIGURE 10 General functional scheme of the OPS/83-Informix interface.

would, in contexts such as the left-hand sides of rules, have been disallowed by the compiler if written directly in OPS/83 code.

The arguments of the parameter-passing functions BDsym, BDint, and BDreal are the parameters to be passed (Fig. 10); these functions return 0 if no error has occurred and 1 otherwise (the only cause of error for a group 1 function is a parameter vector overflow resulting from an attempt to pass too many parameters for a single SQL query; the limit implemented is 10, which is ample for our ICU system).

The function otherBD, which passes nonquerying SQL commands effecting data insertion, deletion, updating, etc., has three string arguments, each containing one of the segments into which the SQL command to be passed must be split. This admittedly messy way of passing the SQL command is made necessary by the OPS/83 limit of 128 on the number of characters in a string; the total of 384 characters allowed by the three arguments of otherBD suffices for the SQL commands used in our ICU system. The SQL query-passing function readBD has three string arguments fulfilling the same function as those of otherBD, and an additional integer argument specifying in which of the three results vectors is the results of the query to be stored. These functions return 0 if no error has occurred and an Informix error code otherwise (note that readBD returns an error code if a query has an ambiguous or multiple result).

The results-reading functions readBBDs, readBBDi, and readBBDr each have two arguments; the second specifies which of the three results vectors is addressed (0,1, or 2), and the first specifies which of the fields of that vector is to be read from. The values returned by these functions are of course query field values read from a results vector.

This interface is designed to return only one value from the queries because, due to the design of the knowledge base of the monitoring system, this only

Informix SQL table

```
CREATE TABLE PATIENT
        (PIN            CHAR (20),
        Name            CHAR (50),
        DateIn          DATE,
        BedNum          INTEGER,
        Age             INTEGER,
        UNIQUE (PIN) );
```

Expert System elements

```
type MPatient = element
        (PIN : symbol;
         Name : symbol;
         DateIn : real;
         BedNum : integer;
         Age : integer;
);
```

Normal SQL query

```
SELECT Age FROM Patient
WHERE PIN="12-3341-51A"
```

SQL query embedded into OPS/83

```
-- We suppose MPatient.PIN="12-3341-51A"
call BDsym (&MPatient.PIN);
&Error = call ReadBD (|SELECT Age|,
| FROM Patient|,| WHERE PIN=%s;|,0);
if (&Error=0) { &MPatient.Age=ReadBBDi(0,0));}
-- Now, we have recovered the age value
-- and stored it into the element MPatient
```

FIGURE 11 Execution secuence of a simple SQL query into OPS/83.

needs to recover concrete values within the database (the most recent value of X, the tendency of X, the number of alarms of type X, etc.).

A simple example of execution of a SQL query is shown in Fig. 11.

As an example of the use of the interface in a real rule implemented in our monitoring system, consider the following diagnostic rule for diuresis (Fig. 12).

In the second pattern we verify the value of the CVP (cardiovascular pressure). This parameter is continously measured; thus, it is never necessary to be obtained from the database as it will allways exist in the working memory.

In the third pattern of the left-hand side of this rule, the condition BDsym(@.PIN) = 0 stores the current patient's identity number in the common block and checks that this operation has been properly performed; the condition readBD(.....) = 0 instructs Informix to deposit, in results vector 0,

In natural language, the rule is:

IF

The immediate objective is to diagnose a diuresis, the CVP is LOW and diuresis=poliuria.

THEN

Diagnose: insipid diabetes.

This rule is implemented in OPS/83, as extended by the interface, as follows:

```
RULE DIAG_Diuresis_5
{
 &1 (Task objective=|diuresis_diagnose|; type=|BD|);
 &2 (Data VarName=|CVP|; Clasific=|L|);
 &3 (Patient ( BDsym(@.PIN) = 0 /\ readBD(|SELECT Clasific FROM DATA A WHERE PIN = %s AND
VarName = "DIURESIS" AND DAY = (SELECT| , |MAX(DAY) FROM DATA WHERE PIN = A.PIN AND
VarName = A.VarName) AND A.HOUR = (SELECT| , |MAX(HOUR) FROM DATA WHERE PIN = A.PIN AND
Var_Name = A.VarName AND DAY = A.DAY);| , 0) = 0 /\ readBBDi(0,0)=|P|));
->
OPSmsg (|Insipid diabetes|);
};
```

FIGURE 12 Example of a diuresis diagnostic rule.

the last station value recorded for that patient in the Informix table DATA (and checks that this operation has been properly performed); and the condition readBBDi(0,0) = |P| controls whether Poliuria is the current oliguria state value read by readBBDi from the first field of results vector 0. The %'s in the readBD command is a place-holding symbol; when reconstructing the desired SQL command from their three string arguments, readBD and otherBD recognize this symbol as a request for insertion of the next parameter in the parameter vector.

It is to be noted that although access to the database is clearly much slower than access to the elements stored inside the working memory, due to the existing latency between the acquisition of one physiological parameter and the following one (10–15 min), there is an existing problem at the time of making a request to the database. It has been proved that, on average, this request has an associated delay of 1 s, and in the worst case 3 s.

An example of using interface functions on the right-hand side of rules for updating an external database is shown in Fig. 13. All insertion rules (update included) are preceded by the execution of a screening rule, so that only valid data can enter the database.

As in the previous case, the arterial presure is always present in the working memory, so we do not need to access the database (second pattern). However, the ICP is a parameter that can be either in the working memory or in the database. In this case, we will try to find this information in the working

In natural language, the rule is:

IF

The immediate objective is to diagnose a craneoencephalic traumatism and the measured Arterial Pressure is high, and Intracraneal Pressure value is unknown (it does not exist in the working memory nor in the database).

THEN

A rescaling must be done in the higher limit of normality in the Arterial Pressure, setting the limit to 160. Apply a Beta-blockers treatment until normality.

This rule is implemented in OPS/83, as extended by the interface, as follows:

```
RULE DIAG_Craneo_Traum_2
{
 &1 (Task objective=|craneo_traum_diagnose|; type=|BD|);
 &2 (Data VarName=|AP|; Clasific=|H|);
 &3 (Data VarName=|ICP|; Clasific=NIL);
 &4 (Patient ((BDsym(@.PIN) = 0 Λ readBD(|SELECT Clasific, Day, Hour FROM DATA A WHERE PIN = %s
AND VarName = "ICP" AND Day = (SELECT| , |MAX(Day) FROM DATA WHERE PIN = A.PIN AND VarName
= A.VarName) AND A.Hour = (SELECT| , |MAX(Hour) FROM DATA WHERE PIN = A.PIN AND VarName =
A.VarName AND Day = A.Day);| , 0) <> 0) V SecondsOPS(readBBDi(1,0) , readBBDi(2,0)) < TimeOPS(-
1200));
->
call BDsym(&4.PIN);
if (otherBD (|UPDATE CLASIFIC SET LimitSnormal = 160 WHERE PIN=%s AND VarName="AP";|,||,||)<>0)
    {
     call errorOPS (|Err_Insert|);
    }
else call OPSmsg(|Apply a Beta-blockers treatment until normality);
};
```

FIGURE 13 Example of a diuresis diagnostic rule.

memory, and after, if we have not found it, the system will seek it in the database using the SQL query shown in Fig. 13. If the value is not present in the database or the value found is not valid (belonging to the past 20 min), then a rescaling must be done in the higher limit of normality in the arterial pressure, setting the limit to 160. This rescaling is made by an updating of the database classification table. It is to be noted that the last pattern that can be executed is the database query, so if any of the previous patterns do not match, the query will not be executed (using this strategy, the system will execute only the necessary queries).

REFERENCES

1. Aikins, J. S., Kunz, J. C., Shortlife, E. H., and Fallat, R. J. PUFF: An expert system for interpretation of pulmonary function Data. In *Readings in Medical Artificial Intelligence: The First Decade*. Addison-Wesley, Reading, MA, 1984.
2. Allen, J. F. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11): 832–843, 1983.
3. Al-Zobaidie, A., and Grimson, J. B. Expert systems and database systems: How can they serve each other? *Expert Systems* 4(1): 30–37, 1987.
4. Arcay, B., and Hernández, C. Adaptative monitoring in I.C.U.: Dynamic and contextual study. *J. Clin. Engrg.* 18(1): 67–74, 1993.
5. Arcay, B., Moret, V., Balsa, R., and Hernández, C. Physical and functional integration system for intellegent processing and priorization of variables in an ICU. In *Proc. 11th Int. Conf. IEEE EMBS*, Seattle, 1989.
6. Barnett, G., Cimino, J., *et al.* DXplain: An experimental diagnostic decission-support system. *J. Amer. Med. Assoc.* 257: 67.
7. Barr, A., and Feigenbaum, E. A. *The Handbook of Artificial Intelligence*, Vol. 1. Pitman, London, 1981.
8. Chittaro, L., Constantini, C., Guida, G., Tassa, C., and Toppano, E. Diagnosis based on cooperation of multiple knowledge sources. In *Proc. Second Generation Expert Systems Specialized Conf.* Avignon-France, pp. 19–33, May 29–Jun 2, 1989.
9. Dafonte, J. C., Taboada, J. A., and Arcay, B. Database based reasoning for real time monitoring and data analysis in intensive care units. *Expert Systems* 14(4): 190–198, 1997.
10. Dafonte, J. C., Arcay, B., Bóveda, C., and Taboada, J. A. Intelligent management of a relational database in a perinatal monitoring system. *J. Clin. Engrg.* 24(1): 34–40, Jan.–Feb. 1999.
11. Dawant, B. M., Uckun, S., Manders, E. J., and Linstrom, D. P. The SIMON Project. *IEEE Engrg. Med. Biol.* 12(4): 82–91, 1993.
12. Filman, R. E. Reasoning with worlds and truth maintenance in a knowledge-based programing enviroment. *Commun. ACM* 31 Apr. 1988.
13. Forgy, C. L. *The OPS/83 User's Manual*, Sistem Version 2.2. Production Systems Technologies, 1986.
14. Franklin, D. F., and Ostler D. V. The P1073 Medical Information Bus. *IEEE Micro* 52–60, Oct. 1989.
15. Galán, R., and Martínez, G. Sistemas de Control Basados en Reglas. In *Curso de Control Inteligente de Procesos* (A. Jiménez Avello, R. Galán López, R. Sanz Bravo, and J. R. Velasco Pérez, Eds.). Univ. Politécnica de Madrid, ETS de Ingenieros Industriales. Sección de Publicaciones. [In Spanish.]
16. Gardner, R. M. Computerized data management and decision making in critical care. *Surgical Clinics N. Amer.* 65(4): 1041–1051, 1985.
17. Hayes-Roth, B., Washington, R., Ash, D., Hewett, R., Collinot, A., Vina, A., and Seiver, A. Guardian: A prototype intelligent agent for intensive-care monitoring. *Artif. Intell. Med.* 4: 165–185, 1992.
18. Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. *Building Expert Systems*. Addison–Wesley, Reading, MA, 1983.
19. Informix Software Inc. *Informix-SE for DOS*, Administrator's Guide v.4.1, 1992.

20. Informix Software Inc. *Informix-ESOL/C for DOS*, Programer's Manual v.4.1, 1992.
21. Kalli, S., and Sztipanovits, J. Model-based approach in intelligent patient monitoring. In *Proc 10th Int. Conf. IEEE EMBS*, New Orleans, pp. 1262–1263, 1988.
22. Kampman J., Hernandez C., and Schwarzer, E. Process control in intensive care: An integrated working place. In *Proc 11th Int Conf IEEE EMBS*, Seattle, pp. 1215–1216, 1989.
23. Laffey, J. L. "El experto en tiempo real. Binary", Junio, pp. 110–115, 1991.
24. Larsson, J. E., and Hayes-Roth, B. Guardian: A prototype intelligent agent for intensive-care monitoring. *IEEE Intell. Systems* 58–64, 1998.
25. Leung, K. S., Wong, M. H., and Lam, W. A fuzzy expert database system. *Data Knowledge Engrg.* 4(4): 287–304, 1989.
26. Mac Dermott, D. A termporal logic for reasoning about processes and plans. *Cognitive Sci.* 6: 101–155, 1982.
27. Marín Morales, R. *Razonamiento temporal Aproximado*. Univ. Murcia, Spain, 1995.
28. Miller, P. *Expert Critiquing Systems: Practice-Based Medical Consultation by Computer*. Springer-Verlag, New York, 1986.
29. Miller, R., McNeil, M., Challinor, S., *et al.* INTERNIST-1/Quick medical reference project: Status report. *Western J. Med.* 145: 816–822, 1986.
30. Miller, R., Pople, Jr., H., and Myers, R. A. INTERNIST-1: An experimental computer—based diagnostic consultant for general internal medicine. *New Engl. J. Med.* 307: 468–476, 1982.
31. Minker, J. Deductive databases: An overview of some alternative theories. In *Methodologies for Intelligent Systems*. North-Holland, Amsterdam, 1987.
32. Mora, F. A., Passariello, G., Carrault, G., and Le Pichon, J. Intelligent patient monitoring and management systems: A review. *IEEE Engrg. Med. Biol.* 12(4): 23–33, 1993.
33. Nilsson N. J. Principles of Artificial Intelligence. Tioga Publishing Company, Palo Alto, CA, 1980.
34. Paul, C. J., Acharya, A., Black, B., and Strosnider, J. K. Reducing problem-solving variance to improve predictability. Communications of the ACM, 34(8): 80–93, 1991.
35. Perkins, W. A., and Austin, A. Adding temporal reasoning to expert-system-building environments. *IEEE Expert* 5(1): 23–30, 1990.
36. Pryor, T. A. *et. al.* The HELP system. *J. Med. Systems* 7: 87, 1983.
37. Raphael, B. *The Thinking Computer: Mind Inside Matter*. W H Freeman, San Francisco, 1976.
38. Reggia, J., Perricone, B., Nau, D., and Peng, Y. Answer justification in diagnosis expert systems. *IEEE Trans. Biol. Med. Engrg.* 32: 263–272, 1985.
39. Schecke, Th., Rau, G., Popp, H., Kasmacher, H., Kalff, G., and Zimmermann, H. A knowledge-based approach to intelligent alarms in anesthesia. *Engrg. Med. Biol.* M, 38–43, 1991.
40. Shortliffe, E. H. *Computer-Based Medical Consultations: MYCIN*. Elsevier/North Holland, New York, 1976.
41. Shortliffe, E. H., Scott, A. C., Bischoff, M. B., Campbell, A. B., and Van Melle, W. ONCOCIN: An expert system for oncology protocol management. In *Proc. IICAI-81*, p. 876, 1981.
42. Sierra, C. A. Estructuras de control en inteligencia artificial: Arquitecturas reflexivas y de metanivel. In *Nuevas Tendencias en Inteligencia Artificial*, Universidad Deusto, 1992. [In Spanish.]
43. Simons, G. L. *Introducción a la Inteligencia Artificial*. Diaz de Santos, Madrid, 1987. [In Spanish.]
44. Stankovic, J. A., and Ramamritham, K. *Hard Real-Time Systems: A Tutorial*. Comput. Soc. Press, Washington, DC, 1988.
45. Steiner, A., Kobler, A., and Norrie, M. C. OMS/Java: Model extensibility of OODBMS for advanced application domains. In *Proc. 10th Int. Conf. on Advanced Information Systems Engineering*, 1998.
46. Taboada, J. A., Arcay, B., and Arias, J. E. Real time monitoring and analysis via the medical information bus. Part I. *J. Internat. Fed. Med. Biol. Engrg. Comput.* 35(5): 528–534, 1998.
47. Taboada, J. A., Arcay, B., and Arias, J. E. Real time monitoring and analysis via the medical information bus. Part II. *J. Internat. Fed. Med. Biol. Engrg. Comput.* 35(5): 535–539, 1998.
48. Tsang, E. P. K. Time structures for AI. In *Proc. Int Joint Conf. Artificial Intelligence*, pp. 456–461, 1987.

49. Uckun, S., Lindstrom, D. P., Manders, E. J., and Dawant, B. M. Using models of physilogy for intelligent patient monitoring. In *Proc. 13th Conf. IEEE EMBS*, pp. 1308–1309, 1991.
50. Walley, P. Measures of uncertainty in expert systems. *Artif. Intell.* 83: 1–58, 1996.
51. Wiederhold, G., Barsalou, T., Lee, B. S., Siambela, N., and Sujansky, W. Use of relational storage and a semantic model to generate objects: The PENGUIN Project. In *Database 91: Merging Policy, Standards and Technology, The Armed Forces Communications and Electronics Association*, Fairfax VA, pp. 503–515, June 1991.
52. Winston, P. H. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1979.

24

WIRELESS ASYNCHRONOUS TRANSFER MODE (ATM) IN DATA NETWORKS FOR MOBILE SYSTEMS

C. APOSTOLAS

Network Communications Laboratory, Department of Informatics, University of Athens, Panepistimioupolis, Athens 15784, Greece

G. SFIKAS

R. TAFAZOLLI

Mobile Communications Research Group, Center for Communication Systems Research, University of Surrey, Guildford, Surrey GU2 5XH, England

Database and Data Communication Network Systems, Vol. 3

I. INTRODUCTION

The concept of wireless asynchronous transfer mode (WATM) Networks has been a topic of great research interest. The establishment of the WATM group [1] within the ATM Forum in 1996, the work carried out by EU ACTS projects such as WAND, SAMBA, and MEDIAN [2,3], and the initiative on the broadband radio access network (BRAN) [4] within ETSI justify the significance of deploying a wireless ATM network to carry multimedia traffic with support of terminal mobility or portability.

ATM networks offer certain advantages when compared to traditional LAN technologies like Ethernet and FDDI. Adopting asynchronous time division multiplexing (ATDM), they provide quality of service (QoS) guarantees and efficient support of different application types over the same physical medium. Moreover, wireless access eases relocation of terminals and expansion of networks, and enables user mobility and equipment portability. Furthermore, the wireless technology is the enabling factor for universal personal communications. Existing cellular systems deal mainly with voice communication or offer supplementary low-bit-rate data services over a network optimally designed for voice. Moreover, DECT [5] does not offer the high transmission rate required for broadband multimedia communications, and standards like IEEE802.11 [6] and HIPERLAN [7] support high bit rates but they do not provide for mobility. Furthermore, they do not offer QoS guarantees. However, the merging of wireless and ATM technologies could lead to universal personal multimedia communications. This concept is envisioned by the emerging third generation communication systems like UMTS [8], IMT2000 [9], and BRAN.

This chapter provides the migration path from existing fixed ATM LAN standards to ATM wireless LAN (ATM WLAN), dealing mainly with the functions related to radio link access, the implications on transport protocols, and the support of user mobility. ATM WLAN is envisioned as a system of densely populated wireless access parts with small coverage areas and an ATM-oriented backbone network that could span in a geographical area, justifying the term LAN. Of course, some of the solutions presented could be adopted in wide area networks too.

To establish the above-mentioned migration path, one must take into account the characteristics and requirements of the services supported in ATM networks. The same service characteristics must be supported in the ATM WLAN. Moreover, the protocol reference model (PRM) of conventional ATM

LANs must be considered to identify the required additions and modifications, and to set the design objectives for the ATM WLAN PRM.

The definition of the latter is influenced by the study of existing standards for wireless communications. In this chapter, HIPERLAN [7] is investigated because it targets the same market as ATM WLAN, namely local area wireless networking. Its architecture and protocols are outlined in Section V. Moreover, it is examined whether HIPERLAN offers a multiple access control (MAC) scheme that could be exploited to support the ATM WLAN concept. It is found that encapsulation of ATM cells (packets) in HIPERLAN packets combined with the HIPERLAN-distributed multiple-access scheme results in low transmission efficiency and the inability to offer QoS guarantees. Moreover, ATM networks are characterized by point-to-point links leading to star-based topologies, at least for the access part of the network. This issue affects the selection of the multiple access scheme for ATM WLAN and favors centralized adaptive schemes. To assist the selection of the multiple access mechanism in ATM WLAN, the performance of adaptive TDMA is compared to that of HIPERLAN MAC. It is found that the adaptive TDMA scheme outperforms HIPERLAN MAC in terms of user capacity.

Moreover, adaptive TDMA is capable of applying traffic policing functions on the traffic of the air interface. This means that QoS provisioning is enabled. Then the concept of traffic monitoring and QoS provision as included in ATM networks is inherited by ATM WLAN. Furthermore, this chapter refers to the issues affecting the selection of the proposed MAC scheme and shows how ATM traffic classes are mapped onto ATM WLAN MAC traffic. Moreover, it verifies the capability of the proposed MAC scheme to preserve the bandwidth of connections and guarantee QoS requirements.

However, it is still important to investigate the interaction between the proposed multiple access scheme and the transport layer supporting user applications. Thus, the performance of the transport control protocol (TCP) [10] over the adopted MAC scheme for the ATM WLAN is investigated. TCP is the main transport layer protocol for Internet applications. ATM WLAN should provide for TCP support, and deal with the problem of TCP throughput degradation. The latter arises because of congestion and ATM buffer overflows when TCP resides on top of the ATM PRM, or because of TCP segment losses when TCP is used over wireless links. In this chapter it is shown that by adopting an ATM cell retransmission policy the throughput of the TCP layer can approach the maximum achievable. Moreover, by implementing Early packet discard [11] in the MAC of ATM WLAN, the problem of TCP throughput degradation when the network is driven in congestion is alleviated.

II. SERVICES IN ATM WLAN

ATM WLAN will be able to integrate both fixed and wireless user–network interfaces (UNIs) in the same backbone ATM-oriented network. Under this assumption it is expected to come across with a design that will support all the applications and all the traffic classes specified in fixed ATM networks [12].

TABLE 1 Services to be Supported in ATM WLAN

Application	Bit rate (Kbps)	BER	PER	Call attempts in small business (50 users)	Call attempts in medium business (150 users)	Call attempts in large business (4700 users)
Telephony	32, 64	10^{-7}	10^{-3}	5	40	200
Videophone	64–2000	1.3×10^{-6}	8×10^{-6}	1.5	10	50
Video conference	5000	1.8×10^{-6}	5×10^{-6}	1	1	1
MPEG1	1500	2.5×10^{-6}	9.5×10^{-6}	0.5	3	10
MPEG2	10000	1.5×10^{-6}	4×10^{-6}	0.5	3	10
Hi-Fi distribution	400	10^{-5}	10^{-7}	0.9	0.9	0.9
Low-speed file	64	10^{-7}	10^{-6}	1	10	50
High-speed file	2000	10^{-7}	10^{-6}	3	20	100

Briefly, the aim is to support telephony, fast file transfers, distributed computing, and multimedia on the desktop. The bit rate requirements for these services combined with the expected population of ATM WLAN customers define the required transfer rates in the ATM WLAN. Moreover, the QoS requirements for the above-mentioned application classes influence the mechanisms related to the bandwidth allocation and error control.

In Table 1 [13] the average bit rate and the bit and packet error rate (BER, PER) requirements for some applications are presented. Moreover, the number of call attempts per user during the busy hour is also shown. In the scope of the ATM LAN (fixed or wireless), the term call implies not only a telephony service but an established connection (or set of connections) carrying traffic for some application. Based on the average bit rate requirements and the number of call attempts in Table 1, and under the assumption that one wireless module of ATM WLAN will service one small business (considering user population), the available bandwidth on the air interface of ATM WLAN should be around 20 Mbps.

To have a complete view of the service requirements, Table 2 shows service end-to-end delay requirements and their jitter requirements (i.e., the difference between the actual interarrival times of two subsequent packets of a connection at two points of the network and those expected).

TABLE 2 Timing Requirements of Some of the Supported Services

Application	End-to-end delay (ms)	End-to-end packet delay variation (ms)
20 Mbps HDTV video	0.8	1
Telephony with echo canceler	<500	130
Videophone (64 Kbps)	300	130
MPEG1 (NTSC)	5	6.5

III. FIXED ATM LAN CONCEPT

ATM WLAN is expected to have similar architecture and protocol family and support the same set of services as that of fixed ATM LAN. Thus, the former should inherit protocols and mechanisms similar to those residing in the latter, of course modified to adapt to the wireless and mobility-enhancing environment.

A. Topology and Defined Network Interfaces

The protocol architecture and the topology of a fixed ATM LAN are inherited from the B-ISDN protocol reference model and architecture [14]. Figure 1 presents a topology scenario of an ATM LAN and shows the different signaling protocols executed among the network entities. Three main interfaces are defined, namely public UNI, private UNI, and private network-to-network interface (P-NNI). Public UNI is defined between customer premises equipment (CPE) and an ATM switch or multiplexer in a public domain ATM network. Private UNI is defined between a user terminal and an ATM switch of the same private ATM LAN. Finally, P-NNI is defined among the switches that form the core network of a private ATM LAN.

Private and public UNI specifications cover:

- the electrical standards used to interconnect two ATM devices,
- the signaling and the functions related to call setup,

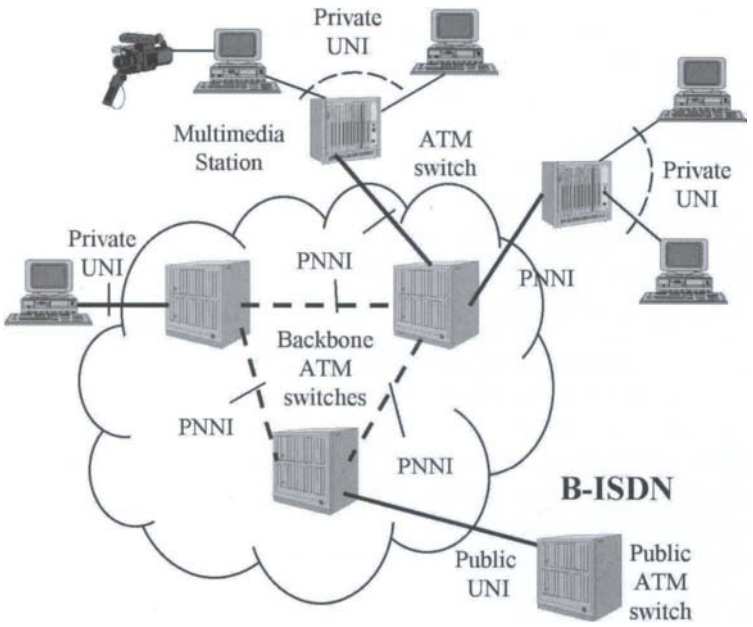


FIGURE 1 Topology paradigm for fixed ATM LAN.

- the signaling, the databases, and the operations required for the management of an ATM LAN at UNI (address allocation to terminals, definition of an allowable number of established connections, specification of the electrical interface, etc.), and
- the functions that monitor the traffic generated by the user terminals, avoid congestion, and allocate bandwidth efficiently while satisfying the QoS of the active calls.

There are certain key components of the UNI functionality affecting the design of an ATM WLAN, and they will be presented later in this section. At this point, one could note that the design of an ATM WLAN requires the modification of some UNI specifications (starting from the physical medium) to come across with a wireless or mobile UNI implementation (W-UNI).

Furthermore, P-NNI specifications cover:

- the organization of ATM switches into logical nodes and layers of an hierarchical backbone network structure in terms of addressing and routing,
- the functions and the signaling related to the topology updates within the backbone part of the ATM LAN,
- the exchanged signaling and the procedures for the call routing during the setup phase, and
- the functions, database specifications, and signaling related to the backbone network management.

In Section VIII it is shown that P-NNI protocols can be adapted to support mobility management, in-call connection rerouting, and call setup rerouting because of mobility.

B. Protocol Reference Model of ATM LAN

Figure 2 presents the PRM of the fixed ATM LAN adopted from the B-ISDN equivalent model [14]. This model applies to both UNI and NNI within an ATM LAN. Layer management deals with the control and maintenance of every horizontal layer independently, and for this purpose peer-to-peer messages have been defined in every layer. Moreover, the corresponding operation administration and maintenance (OAM) functions of that layer are also defined [14]. Plane management is related to the control of the protocol reference model in a vertical manner, by providing to every layer information related to other layers. This happens by defining a higher-layer protocol called interim layer management interface (ILMI), which is responsible for the maintenance of information elements related to all the layers of the PRM, as well as for the exchange of these element values among the network entities. C-plane contains the protocols related to call establishment and termination, while U-plane is responsible for carrying and monitoring the traffic generated by user applications.

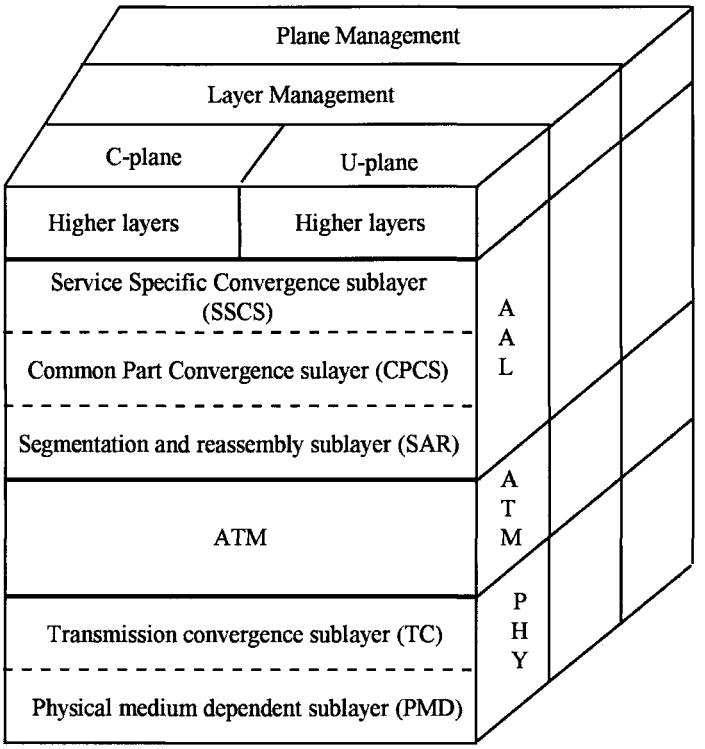


FIGURE 2 Protocol reference model (PRM) of an ATM LAN.

1. Physical Layer

The physical layer exchanges with the ATM layer packets of fixed length and structure called cells. The physical layer consists of two sublayers:

- Transmission convergence sublayer, which deals with error correction at the cell header, construction or recovery of frame structures that may exist in the transmission link (e.g., SDH), scrambling of the cell payload contents, and cell delineation (i.e., identification of cell boundaries within transmitted frames), and
- Physical medium-dependent sublayer which deals with the transmission issues (e.g., modulation, line coding, bit synchronization).

At the moment, all physical layer specifications define fixed transmission mediums (fiber optics, twisted pairs, and coaxial cable [14]). The links are point-to-point, as shown in Fig. 1, and there is no need for multiple access algorithms, like those used in traditional LANs (Ethernet, Token ring, FDDI), because different transmission directions are physically separated (different cable is used for each direction). However, in the case of an ATM WLAN the broadcast nature of radio transmission requires the implementation of a MAC scheme to regulate the allocation of the radio channel.

2. ATM Layer

The ATM layer is the heart of the ATM LAN PRM. All the main functions that characterize an ATM-oriented network reside in this layer. The ATM layer exchanges with the ATM adaptation layer (AAL) fixed size (48 bytes) service data units that form the ATM cell payload.

Moreover, the ATM layer performs bandwidth allocation and statistical multiplexing. Bandwidth allocation is based on the ATDM concept. Generally, in ATM networks bandwidth allocation is performed over time. In every link there is a maximum transfer capability expressed in cells per second (cells/s) that can be viewed as a continuous transmission of cell slots. The filling of cell slots by transmitted cells is performed asynchronously and on a demand basis. There are no predefined time slots allocated to a connection. The straightforward result is that the bandwidth allocation mechanisms can follow the traffic fluctuations of VBR connections. Moreover, bandwidth is not reserved for connections when it is not required, but it can be shared among them. Assuming that the instantaneous total required bandwidth does not exceed the total link capacity or the ATM switch capacity, ATM can support more connections using the same total bandwidth in the network as fixed bandwidth allocation mechanisms. This is called statistical multiplexing gain, and it means a better bandwidth exploitation than fixed bandwidth allocation. Thus, several new applications can be supported in a bandwidth-efficient way, increasing the network capacity and possibly reducing the service cost.

However, bandwidth allocation occurs not only in the links among network entities but also in the ATM switches in terms of buffer space and bus-bandwidth reservation. As ATM cells flow from the user terminals deep in the backbone network, the total traffic offered in a switch could overload it (or some of its ports). Then the network is driven into congestion locally and neighboring switches are affected too. To avoid such situations, traffic monitoring and control mechanisms have been introduced in the ATM layer to prevent and react to congestion [12,15]. The main idea is that for every new connection there is a call setup phase when the source specifies its QoS requirements and traffic pattern, and negotiates with the network about its traffic limits. To prevent congestion, the network, executing the Call Admission Control algorithm, decides to accept or reject the new call (based on the expected satisfaction of the QoS parameters of the established calls and the new call) and defines the traffic limits of the sources of an accepted call. These traffic limits are fed into the traffic policing functions. For every generated cell in a connection these functions decide whether the cell could drive the network to congestion (because its source exceeds its traffic limits), and whether it will be tagged as redundant. When congestion occurs (or is about to occur), tagged cells are discarded.

In the case of ATM WLAN, the traffic monitoring functions are of great importance, because the radio channel is a limited and expensive resource. Moreover, it is very important to monitor the allocation of radio channel capacity as it has a direct impact on the QoS of the established connections. The access of the radio link by a source must take place according to the rate limits of the traffic source and without violating the bandwidth that should be

allocated to other established connections. As presented in Section V.D, this mechanism is not adopted in HIPERLAN where bandwidth violations could occur. However, the multiple-access mechanism described in this chapter for ATM WLAN can be based on the monitoring of the bandwidth allocation requests and on a negotiated traffic contract.

3. C-Plane

In the higher layers the main component of this plane is the Q.2931 [14] protocol inherited from the ISDN technology. It defines the signaling that takes place among network entities to initiate and terminate connections, to monitor the status of calls, and to increase or decrease the number of communicating parties in point-to-multipoint connections. Currently, Q.2931 supports point-to-point and point-to-multipoint connections only. Moreover, it supports the establishment of permanent or semi-permanent virtual connections (PVCs), as well as the establishment of switched virtual connections (SVCs). In the AAL the C-plane functions are carried out by the signaling AAL (SAAL), a connectionless transport protocol presented in [16].

4. Plane Management

While Q.2931 provides the required signaling for call control (setup, termination, monitoring), ILMI [14] defines the procedures and the related signaling required for the management of the UNI. ILMI defines a set of managed entities that describe a UNI and organizes them into a database called a management information base (MIB). This database is distributed at the user side and network side of the UNI and provides configuration and status information related to the specific UNI. The protocol used to carry the required signaling between the user and network side of the UNI, and interrogate or modify the MIB is the simple network management protocol (SNMP) [17] residing on top of SAAL. Moreover, ILMI specifies the address registration procedure for a terminal. This procedure could be used in an ATM WLAN too, for handoff and location update procedures.

C. Traffic Management in ATM Networks

Traffic management in ATM networks includes all the functions used to prevent or react to congestion. A subset of the traffic management functions is the traffic policing mechanisms (or usage parameter control) that protect the network resources from malicious or unintentional misbehavior by traffic sources that could affect the QoS of the established connections.

1. Service Categories in ATM Networks

According to [12], ATM networks support five service categories (CBR, real-time VBR, non-real-time VBR, UBR, and ABR). These service categories relate traffic characteristics and QoS requirements to network behavior.

CBR service is intended to support real-time applications with tightly constrained delay and delay variation. All the ATM cells transmitted at a rate up to the specified peak cell rate (PCR) should be considered conforming and experiencing the negotiated QoS. The source can emit ATM cells at the PCR at

any time and for any duration. Like CBR, the real-time VBR service category is intended for applications with tightly constrained delay and delay variation. However, in this case, if traffic is emitted at the PCR for a long period the traffic conformance is violated. The non-real-time VBR service category is intended for applications without hard delay requirements. Unlike real-time VBR, there are no guarantees for delay constraints. Only the expected CLR is specified. The UBR service category is used for non-real-time applications. In the UBR case, no QoS parameter is specified. Moreover, no traffic specification must be provided. Finally, the ABR service category is intended for applications without tight delay requirements. It is the only service category where the traffic specification can change during the call. This is achieved through a flow control mechanism feeding information in the reverse direction, toward the traffic source. The latter is expected to use the feedback information to adjust its transmission rate and experience low CLR.

Whenever a new connection must be established, its service category must be defined. Moreover, a traffic contract between the traffic source and the network part must be agreed. The traffic contract consists of a traffic specification and a QoS commitment.

The traffic specification contains the values of the parameters characterizing a connection of a specific service class. These parameters are PCR, sustainable cell rate (SCR), minimum cell rate (MCR), maximum burst size (MBS), and cell delay variation tolerance (CDVT).

The QoS commitment contains the values for the QoS parameters associated with the service category of the established connection. The following QoS parameters are defined in ATM: peak-to-peak cell delay variation, maximum cell transfer delay, cell loss ratio, cell error ratio, severely errored cell block ratio and cell misinsertion rate. The former three parameters are negotiated while the other three are not. The QoS commitment means that the QoS parameters will be satisfied for the ATM cells emitted according to the traffic specification. The traffic policing functions monitor the ATM cell stream of a connection and identify those cells possibly violating the agreed traffic specification.

2. Traffic Policing Functions in ATM Networks

The traffic specification of a connection is provided for two reasons. First to provide the network with the necessary information to decide whether the new call can be accepted. Based on the traffic specifications of the new call and the already established calls, the network predicts whether it will be able to satisfy the QoS of all the above-mentioned calls if it accepts the new call. Second, the traffic specification parameters are fed in the mechanisms that monitor the traffic activity of a connection and report bandwidth violations. The latter occur as an excessive transmission of ATM cells above the PCR, or above the PCR for a duration longer than a well-defined period. The above-mentioned monitoring and reporting mechanisms are realized by the generic cell rate algorithm (GCRA), which resides in the network part of the UNI. GCRA is an algorithm that applies in a stream of transmitted ATM cells and characterizes them as conforming or nonconforming (with respect to a certain traffic contract). The characterization is based on the difference between the ATM cell actual and expected arrival times.

IV. MIGRATION FROM ATM LAN TO ATM WLAN

A. Definition for a New PHY Layer Based on Radio

The selection of a radio channel as the transmission medium in the ATM WLAN makes the first difference from that of an ATM LAN. In the latter the transmission medium is highly reliable so the procedures handling transmission errors are minimum and confined in a single-error correction/multiple-error detection capability, and this happens only for the header of the ATM cells. Opposite to fiber optics and cables (used in ATM LANs), the radio channel is a rather hostile transmission medium especially when the terminal moves, or the propagation environment changes because of movements. Finally, shadowing could obstruct significantly the transmission and result in a loss of communication link. This has a straightforward impact on the performance of the upper layers of an ATM WLAN PRM as shown in Section VII.D. The necessary mechanisms must be adapted to alleviate the effects of channel errors and improve the performance of an ATM WLAN.

B. Requirement for MAC Layer Definition in the ATM WLAN PRM

Another basic difference between an ATM LAN and an ATM WLAN is that the former does not contain any broadcast medium. All the entities are connected point-to-point using cables; thus there are no MAC functions and thus no MAC layer in the PRM of an ATM LAN. There is only one transmitter and one receiver for each direction at every interface, and different cables are used for the different directions. However, in the case of radio, its broadcast nature imposes the necessity of a MAC layer within the PRM of an ATM WLAN. Applying the MAC functions, logical links between communicating entities can be established. The MAC technique should enable the bandwidth allocation based on ATDM to take advantage of the discontinuous data generation that characterizes voice and file transfers and be able to follow the traffic fluctuation of VBR sources like video. Moreover, it should apply traffic monitoring operations, like those residing in fixed ATM networks to prevent congestion and enhance QoS provision.

C. Extension of C-Plane Protocols to Support User Mobility

An ATM LAN does not need to support roaming of its users, so there is no provision for mobility management in its protocol family. Mobility is an advantage of wireless networks and should be supported in an ATM WLAN. Thus, the protocols residing in an ATM LAN must be modified or extended to support mobility management issues like location update, connection rerouting, and call setup rerouting. Section VIII describes how existing protocols of the ATM LAN PRM are modified to cope with mobility management within an ATM WLAN.

D. ATM WLAN Architecture and PRM

Following the discussion in the previous section, Fig. 3 shows the envisioned system architecture for an ATM WLAN. Figures 2 and 3 show similar system

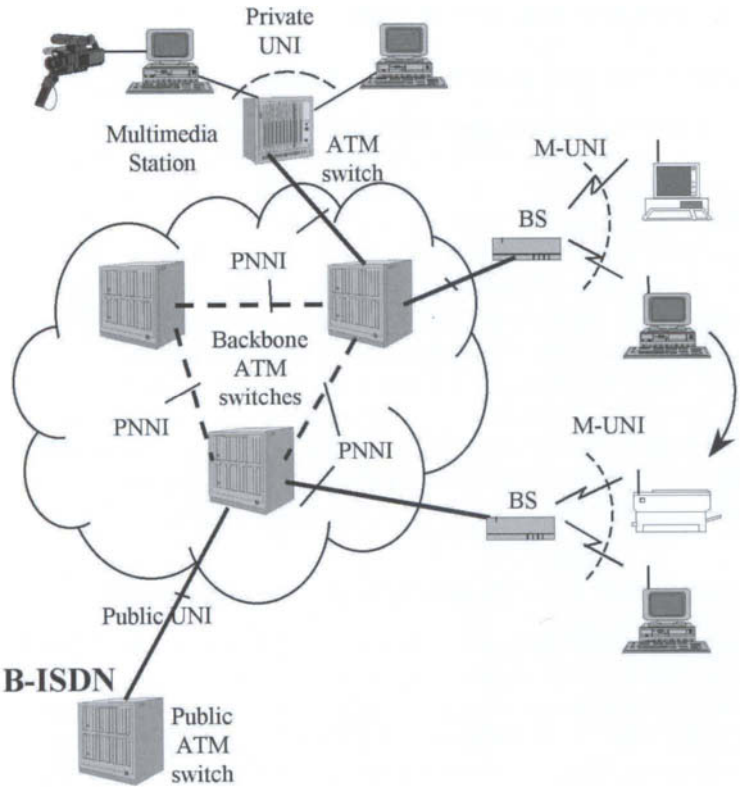


FIGURE 3 Topology paradigm for an ATM WLAN.

architectures in terms of topology and network entities. Small ATM switches at the access part of the ATM LAN have been replaced by ATM base stations (BSs), which, however, inherit most of the functions met in ATM switches. Moreover, backbone ATM switches in an ATM WLAN are enhanced with protocols supporting mobility. While their architecture does not have to be changed, their protocols of the C-plane must be extended. Moreover, an ATM WLAN does not support only wireless terminals but integrates fixed and wireless terminals using the same backbone network.

Figure 4 presents the ATM WLAN PRM, where it is shown that MAC and ATM functions are integrated in a common layer called ATM–MAC. The multiple access scheme of ATM WLAN must provide high bandwidth utilization and satisfy the criteria outlined in Section IV.B. This multiplexing protocol could be totally distributed like that used in a HIPERLAN, or it could allow BS to have the control of terminal transmissions (centralized approach) as occurs in DECT [18]. Then dynamic bandwidth allocation could be based on either CDMA or a dynamic TDMA scheme, since both techniques are used in cellular systems and exploit the presence of base stations to regulate the traffic on the air interface.

CDMA has been used in existing cellular systems and adopted for third generation personal communication networks, including UMTS, to support bit rates up to 2 Mbps [25]. It is applied over spread spectrum signals [26] by assigning different sequences of digital pulses (chips) per bit to different pairs of

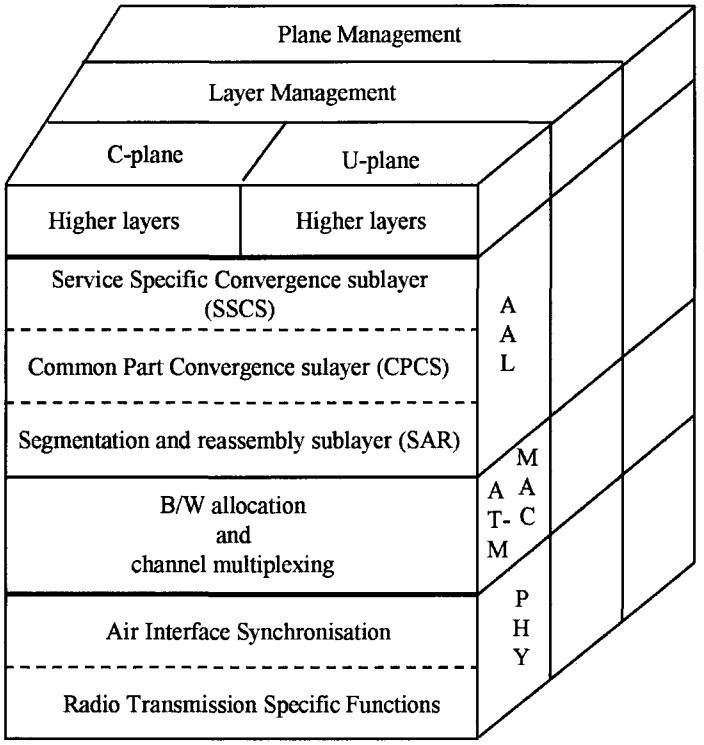


FIGURE 4 Protocol reference model (PRM) of an ATM WLAN.

transmitters and receivers, provided that for a given pair the same chip sequence (code) corresponds to every information bit. Also, the code sequences assigned to different channels must be mutually orthogonal. Following this convention, both a frequency band and a code define a channel.

In CDMA, the number of users that can be accommodated is proportional to the spreading factor (the ratio of spread spectrum signal bandwidth to the bit rate offered) [21]. Then the required system bandwidth should be increased compared to the channel gross bit rate by a factor equal to the spreading factor (usually on the order of 100) and this makes CDMA prohibitive for the bit rates required in an ATM WLAN.

Moreover, all the signals at the BS receiver should arrive with equal power. For this reason, power control must be applied; otherwise, the power of the received signals will depend on the distance between transmitter and receiver. Then the transmitters of undesired signals close to the receiver interfere strongly with the desired signal and prevent proper communication, leading to a reduction of system capacity [22].

A dynamic form of TDMA looks like the optimum solution for the multiple-access scheme of an ATM WLAN. However, a HIPERLAN being the latest development in the area of wireless networking introduces a distributed multiple-access scheme aiming to support multimedia traffic on the radio channel. In the following section it is examined whether this multiple-access scheme of a HIPERLAN copes with ATM traffic and satisfies the requirements of ATM WLAN MAC.

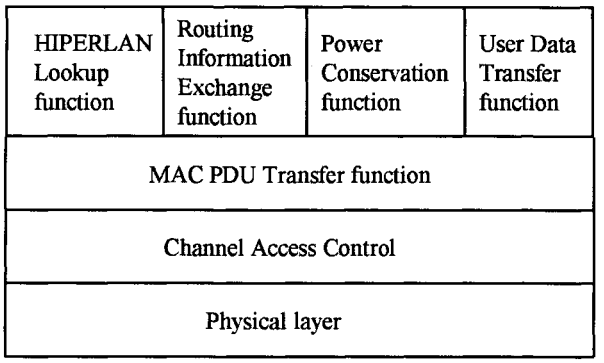


FIGURE 5 HIPERLAN reference model.

V. HIPERLAN, A CANDIDATE SOLUTION FOR AN ATM WLAN

HIPERLAN is a European standard for a high-performance radio local area network. The standard was prepared by the Radio Equipment and Systems (RES-10) Group of the European Telecommunications Standards Institute (ETSI) and aims to cover the physical and MAC layers of the OSI architecture. As such, it is a proposal for the implementation of the two lowest layers of the ATM WLAN PRM. The main objectives of the standard are:

1. The easy establishment of wireless ad hoc LANs, by using distributed topology and routing functions in the MAC layer, and
2. The transparent support of existing applications and protocols residing in layers above the MAC layer, by defining a MAC service access point (SAP) fully compatible to ISO 15802-1 specifications.

Therefore, a HIPERLAN can provide a wireless platform that offers a connectionless mode of packet transmission and supports existing protocols in the data link, network, and transport layers of the OSI architecture.

The protocol reference model of a HIPERLAN is presented in Fig. 5 [7]. The physical layer deals with the modulation and demodulation, channel coding, synchronization, and equalization of the radio channel. The channel access cycle (CAC) layer is related to the access of the radio link by the wireless terminals. The MAC layer contains the control functions related to topology updating and route determination. Moreover, it assigns channel access priorities to packets based on their lifetime and on the priority assigned to them by upper layers.

The multiple access technique, used by HIPERLAN terminals to acquire the radio channel, is totally distributed, based on packet priority declaration and channel sensing. It is important to examine whether this technique satisfies the criteria listed in Section IV.B regarding ATM WLAN MAC.

A. The Physical Layer of a HIPERLAN

The transmission range of the HIPERLAN transmitter is up to 50 m, enabling a channel bit rate (gross bit rate) around 23 Mbps. A specific region (5.15 to 5.30 GHz) of the frequency spectrum is allocated for HIPERLAN operation.

The allocated frequency spectrum is divided into five frequency bands (each frequency band associated with a carrier). All the nodes that belong to the same HIPERLAN should use the same unique carrier. Packets can be relayed to nodes residing out of the transmission range of source nodes, following the concept of forwarding controlled by the MAC layer of a HIPERLAN.

The physical layer is also responsible for reporting an idle channel condition to the channel access cycle layer. This happens by monitoring the received power and detecting whether it is below a well-defined threshold for a certain period (equal to 1700 high-bit-rate periods). The idle channel condition enables a HIPERLAN terminal to transmit the packet with the highest channel access priority without following the contention-based multiple-access algorithm.

B. The Channel Access Control (CAC) Layer of a HIPERLAN

The CAC layer deals with the decision of whether to transmit a packet. It is based on a multiple-access technique called elimination yield–nonpreemptive priority multiple access (EY_NPMA). Transmitted information is contained in unicast (one specific destination node is defined) and multicast (destination nodes of the packet are more than one) packets. A correct reception of a unicast packet is always followed by a transmission of an ACK (acknowledgement) packet by the destination node. This does not happen in the case of multicast packets. A channel access cycle starts after the transmission of an ACK packet or after the end of the expected transmission of an ACK packet. It is assumed that all the competing terminals that have listened to, or expected, the ACK packet, are synchronized to enter the channel access cycle. Every node contends for only one of its packets per channel access cycle, that with the highest access priority among the packets residing in the node. According to the EY_NPMA principles, the channel access cycle has four phases (Fig. 6).

The first phase is priority resolution: Generally packets can have five different access priorities (0 to 4, 0 being the highest priority). The access priority of a packet is calculated at the transmitting node, based on the remaining lifetime and the user priority of that packet, as well as on the number of intermediate nodes (hops) that the packet must traverse before being delivered at the final destination. The number of hops is known and assigned by the MAC layer, assuming that the source node has all the required information to define the routing of its packets. During the priority resolution phase, up to five priority time slots (each one being 256 bits long) could occur, each one corresponding to a level of packet access priority. During this phase, the nodes that have entered the channel access cycle schedule transmissions of a well-defined bit sequence

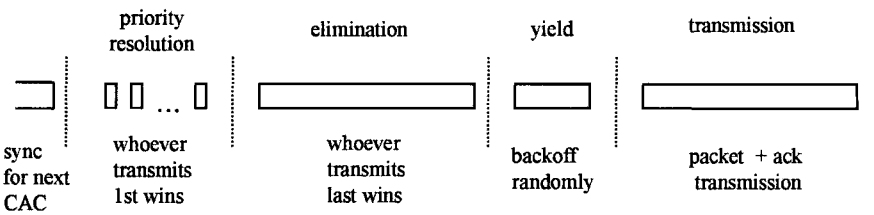


FIGURE 6 Channel access cycle in a HIPERLAN.

in the priority time slots corresponding to the access priorities of their packets, and listen to the previous priority time slots corresponding to higher access priorities. Transmissions in previous priority time slots disable the scheduled transmissions in the next time slots, reject nodes with lower access priorities from the channel access cycle, and drive EY_NPMA into its second phase. Therefore, the duration of the priority resolution phase is determined by the highest packet access priority in every channel access cycle. Moreover, more than one node could survive in this phase if they represent packets with the same access priority, and this priority is the highest in the priority resolution phase.

The second phase of the channel access cycle is called elimination: In this phase, up to 12 time slots (each one being 256 bits long) could be accessed (using a well-defined bit sequence) by the nodes that have survived the first phase of the channel access cycle. The transmission in these time slots is continuous and starts always in the first time slot. The duration of every node's transmission is based on a probability of the time slot access and is given by the binomial distribution. Every node transmits for the decided period and then listens to the channel to examine whether there are any nodes trying to access the channel, too. Longer transmissions in this phase reject listening nodes from the channel access cycle. More than one node could survive in this phase whose duration is determined by the longest transmission (up to 12×256 bits).

The third phase of the channel access cycle is called yield: All the nodes that have survived decide to sense the channel for a period that is a multiple (1 up to 14) of a high-rate 64 bit block. The first node to transmit acquires the channel because it disables all the listening nodes from the channel access cycle. However, in this phase also, more than one node can survive, and this results in packet collision.

The fourth phase is the transmission of packets by nodes that have survived in the channel access cycle. If no collision happens (only one node survived finally), a unicast packet could be transmitted followed by an ACK packet, indicating that the packet was received correctly or damaged by radio channel errors. It should be noted that not all the nodes listening to the transmitted packet might be able to listen to the related ACK packet, because the destination node could have a different coverage area than the source node. To start a new channel access cycle, nodes should be able to be synchronized based on the end of the ACK transmission or at the end of the expected ACK transmission.

C. The MAC Layer in a HIPERLAN

The MAC layer in a HIPERLAN is involved in the following procedures:

1. Network establishment, addition of a node in a network, and removal of a node from a network. These procedures are carried out by the LookUp function of HIPERLAN MAC.
2. Topology updates and packet routing determination as well as packet relay (forwarding), controlled by the routing information exchange function and the user data transfer function.
3. Power conservation by declaring periods in which the receiver of a node is active and can listen to transmitted packets. The related control functions reside in the power conservation function of the HIPERLAN MAC.

4. Calculation of the channel access priority of packets to be transmitted based on the number of hops and remaining packet lifetime. This function is carried out in the HMPDU (HIPERLAN MAC protocol data unit) transfer function.

Generally, packets submitted to the MAC for transmission are assigned with one out of two user priority level (0 for high and 1 for low). The remaining lifetime of each packet can be calculated based on its total lifetime and on the transit delay it has experienced. The node presents in the next channel access cycle the packet with the highest channel access priority among its packets.

The mechanism used in a HIPERLAN offers a way to distinguish between delay sensitive and nondelay critical applications. However, it is still to be shown if a HIPERLAN can achieve the same capacity as adaptive centralized methods when delay sensitive services are multiplexed in the network.

D. HIPERLAN Inability for Traffic Policing

In a HIPERLAN there is no call establishment phase before the information transfer takes place. So there is no knowledge about the total traffic available in the network and no call admission functions. This means that overload conditions in the network cannot be avoided because traffic policing functions are not available and quality of service cannot be satisfied even if the throughput of the multiple-access scheme is high. Even if traffic control functions are assumed in the layers above MAC, still a HIPERLAN could not support optimally the QoS of different services because the MAC layer deals with packet priorities (based on aging) but not with service priorities. For example, if video transmissions and file transfers are multiplexed in the network, although initially video packets will have higher priority over data packets, gradually the latter will acquire (by the MAC algorithm) high priorities and contend for the channel on equal terms with video packets.

Moreover, the multiple-access mechanism in a HIPERLAN cannot provide bandwidth guarantees that result in QoS preservation. This happens because the MAC layer in a HIPERLAN cannot disable sources from competing for channel resources when their traffic emission exceeds the values negotiated during call setup; i.e., there are no traffic policing functions. However, traffic policing and bandwidth preservation are necessary in ATM networks. In Fig. 7 it is shown that the multiple-access scheme of a HIPERLAN does not provide bandwidth guarantees for connections that conform to their negotiated traffic limits. VBR video connections based on [19] are assumed, at the average bit rate of 2 Mbps. Figure 7 presents the dropping probability (P_d) of conforming and nonconforming connections when the maximum capacity of the system is achieved, and for different amounts of bandwidth violation (in Mbps) by nonconforming connections. It is shown that both conforming and nonconforming connections experience similar packet loss; i.e., the network cannot protect the bandwidth of the conforming connections. This happens because the multiple-access scheme of a HIPERLAN tries to share equally the total bandwidth without taking into account any traffic contract. P_d should be stable for conforming connections whatever the amount of bandwidth violation. Thus in a HIPERLAN, there is no way to combat bandwidth violation occurring

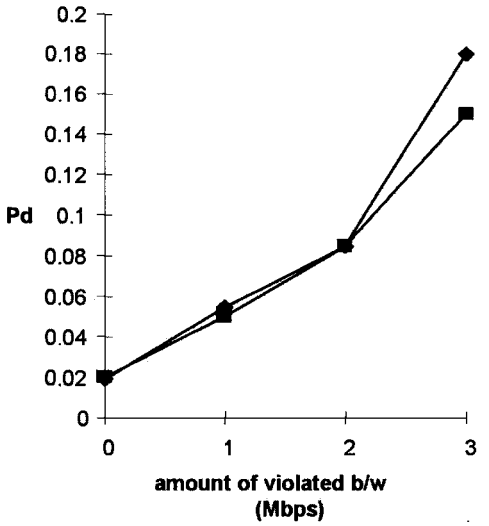


FIGURE 7 Dropping probability (P_d) for (♦) conforming and (■) nonconforming connections in a HIPERLAN versus the amount of excessive bandwidth.

as a result of the short-term variations of traffic. However, ATM networks incorporate traffic policing functions that deal with exactly this problem [12], and such mechanisms are expected to reside in an ATM WLAN too, as they have a straightforward impact on the quality of service offered. By modifying the existing HIPERLAN standard, traffic policing could be applied. To do so, it is proposed that the HIPERLAN standard be modified in order to adopt a centralized topology according to the ATM concept. Moreover, a centralized multiplexing protocol based on adaptive TDMA can be introduced and compared to the distributed HIPERLAN CAC in order to assist the design of the optimum ATM WLAN MAC.

E. Comparison of a HIPERLAN and Adaptive TDMA

1. Network Configuration for ATM over a HIPERLAN

The modified HIPERLAN system architecture consists of a fixed part and a wireless part (Fig. 8). The fixed part is a traditional ATM network, while the wireless part extends the ATM concept over the physical layer of a HIPERLAN.

The wireless part consists of wireless terminals executing the user side of the protocol for ATM over a HIPERLAN, and BSs executing the network part of the corresponding protocol. The interface between BSs and backbone ATM switches could be based on P-NNI. Within the proposed system architecture, BSs perform a superset of HIPERLAN forwarder [7] functions. In addition to packet switching they perform call admission control, bandwidth allocation, and ATM address registration. BSs communicate through the fixed network and according to the ATM concept. The terminals in the wireless part of a HIPERLAN-based ATM act like HIPERLAN nonforwarder nodes; i.e., they expect a forwarder to relay their packets toward the final destination.

Within the proposed scenario many terminals are logically attached to one BS whose coverage area defines a cell. BS communicates with all its adjacent

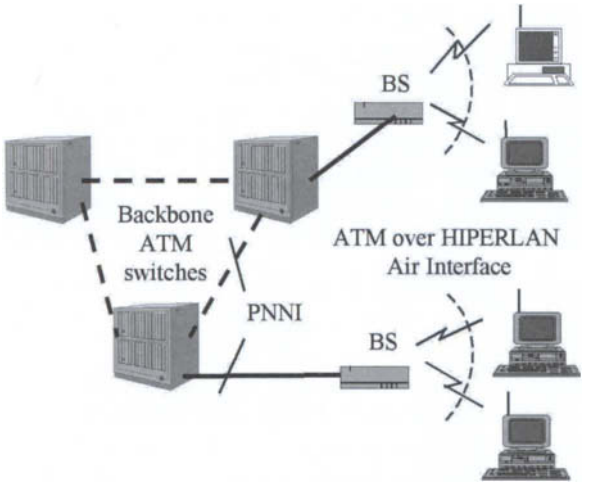


FIGURE 8 HIPERLAN-based ATM WLAN configuration.

terminals using a carrier different from those used by neighboring BSs. Because there are three main carriers defined for HIPERLAN operation, a frequency reuse factor of 3 could be used.

2. Modifications of HIPERLAN PRM for ATM WLAN Operation

To support adaptive bandwidth allocation and traffic policing in the wireless part of the HIPERLAN-based ATM WLAN, certain modifications are proposed in the CAC and MAC layers of a HIPERLAN. The objective is to define a superframe structure for the transmissions on the air interface in every cell (Fig. 9). This superframe consists of a contention-based period, where all the HIPERLAN-specified transmissions can be supported, and a contention-free

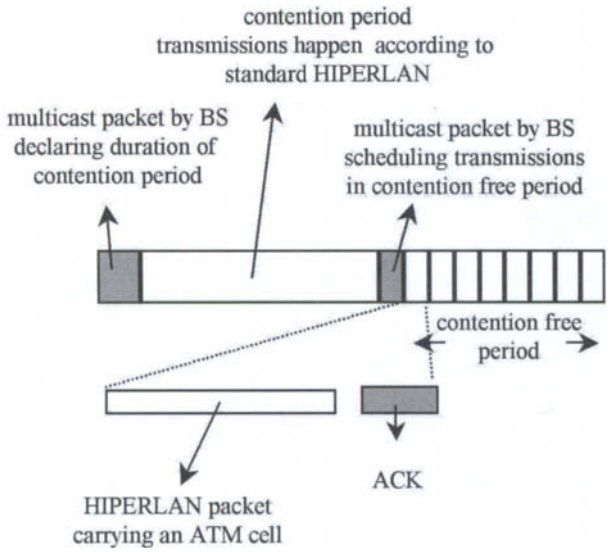


FIGURE 9 Air interface in HIPERLAN-based ATM WLAN.

period, where packets carrying ATM traffic are transmitted following the TDMA concept. The duration of the superframe is fixed while the boundary between its periods is variable, based on the terminal population and total ATM traffic within a cell. To support the second period of the superframe the HIPERLAN MAC and CAC must be slightly modified and their interface must be extended. After the applied modifications, CAC transmits a packet in one of the three following cases (where the first two are already defined in the existing HIPERLAN standard):

1. The channel has been found free, the relevant indication has been given to the MAC, and a packet has been issued at the SAP between the MAC and the CAC.
2. A new channel access cycle has been initiated, the corresponding indication has been signaled to the MAC, and a packet has been issued at the service access point between the MAC and the CAC. In this case the CAC will compete for the channel entering a new channel access cycle.
3. A new time slot has begun, this event has been signaled to the MAC to indicate the beginning of this time slot, and the MAC has delivered to the CAC a HIPERLAN packet that carries an ATM cell in its payload. All the packets transmitted this way are unicast packets and they are acknowledged according to the CAC procedures. The end of the acknowledgement transmission defines the end of the time slot.

The third case requires modifications in the CAC and MAC layers of a HIPERLAN to support the superframe structure. For that reason, the superframe delineation function is added in the CAC layer (Fig. 10). At the beginning of the contention-free period of a superframe, the BS transmits a packet schedule multicast packet to allocate the time slots to active ATM connections (Fig. 9). The reception of this multicast packet forces CAC to enter the superframe delineation function and indicate the beginning of all the time slots to the MAC.

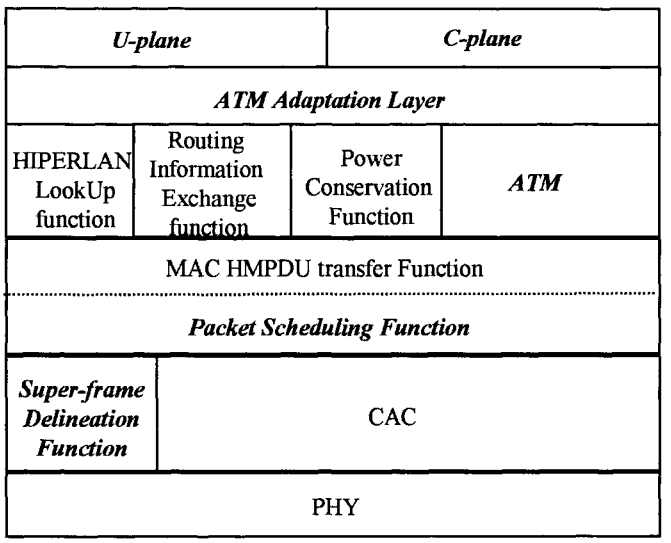


FIGURE 10 PRM for HIPERLAN-based ATM WLAN (the proposed additions) are in italics.

Moreover, the packet scheduling function (PSF) is added as an extension to the HMPDU transfer function in the MAC layer of the HIPERLAN (Fig. 10). The PSF selects a packet from the ATM connection scheduled for a time slot and delivers it to the HIPERLAN CAC for transmission. As already stated the scheduling information is contained in the packet schedule multicast packet, which is delivered to the PSF.

The modified HIPERLAN PRM for WATM operation (Fig. 10) applies at the UNI between terminals and BS. Adapting the presented PRM, full compatibility is achieved between HIPERLAN-based ATM WLAN terminals and terminals in fixed ATM.

The existence of the superframe delineation function causes a reduction of the overheads added by the HIPERLAN MAC during the contention-free period. In this period, only packets carrying ATM cells are transmitted on the air interface, so all the transmitted packets have the same size and format. Then all the parts related to variable length and format in the MAC protocol data units described in HIPERLAN standard are obsolete. The same assumption stands for the CAC layer. If the MAC, CAC, and PHY layer overheads are added to transmit ATM cells, the transmission efficiency is 18%. Contentions occurring because of the channel access scheme of a HIPERLAN will reduce further the bandwidth efficiency of the system. To increase it, in the proposed ATM over HIPERLAN, a variable length packet format is associated to the contention-based period of the superframe and a fixed packet format is used within the contention-free period. After the elimination of unnecessary overheads of the MAC and CAC, the overheads added by PHY bring the transmission efficiency to 22%, achieving a 4% improvement compared to standard HIPERLAN packet format. The unnecessary overheads consist of the fields related to the MAC source and destination addresses that are redundant because all the packets in the contention-free period are exchanged between a terminal and the BS.

Moreover, the multiple-access scheme adopted in the contention-free period is expected to improve further the bandwidth efficiency compared to the standard HIPERLAN channel access scheme.

The contention-based period is used to carry out HIPERLAN control functions, to support connections with no delay requirements or allocated bandwidth, and to transmit ATM messages related to call control and address registration.

3. Bandwidth Allocation in a HIPERLAN-Based ATM WLAN

The algorithm used for the allocation of the radio channel should be flexible enough to allow bandwidth sharing among terminals according to their instantaneous capacity requirements. Moreover, it should be able to provide bandwidth guarantees and apply traffic policing functions to established connections. The packet scheduling function introduced in the previous section deals with this problem. In the terminal side and for every established ATM connection, it keeps a separate virtual buffer, and based on the occupancy of this buffer, it reports the current bit rate of the connection to the packet scheduling function of the BS.

The calculation of the connection rate is based on the ATM cell interarrival process within a certain period. To report the latest calculated rate of a connection, the terminal sends to the BS a resource management ATM cell.

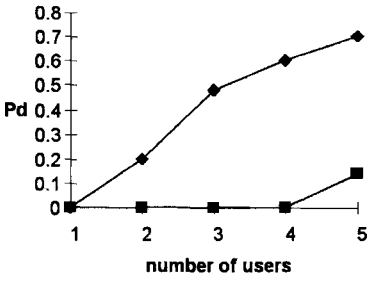


FIGURE 11 Packet loss probability: (♦) standard HIPERLAN and (■) HIPERLAN-based ATM WLAN.

The resource management ATM cell is transmitted in a time slot allocated to the corresponding virtual connection or in the contention period of the superframe. The packet scheduling function residing in the BS uses the information in the resource management ATM cells to regulate the bandwidth allocation in the contention-free period of the superframe. The algorithm is based on the negotiated traffic contracts defined during call setup according to ATM standards. These contracts are used by the traffic policing functions of the ATM layer to assign priorities to bandwidth requests. These priorities control the scheduling of packets in the contention-free period of the superframe. Following this technique the bandwidth requirements of individual sources are monitored and bandwidth violations are avoided. Thus, QoS can be guaranteed.

4. Simulation Results

For the presented simulation results, a superframe lasting 1.215 ms and consisting of a contention-free period with 15 time slots was assumed. The first time slot of the contention-free period is used by the base station to schedule terminal transmissions in the following time slots. Figures 11 and 12 compare the packet delay and packet loss probability in a standard HIPERLAN to those experienced in a HIPERLAN-based ATM WLAN. The VBR traffic sources have an average rate of 1 Mbps and are modeled on that in [19]. It is obvious that in the case of a standard HIPERLAN, the high transmission overheads combined with the contention-based multiple access result in poor system performance and user capacity. When the proposed modifications are applied to

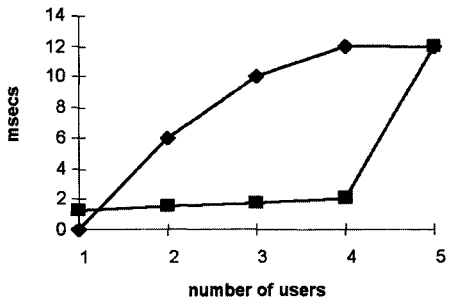


FIGURE 12 Mean access delays (max allowed delay:12 ms): (♦) standard HIPERLAN and (■) HIPERLAN-based ATM WLAN.

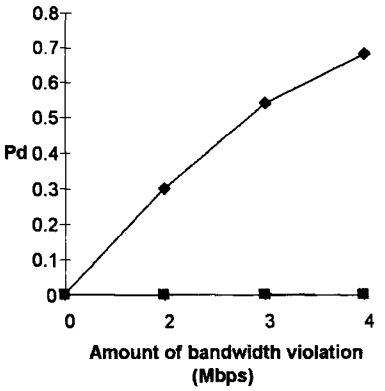


FIGURE 13 Dropping probability (P_d) for (■)conforming and (♦) nonconforming connections in HIPERLAN-based ATM WLAN, versus the amount of excessive bandwidth.

come across with the HIPERLAN-based ATM there is a significant improvement in user capacity. Assuming 2% as the maximum acceptable packet loss probability, the capacity of a standard HIPERLAN when it encapsulates ATM is only one user (1 Mbps on average) while the modified HIPERLAN based on TDMA can support 4 users (4 Mbps on average). Moreover, Fig. 13 shows that the packet loss probability for conforming sources is kept constant in the case of a HIPERLAN-based ATM WLAN when the system operates at the maximum user capacity and irrespectively of the amount of excessive traffic offered by nonconforming functions.

Thus, a centralized approach based on adaptive TDMA is preferred over the contention-based multiple-access scheme when ATM traffic is supported within a wireless system. The advantage is both quantitative in terms of user capacity and qualitative in terms of guaranteed QoS offered to conforming connections.

VI. OPTIMUM DESIGN FOR ATM WLAN

A. Cellular Architecture for a ATM WLAN

The medium access technique and the topology of the network entities are strongly related. With contention-based protocols, the decision for transmission is taken individually by every node, so the resulting topology is peer-to-peer. On the other hand when polling, or TDMA- or CDMA-based protocols are used, it is mandatory to define an entity responsible for channel assignment to terminals. This entity is the BS, and the resulting topology is star with the BS acting as the central element scheduling all the transmissions in its coverage area (cell). The BS not only schedules the transmissions but also relays the packets toward its coverage area or the fixed network where it is attached. Thus, the existence of the BS eliminates the hidden terminal phenomenon [20].

The installation of several BSs with partially overlapped coverage areas allows the definition of clusters of cells. Because BSs are fixed, their coverage

areas are deterministic or predictable and network planning or coverage area planning is easier than peer-to-peer networks. Furthermore, there are more benefits resulting from the presence of the BS in star-based topologies:

- The transmission range of a terminal corresponds to the radius of a cell, while in a peer-to-peer network the transmission range of every node should be equal to the diameter of the network. This results in lower power consumption for the star topology.
- Power consumption is also achieved because of transmission scheduling. Generally, the power of a terminal is consumed by both its transmitter and its receiver. In a packet scheduling scenario, power is not consumed by the receiver to sense the carrier. Moreover, terminals do not have to be involved in unsuccessful packet transmissions because of collision and enter the sleep mode until the instant indicated by the BS for packet transmission or reception.
- BSs belong to the fixed part of the network, so they do not have power consumption problems. It is easier then to execute more efficient multiple-access algorithms based on the global knowledge of traffic. Complicated channel access mechanisms are not recommended for portable terminals, and there is no straightforward way for knowledge of the total traffic load. This has an impact on the capability of preserving bandwidth for handoff and also on the capability for mobility support.
- Moreover, the capability of applying more sophisticated multiple-access methods in star-based networks enables them to accommodate more than one complex service with variable traffic patterns. This is a crucial issue considering the expectations from the ATM WLAN.
- The presence of the BS makes easy the deployment of control functions for security, mobility management, and network management.
- If the wireless network must be interconnected with a wired infrastructure, the presence of a BS reachable by all terminals is an advantage. Then the internetworking functions can be integrated with the traffic controlling functions of the BS and can guarantee access to the wired network for every terminal. Moreover, traffic from the access point to wireless terminals will be the main part of the whole network traffic, and the ability of the BS to tailor its multiple-access scheme to the new traffic needs is quite advantageous.

Moreover, there are certain characteristics of ATM technology that favor the selection of a centralized solution regarding the topology and multiple-access scheme of an ATM WLAN:

- Point-to-point physical connections among network entities, with physically separated forward and backward directions of transmission. The physical connectivity has a straightforward impact on the protocol architecture of ATM. The entities forming a network are divided into a group forming the user part (terminals) and a group forming the network part (switches). The signaling related to call control always takes place between an ATM terminal and an ATM switch or between ATM switches. There is no case of having signaling for call control between two ATM terminals.
- The existence of a call setup phase, between a terminal and the network, for every connection in the system (whatever the connection lifetime is). The

traffic contract for the connection and the route that will be followed by all the packets in that connection unless a fault condition occurs are defined. All the above-mentioned reasons justify the envisioned cellular system architecture for an ATM WLAN as presented in Fig. 3.

B. Traffic Policing in the MAC of an ATM WLAN

This section proposes an optimum solution for the MAC layer of an ATM WLAN based on adaptive TDMA and integrates the mechanisms required to preserve QoS and support dynamic bandwidth allocation.

The objectives of the multiple-access layer regarding the QoS of connections are:

- The transmission of submitted packets within the negotiated limits of access delay. End-to-end packet delay consists of the following components.
  1. Access delay in the calling party's access network.
  2. Queuing delay in the backbone network.
  3. Possible access delay in the called party's access network. Maximum access delays are defined per virtual connection during call setup. Generated packets having experienced access delays longer than the negotiated are dropped.
- The preservation of bandwidth assigned to connections in the access network. It is assumed that a part of the total bandwidth is assigned per connection. When VBR sources transmit at a rate higher than their allocated bandwidth, they steal bandwidth allocated to other connections. It is expected that the backbone network of an ATM WLAN contains the mechanisms for preventing such violations. Similar mechanisms are introduced in the access network, too.
- The assignment of bandwidth only to connections with pending packets or with active sources. This is another way to preserve QoS because available bandwidth allocated to a silent source can be given to support the excess bandwidth of an active connection.

The completion of the above-mentioned objectives will result in the satisfaction of the defined QoS requests.

1. Traffic Patterns of Connections

The following categories of connections are assumed in the ATM WLAN MAC based on the way that information is submitted by the corresponding sources:

- CBR connections, assumed active for the whole duration of the call. The management of these connections is rather easy. A constant portion of the total bandwidth, equal to the bit rate of the connection, is allocated for the whole duration of the call. In order to preserve the total bandwidth and enable several terminals to access the channel, the call duration could be specified.
- On–off connections, characterized by two states: an idle state when no transmission occurs and an active state when information is transmitted at a peak rate. An adaptive multiple-access scheme could take advantage of the idle periods of connections and multiplex sources of the same type with total peak bandwidth more than the bandwidth of the network. The duration of active

and idle states is given usually by exponential distributions, and this provides a mechanism for adjusting the multiplexing gain. For example, when a voice source is characterized by one idle state with mean duration of 650 ms, an active state with mean duration of 350 ms, and a bit rate of 32 Kbps, it is expected that, on average, three sources could be multiplexed on a 32-Kbps channel. The multiple-access scheme could disable active connections from using the channel when they have exceeded a negotiated period in the active state, or when they have exceeded a negotiated ratio of active period over silence period.

• Continuous VBR connections, transmitting at a range of bit rates. Although they can be assigned an average bit rate (calculated as generated bits over the call period), it is expected that rarely do they transmit at this rate. However, most of the time the current transmission rate is close to the average bit rate. A useful piece of information is the period of time for which these sources keep a constant rate. For example, every 33 ms there is a change in the bit rate of a video source, when video is transmitted at a picture frame rate of 30 frames per second and the bit rate changes per image frame. In that case a bandwidth renegotiation could take place every 33 ms. Then the bandwidth required for a longer time period can be calculated in an initial short period and packet scheduling can become easier. For example, in a TDMA scheme with frame duration equal to 6 ms, the bit rate of a video source is valid for five TDMA frames.

2. Exploiting Silence and Low-Bit-Rate Periods of Connections

When sources enter the silent period, the bandwidth assigned to them could be reallocated to connections with pending packets. Moreover, the silent sources should have a way of indicating their transition back to active in order to acquire the required bandwidth. This indication should be fast enough, to enable the fast allocation of bandwidth, although the mechanism used should preserve the total bandwidth in the network. Let's assume that voice sources, characterized by silences and talkspurts, are multiplexed in the network, and that the voice-source bit rate during talkspurt is 32 Kbps, while the affordable access delay is 20 ms. Moreover, let's assume that the transmitted packet has a payload of 384 bits (ATM cell like) and that convolutional code (2,1) is applied on the generated voice information. Then, during talkspurts, one packet is delivered at the MAC layer every 6 ms. If the packet experiences delay more than 20 ms it is dropped. When a voice source becomes active at t_0, it should take bandwidth to transmit its first packet, not before $(t_0 + 6)$ ms and not later than $(t_0 + 26)$ ms.

In PRMA [23], transmissions take place in time slots that are divided into traffic time slots (assigned to active connections by the base station) and reservation time slots that occur periodically (in well-defined time intervals). All the sources entering into their active state try to transmit the first packet of the active state in the reservation time slots following contention-based multiple-access methods like slotted ALOHA [24]. So, in PRMA it is guaranteed that if a source becomes active, it will contend for bandwidth as soon as its first packet is ready. If the transmission is successful, then the base station allocates traffic channels to the active source; otherwise, subsequent attempts for transmission in the reservation slots take place. So, in PRMA it is not guaranteed that a voice source, becoming active, will transmit the first packet within the

required time limits. This could lead to packet loss because of excessive delay or because failure to transmit the first packet of the active state could influence the rest of the packets in that state. Moreover, bandwidth is wasted because of the contention-based multiple access of the reservation slots. If slotted Aloha is used, the maximum achievable throughput in the reservation slots periods is 40% for same sources in the network or 50% for nonuniform sources [24]. Assuming that two slots are used for reservation every 6 ms, the bandwidth assigned for reservation slots is 128 Kbps (every slot carries 384 information bits). The wasted bandwidth in that case ranges from 64 to 76.8 Kbps. Moreover, even if the bandwidth loss is affordable, the access delay experienced due to the contention-based access of reservation slots could be prohibitive for multimedia services.

However, polling could be used as another mechanism for indicating the transition of a source from idle to active. A base station allocates periodically traffic slots to idle sources. The time interval between two traffic slots allocated to the same idle sources could be based on the delay requirements of the source or on the statistics describing the source transitions. Bandwidth is lost because the base station could poll inactive terminals. Assuming that one traffic slot is assigned every 20 ms to an idle source (to compensate for maximum affordable delay), the wasted bandwidth is 19.2 Kbps per idle source, i.e., three to four times less than that in the case of PRMA. Moreover, polling guarantees stability and upper bounded delays for the indication of active source states in contrast with the contention in the reservation slots of PRMA.

Furthermore, short time slots, called reservation subslots, could be used by terminals to indicate transition of the source to active or the existence of a packet ready for transmission. Based on the information carried in the reservation subslots, the base station could assign traffic slots to sources or it could keep excluding idle sources from the bandwidth allocation. Assuming that every subslot contains 8 bits of information and is given to the same idle source every 16 ms, the bandwidth assigned for reservation subslots is 500 bps per idle source. Then 128 sources could be simultaneously multiplexed to have the same wasted bandwidth as in PRMA. Different kinds of services have different delay requirements, so the polling interval could be different for different connections. This could lead to a multiframe structure on the air interface that consists of different cycles corresponding to connections with different polling requirements. A class of connections could be polled (using the subslots) every 1 ms (polling channel: 8 Kbps per connection), another class every 6 ms (1.3 Kbps), etc. Introducing reservation subslots, information transfer and bandwidth reservation are separated, and contention in the traffic slots is avoided.

The ATM WLAN will support not only temporary virtual connections that exist for the duration of the corresponding call, but also semi-permanent connections that could exist on a constant basis (i.e., the connection between an X-terminal and an X-server). When semi-permanent virtual connections are established it is expected that they will be highly asynchronous, so it will not be efficient to apply polling to them. It is better to provide a contention-based mechanism for such connections. Moreover, it is more likely to have successful transmission when many reservation subslots are available and not when there are few and longer time slots. To indicate pending bursts, a

number of reservation subslots will be accessed following slotted ALOHA. Occasionally, these subslots could be uniquely allocated to connections for polling.

3. Bandwidth Renegotiation in ATM WLAN

In order to multiplex various kinds of traffic sources in the ATM WLAN and preserve a high utilization on the air interface, it is appropriate to introduce the concept of in-call bandwidth negotiation. During call setup, a minimum and maximum bandwidth for each service need to be defined based on the QoS characteristics of that service. If the minimum bandwidth cannot be guaranteed, the call is dropped or blocked; otherwise, it is accepted and the required minimum allocated. The excess bandwidth can be allocated to the connection when there is available bandwidth. Moreover, as the traffic builds up, the bandwidth of some of the connections could be reduced again to the required minimum. For non-delay-sensitive applications, the bandwidth renegotiation could be based on the ABR traffic model. The information required at the MAC could be extracted from the ATM flow control cells defined in the ABR traffic class. The proposed MAC could support future delay-sensitive applications based on ABR traffic class too.

4. Air Interface Based on Adaptive TDMA

Figure 14 presents the air interface that supports the features presented in the previous sections. The air interface consists of a sequence of logical frames (Fig. 14). Every frame contains traffic time slots where the base station and the wireless terminals transmit ATM cell-like packets, as well as reservation subslots accessed only by the wireless terminals. Reservation subslots are used to indicate the existence of a packet burst or report the bit rate of

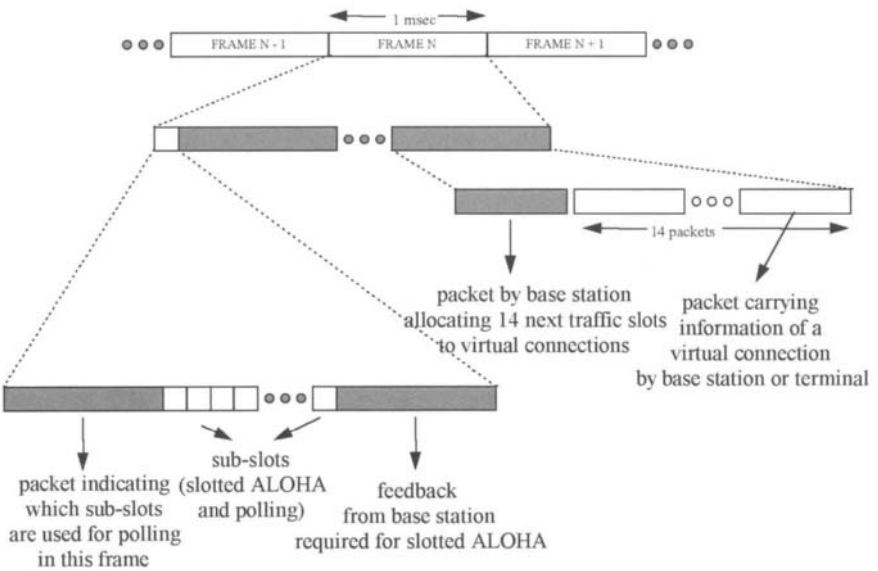


FIGURE 14 Air interface of ATM WLAN.

a source expressed in packets per frame. They are accessed following slotted ALOHA, but periodically some of them are assigned to specific connections for polling. For this reason, the base station indicates which subslots are used for polling and which can be used for contention, by broadcasting a packet with the relevant information. Moreover, the transmissions in the reservation subslots are acknowledged by the base station to indicate collisions that possibly occurred.

Each frame consists of three groups of traffic slots. Each group has 15 traffic slots, the first accessed by the BS for broadcasting a packet, indicating the connections that will use the other 14 time slots of the group. So each frame has in total 45 time slots. All the signaling messages and the user information are carried in the transmitted packets within the traffic slots. The reservation subslots carry only short messages related to the MAC layer.

5. Mechanism for QoS Assurance

During call setup, the BS decides about the traffic pattern of each new connection. Based on the delay requirements of the supported service, the BS determines the polling periods that will be used for the new connection when the corresponding source is idle (how often it will be polled if idle). Moreover, the traffic contract is negotiated between the BS and the corresponding terminal. For bursty sources the traffic contract specifies the number of packets that will be guaranteed with bandwidth within a certain time interval. This value is based on the required average bit rate declared by the terminal. For continuous VBR sources, the traffic contract specifies the guaranteed bandwidth of the connection based on the declared average bit rate of the source. During the call, the traffic contract is used by the BS to assign access priorities to the generated packets.

For highly asynchronous (on–off) connections, every new packet burst is indicated using the reservation subslots. It is assigned with an access priority based on its length, the time of occurrence, and the length of the previous burst in that connection. Moreover, different packets of the burst could be assigned with different access priorities.

For continuous VBR connections, a new bit rate is reported using a reservation subslot. The new rate is used by the BS to determine the generation instants of the new packets in the corresponding terminal. Each new packet is assigned with an access priority value based on the guaranteed bandwidth of the connection and the reported bit rate.

Four levels of priority are defined in the MAC layer of the ATM WLAN:

1. Priority zero, being the highest, corresponds to a guaranteed bandwidth of connections and is given to packets that will be definitely transmitted at the guaranteed rate.
2. Priority one is assigned to packets carrying urgent control messages (like handoff messages).
3. Priority two is given to packets that do not violate the traffic contract but still correspond to a bit rate higher than the guaranteed bandwidth of the connection. This happens when sources transmitting at a bit rate lower than the guaranteed bandwidth start transmitting at a bit rate higher than the

guaranteed bit rate. The combination of priorities zero and two will force VBR services to approach the guaranteed bit rate under a high traffic load in the network, while under light traffic conditions, it will provide with low access delays.

4. Priority three is given to packets violating traffic contracts. When the network is overloaded, priority three packets are dropped and the network is not driven into congestion. However, under low traffic load conditions, priority three packets can be transmitted.

The priority assigned to the packets corresponding to different rates could be given following two ways:

- Statically during call setup when a set of relations of the type {rate, priority} is defined. Then the source is not monitored during the call. For example, priority zero is always assigned to a rate up to 10 Mbps, priority two is assigned to an excess rate (from 10 Mbps) up to 15 Mbps, and priority three is assigned to all the excess rates above 15 Mbps.
- Dynamically based on the agreed traffic contract and on the conformance of the source to that contract. Then the base station monitors the traffic behavior of each source and relates the current state to the past history of the traffic.

The assignment of access priorities is implemented as follows: For every asynchronous source there is a timer to indicate the time instant when a number of tokens, equal to the length of a conforming burst, will be generated. The required information is retrieved from the traffic contract. Whenever a new set of tokens is generated, tokens possibly left during the previous cycle are now dropped. For a continuous VBR connection, state report instants are defined. At these instants the source reports the current bit rate. At the base station, the reported rate $\mathbf{R}$ is used to define the instants when a timer will indicate packet generation at the source. If $\mathbf{R}_G$ is the priority zero rate and $\mathbf{R}_H$ is the priority two rate then it is $\mathbf{R} = \mathbf{R}_H + \mathbf{R}_X$, where $\mathbf{R}_X$ is the priority three rate. The values $\mathbf{R}_G$ and $\mathbf{R}_H$ are used to create tokens with access priorities zero and two, according to the algorithm in Fig. 15. According to this algorithm, priority two tokens are generated based on $\mathbf{R}_H$ and some of them are characterized as priority zero tokens based on rate $\mathbf{R}_G$. A similar approach is followed when a new packet burst is generated.

Figure 16 shows how access priorities are assigned to packets of a virtual connection. As mentioned, the generation of packets is estimated by the BS based on the rate $\mathbf{R}$ and the instant of the state report.

The output of the mechanism for QoS assurance is a sorted list of active virtual connections granted with time slots on a first come first serve policy. The input of the mechanism consists of the fast bandwidth requests contained in the reservation subslots, the knowledge of the activity of the sources, the property of whether connections have guaranteed bandwidth, and the traffic contracts negotiated during call setup. Packet bursts with equal priorities are sorted based on the access delay they have experienced. This access delay is estimated by the base station based on the instance when the indication of the burst occurred.

```
if new state report of connection occurs then
  {
   1. calculate R_G and R_H values;
   2. define activation time for timer 1 based on R_G ;
   3. define activation time for timer 2 based on R_H ;
  }

if timer 1 gets active then diluting_tokens ++;

if timer 2 gets active then
  {
   if (diluting_tokens > 0 ) then
     {
      number_of_Priority0_tokens ++;
      diluting_tokens --;
     }
   else number_of_Priority2_tokens++;
  }
```

FIGURE 15 Generation of priority tokens.

The proposed mechanism exploits the traffic behavior of the connections and is able to support future applications with different traffic characteristics. For connections that consist of CBR periods, the mechanism enables the declaration of the period duration and the allocation of the requested bandwidth if this is available or guaranteed.

The total guaranteed bandwidth cannot be higher than the total bandwidth of the radio link, to cover the case where the total guaranteed bandwidth is requested simultaneously by the connections. Moreover, a portion of the total bandwidth is guaranteed to connections. The portion of network capacity left is used to absorb excessive transmission rates of the sources above the guaranteed bandwidth but within the conforming rate. It could be characterized as a guard bandwidth. The size of this bandwidth is based on the variance of the bit rate exhibited by the traffic sources.

```
if new packet is generated then
  {
   if (number_of_Priority0_tokens > 0) then
     {
      access_priority_for_new_packet = 0;
      number_of_Priority0_tokens --;
     }
   else if (number_of_Priority2_tokens > 0) then
          {
           access_priority_for_new_packet = 2;
           number_of_Priority2_tokens --;
          }
   else access_priority_for_new_packet = 3;
  }
```

FIGURE 16 Assignment of packet access priority.

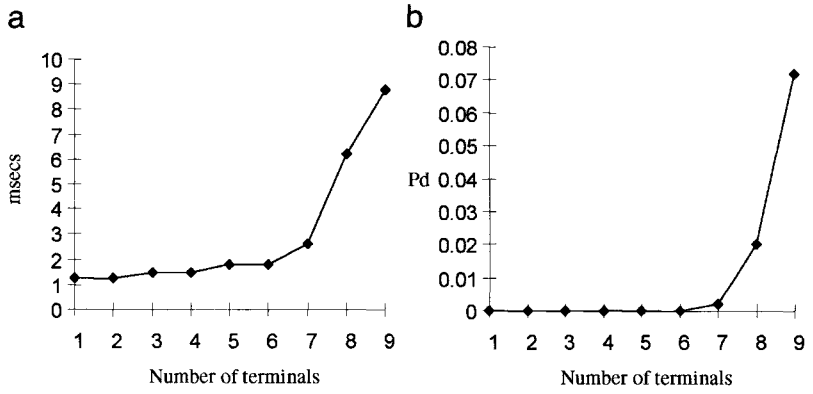


FIGURE 17 (a) Mean access delay and (b) packet dropping probability in ATM WLAN.

6. Adaptive TDMA Performance

The capability of the proposed algorithm to apply traffic policing on connections is proven by simulation of a network with variable bit rate (VBR) video sources. The following approach was followed: Initially the capacity of a coverage area (in terms of number of sources) was estimated by calculating the packet dropping probability when all the sources transmit at the same average bit rate of 2 Mbps (Fig. 17) and conform to that average. Assuming 2% as an acceptable packet loss probability, the capacity is 8 VBR sources (16 Mbps on average). Then the packet loss probability is calculated when five sources conform to the average rate (2 Mbps VBR) while the other three transmit at a higher average rate (2.5 to 4 Mbps per source) to simulate nonconforming sources (Fig. 18). From Fig. 18, it is obvious that the packet loss probability of the conforming sources remains unaffected by the presence of the nonconforming connections. Moreover, the latter experience high packet loss because the network prevents the allocation of radio capacity to them, in order to preserve the bandwidth and the QoS of the conforming connections. However, when no traffic policing applies, conforming connections experience increased packet loss, which reduces the user capacity of the system.

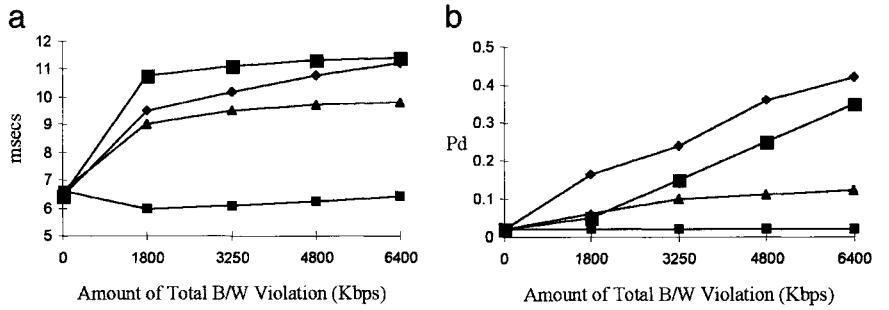


FIGURE 18 (a) Mean access delay and (b) packet dropping probability in ATM WLAN for (■, ▲)conforming (2 Mbps) and (♦, ■) nonconforming (2.5 . . . 4 Mbps) VBR sources (♦, ■) with or (▲, ■) without traffic policing.

VII. SUPPORT OF TCP OVER ATM WLAN

Before native ATM applications become widespread, ATM systems must support Internet-based applications. This happens mainly, by supporting TCP/IP over ATM adaptation layer five (AAL5) in the protocol reference model of ATM networks. Like any other ATM-oriented network, an ATM WLAN aims to support any service that can be offered through the Internet, so it must support TCP in an optimum way. However, several problems arise when TCP operates over ATM or within networks with wireless links. TCP offers compatibility between terminals residing in different networks, since it is the end-to-end transport protocol dominating in the Internet. Since TCP deals mainly with congestion problems, it does not perform efficiently when channel errors occur in wireless systems and packets are dropped. Moreover, when used over ATM, TCP performance deteriorates as cells are dropped due to buffer overflows in ATM switches, unless early packet discard (EPD) [11] is adopted. The abovementioned problems must be taken into account within the multiple-access mechanism of an ATM WLAN in order to improve the performance of TCP.

A. TCP Principles

Transport control protocol [10] provides for reliable stream transport. It is the second (after IP) most important protocol in the TCP/IP family. It provides:

- a reliable stream of bytes,
- virtual circuits for connection-oriented communication, and
- the guarantee that all bytes are delivered uncorrupted and in sequence.

TCP allows several application programs to have active connections simultaneously (multiplexing). The byte stream is buffered and packeted typically in 4-Kbyte segments for transmission in IP datagrams. Full duplex (two-way simultaneous communication) is supported, and piggybacking is used. Reliability and in-sequence delivery is obtained with the sliding window principle.

The sliding window protocol of TCP solves the flow control problem by letting the acknowledgements (ACKs) contain information that lets the sender modify the size of its sending window. The acknowledgements are cumulative: all octets up to the acknowledged octet have arrived correctly. Each ACK contains the amount of available buffer space in the receiver (receiver window size). If this size is 0, the sender stops completely. This mechanism only handles flow control, i.e., the overload problem at the receiver. Other mechanisms are needed to handle congestion control, i.e., avoiding that intermediate routers get congested.

B. Performance Issues of TCP over ATM

Several publications deal with the problems arising when TCP operates over fixed ATM networks, reporting mainly experimental results of TCP over ATM [11,27]. TCP segments are segmented into ATM cells by AAL5. When an ATM cell is lost because of buffer overflows in the ATM layer, a whole TCP segment must be retransmitted in the TCP layer. This increases the traffic load of

the ATM layer and decreases the throughput at the TCP layer simultaneously. ATM cell loss because of congestion (buffer overflows) causes TCP throughput degradation because TCP segments with lost ATM cells must be retransmitted. Moreover, the congestion avoidance mechanism of TCP is invoked and reduces TCP throughput. More bandwidth is wasted because of the transmission of the nondropped (useless) ATM cells belonging to TCP segments that will be retransmitted. The partial packet discard (PPD) [11] algorithm is proposed as a solution but is found to be nonoptimal. However, the EPD [11] algorithm provides optimal throughput. As soon as a buffer occupancy exceeds a certain threshold, whole TCP segments are discarded. However, this results in unfair sharing of bandwidth among UBR connections. Moreover, [11] investigates experimentally the performance of EPD and PPD with respect to the size of the ATM switch buffers, TCP windows, and TCP segments. In order to improve the TCP throughput one should avoid small ATM switch buffers, large TCP windows, and long TCP segments. These are issues affecting the support of TCP over an ATM WLAN.

In [28], experimental results of TCP traffic over ATM connections are discussed. To achieve the maximum throughput, no TCP segment loss must be achieved. Thus, the buffer size of the ATM switches must be equal to the sum of the receiver window sizes of all the TCP connections. If no specific algorithms are deployed, there is a problem of unfairness experienced by TCP connections over UBR traffic class. It is claimed that EPD alleviates the problem of poor TCP throughput but does not improve fairness.

C. TCP Behavior over Wireless Links

TCP deals only with congestion and does not take into account packet loss because of increased BER. Invoking congestion avoidance algorithms to deal with channel errors on the wireless link results in highly degraded throughput.

The existing schemes dealing with the above-mentioned problem are classified into three basic groups: end-to-end proposals, split-connection proposals, and link-layer proposals [29]. The end-to-end protocols attempt to make the TCP sender handle losses through the use of the two techniques. First, they use some form of selective acknowledgements (SACKs) to allow the sender to recover from multiple packet losses without invoking congestion control mechanisms. This could cause an improvement of TCP throughput on the order of 10–30%. Second, they attempt to have the sender distinguish between congestion and other forms of losses using an explicit loss notification (ELN) mechanism.

Furthermore, split-connection approaches completely hide the wireless link from the sender by terminating the TCP connection at the base station. Such schemes use a separate reliable connection between the base station and the destination host. The second connection can use techniques such as negative or selective acknowledgements, rather than just standard TCP, to perform well over the wireless link. Indirect TCP [29] follows this approach. It involves splitting each TCP connection between a sender and a receiver into two separate connections at the base station: one TCP connection between the sender and the base station, and the other between the base station and the receiver. However, the choice of TCP over the wireless link results in several performance

problems. Since TCP is not well tuned for the wireless link, the TCP sender of the wireless connection often times out, causing the original sender to stall. In addition, every packet must go through TCP protocol processing twice at the base station (as compared to zero times for a nonsplit connection approach). Another disadvantage of this approach is that the end-to-end semantics of TCP acknowledgements is violated, since acknowledgements to packets can now reach the source even before the packets actually reach the mobile host. Also, since this protocol maintains a significant amount of state at the base station per TCP connection, handoff procedures tend to be complicated and slow.

The third class of protocols, link-layer solutions, lies between the other two classes. These protocols attempt to hide link-related losses from the TCP sender by using local retransmissions and perhaps forward error correction over the wireless link. The local retransmissions use techniques that are tuned to the characteristics of the wireless link to provide a significant increase in performance. The main advantage of employing a link-layer protocol for loss recovery is that it fits naturally into the layered structure of network protocols. The link-layer protocol operates independently of higher layer protocols and does not maintain any per connection state. The main concern about link-layer protocols is the possibility of adverse effect on certain transport layer protocols such as TCP. The snoop protocol [29] is a link-layer protocol that takes advantage of the knowledge of the higher layer transport protocol (TCP). It introduces a module, called the snoop agent, at the base station. The agent monitors each packet that passes through the TCP connection in both directions and maintains a cache of TCP segments sent across the link that have not yet been acknowledged by the receiver. A packet loss is detected by the arrival of duplicate acknowledgements from the receiver or by a local timeout. The snoop agent retransmits the lost packet if it has the packet cached and suppresses the duplicate acknowledgements. Considering the above-mentioned classification of protocols, the main advantage of this approach is that it suppresses duplicate acknowledgements for TCP segments lost and retransmitted locally, thereby avoiding unnecessary congestion control invocations by the sender.

D. TCP over ATM WLAN

Figure 19 presents the network scenario and the protocol reference model used for the simulation of TCP over the ATMWLAN. Two ATM WLAN segments are defined. Terminals in one segment transmit to terminals in the other segment and vice versa. Thus all the ATM traffic must go through the two access points and the presented backbone ATM switch.

The following operations take place in the layers of the presented protocol reference model:

- A Poisson traffic generator is considered with an average interarrival time of 40 ms and a fixed packet length of 4096 bits. Thus, the average bit rate of the traffic source is 100 Kbps. Furthermore, the available bandwidth for TCP transmissions in the air interface is fixed at 1 Mbps.
- TCP layer implements the slow start algorithm and congestion avoidance algorithm, as well as fast retransmit and fast recovery [10]. When a

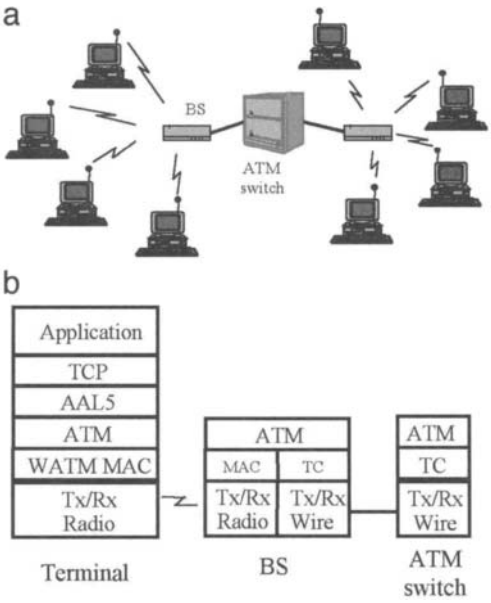


FIGURE 19 (a) Network configuration for TCP over WATM. (b) Protocol reference model of network entities.

duplicate acknowledgement is received, the TCP transmitter retransmits the implied TCP segment out of order. Moreover the timeout interval for every transmitted packet is calculated dynamically according to the algorithms residing in TCP [10]. The receiver delivers to the application layer the received TCP segments in order and stores those that are received out of order. Moreover, the implemented model supports variable maximum sliding window size, variable maximum TCP segment length, and variable packet timeout period. Acknowledgements are transmitted with highest priority and are not affected by the channel errors.

- AAL5 performs segmentation of TCP segments at the transmitting side and reassembly at the receiving side. Moreover, it checks for dropped cells within an AAL frame. At the receiving side, AAL delivers to the upper layer (TCP) AAL frames without lost cells.
- ATM performs switching of cells toward their final destination. EPD is implemented in the ATM layer of the switches, and UBR traffic class characterization is assigned to TCP connections.
- WATM MAC performs buffering of the ATM cells and traffic scheduling at the air interface of the ATM WLAN. No bandwidth guarantees apply for the TCP connections multiplexed in the air interface of ATM WLAN, since they belong to the UBR class. In this layer, ATM cell retransmissions can be selected as an option. The EPD algorithm is deployed in the WATM MAC of BSs and is expected to increase the TCP throughput as the traffic load increases in the network.
- The Tx/Rx radio layer models the channel errors, based on a user-defined cell loss ratio.

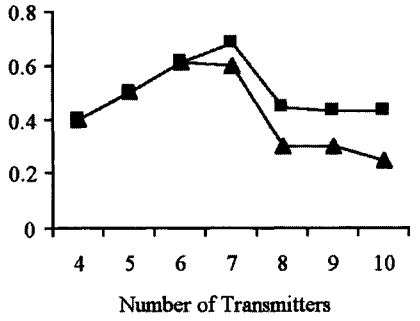


FIGURE 20 TCP layer throughput: (■) with and (▲) without EPD.

E. Performance of TCP over ATM WLAN

Two cases were considered for simulation. The first one examines the TCP behavior as the traffic load in the network increases while the second investigates the TCP performance when transmission errors are introduced (in terms of cell loss rate—CLR).

1. TCP vs Traffic Load in ATM WLAN

In the first case no CLR is assumed, and the behavior of the TCP connections is examined for different traffic loads (expressed as the number of enabled transmitters) and when EPD is active. Two main observations occur from this case. The first is that when EPD comes into effect, there is a certain improvement in TCP performance in terms of throughput. This is shown in Fig. 20 where the average TCP throughput is drawn. The second observation is that there is a limit on the traffic that can be accepted in order to achieve maximum throughput. As shown in Fig. 20, maximum throughput is achieved for seven users while for eight and up to ten users there is a significant throughput decrease. The same phenomenon has been reported in [11] where the TCP throughput decreases as the number of TCP connections increases. This shows that TCP cannot prevent congestion at the air interface and that the traffic monitoring mechanisms presented in Section VI are necessary.

2. Effect of Packet Retransmissions on TCP over an ATM WLAN

In this case the total traffic load in the air interface is kept constant (seven enabled traffic sources corresponding to the maximum achievable throughput), but the CLR is varied from 0 up to 0.1. Moreover, a cell retransmission scheme in the WATM MAC is adopted to examine the potential improvement of the system performance. Figure 21 shows the throughput degradation in the TCP level for different CLR values, when no cell retransmissions take place. Moreover, it is shown how retransmissions improve the throughput performance of TCP. For low CLR and adopting retransmissions, the throughput stays close to the maximum achievable. For the no retransmission cases the congestion avoidance algorithm is invoked, decreasing dramatically the offered traffic in the network.

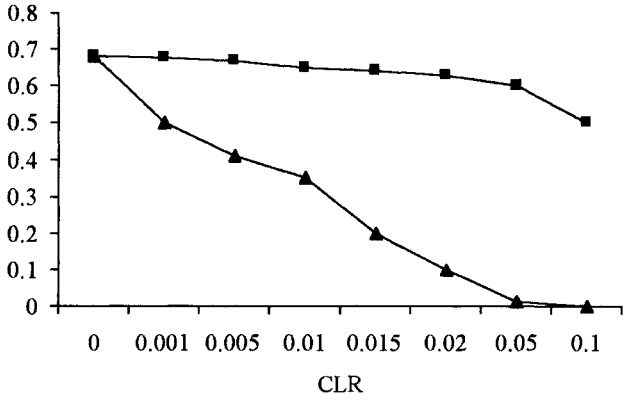


FIGURE 21 Effect of packet retransmissions on TCP throughput: (■) retransmissions and (▲) retransmissions disabled.

VIII. MOBILITY MANAGEMENT IN ATM WLAN

A. Handoff Protocol for an ATM WLAN

Figure 22 shows how an ATM WLAN will support mobile users. It is assumed that terminals communicate with BSs using an extension of the P-UNI protocol. The new protocol, called WATM-UNI, provides for the exchange of handover messages. The ATM switches are organized into peer groups (PGs) and communicate among them using an extension of the P-NNI protocol, called WATM-PNNI. This protocol is also used to connect several BSs with an ATM switch. Furthermore, each mobile station has been allocated a unique, temporary ATM address, valid only in this private LAN.

In [30], a partial reestablishment handover scheme, suitable for ATM wireless LANs, using a backward protocol, when the old radio link is available, is being discussed. A slightly modified version is presented here. Referring to Fig. 22b, when the MS receives a broadcast signal from a BS above a certain power threshold level, the MS sends to the old BS a *handover hint* (1) message (it includes the ATM address of the new BS). Since both BSs are under the same

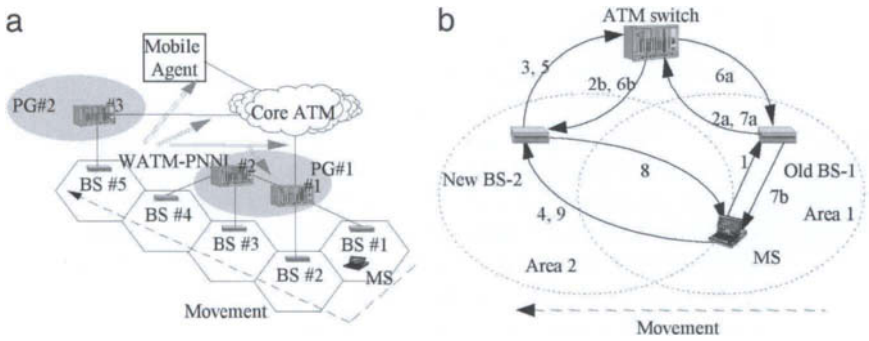


FIGURE 22 (a) Mobility management scenario in ATM WLAN. (b) Backward handover protocol.

ATM switch, a new connection between the new BS and this ATM switch is required (the path between the source and this switch does not need to be modified). This can be done from the old BS by sending a *handover call setup* (2a,2b) message to the new BS. (It is assumed that either the BS or the ATM switch keeps a record of the initial ATM setup message and can use this information for the construction of the handover call setup message, if more than one switch are involved.) The new BS responds with a *handover call connect* (3) message, and the new connection is established. When the MS approaches the new BS it sends to it a *handover request* (4) message (it includes the address of the new BS). This causes the new BS to send a *handover connection redirection* (5) message to the ATM switch, to indicate that the ATM cells should now be rerouted over the new path. The switch sends a *handover call release* (6a) message to the old BS. The old BS sends back to the switch any undelivered ATM cells and marks the end of them with a *handover call release ACK* (7a) message. This way no additional cell loss will take place. A *handover call terminate* (7b) message could be included in the protocol, so the old BS could inform the MS for the termination of their connection. When the redirection has been performed, the ATM switch sends a *handover redirection ACK* (6b) message to the new BS. The new BS sends a *handover request accept* (8) message to the MS. Finally, the MS sends to the new BS a *handover done* (9) message, indicating that it is ready to transmit and receive data.

However, if the old radio link, between the MS and the old BS, fails, message (1) cannot be sent. Upon detection of the failure, the MS sends the message (4) to the new BS. Since the old BS can be found, a *handover hint* (1) message is sent to it, from the new BS, on behalf of the MS (proxy signaling concept); a similar message flow takes place. Because the establishment of the new link takes some time, data may be lost during this handover execution.

B. Location Management in ATM WLAN

The previous algorithm also maintains the connection of the MS inside the ATM WLAN while it is moving from BS 1 to BS 5 (the message are similar to those shown in Fig. 22a. ATM switch #1, upon reception of the message (2a) finds the new ATM switch #2 (since the address of the new BS is known) and transmits the relevant message. Using the existing P-NNI protocol, the new switch can be found, a new connection from the old (#1) to the new (#2) switch can be established, and the handoff can be executed, as described previously. However, if the optimal route (either the shortest route or the route with the lowest cost that can be chosen using P-NNI routing) between the network and the MS is required, along with the avoidance of loops, the following algorithm could be used:

- *Step 1.* Upon receipt of message (2a), switch #1 checks the address of the new BS to find whether the new BS is attached to it. If true, then switch #1 establishes a new connection with the new BS (as shown in Fig. 22 b). If false, go to step 2.
- *Step 2.* The new BS is registered to a switch (#2 in the example) that belongs to the same lower level PG, as switch #1. If true go to step 3; else go to step 4.

• *Step 3.* At this stage, one of the switches in the PG is the border switch that initially receives the traffic (ATM cells) for the MS that requested the handover. If switch #1 is this border switch, it can send message (2a) to switch #2; the protocol can continue as described earlier. If switch #1 is not this border switch, it sends the message (2a) to the switch that transmits the traffic ATM cells to it. Step 2 can then be executed at the new switch.

• *Step 4.* In this case switch #1 is not able to locate in its PG the ATM switch that controls the new BS. Although the new BS could be reached, using the mechanism described in the P-NNI, it is not guaranteed that the chosen route will be the optimal one. A centralized approach has been considered for this case. It is assumed that the mobile agent (MA), shown in Fig. 22a, has knowledge of the location of the MSs in the ATM LAN (such information could be obtained from the registration process of the MSs during power-on) and the connection point that is being used by each one of the connections (this information is available to the MA if the ATM switch/gateway behind each connection informs the MA after the establishment of a connection). Then the ATM switch, after receiving the message (2a) from the old BS, advances the handover request message (2a) to the MA. The MA requests from the switch/gateway that carries the traffic of the connection (the user of the connection that requested the handover) to establish a new link between itself and the new BS.

The locality of the user movement and the distributed nature of the P-NNI protocol have been taken into consideration. The introduction of the MA (home/visitor location register) is necessary since the "mobile connections" have the additional requirement of the location management over the "static" ones and the need of a temporary ATM address. This distributed location database scheme can operate with the location management proposals in the ATM Forum [31]. To reduce the location update traffic, the MS informs the MA, when it moves from one switch to another. Further reduction could be obtained if the location update takes place when the MS moves to a new PG, since the P-NNI dynamic routing is based on the traffic load of the links among the switches. The MA is also expected to be used for the registration/deregistration and authentication of the MS with the ATM WLAN (as in the GSM). Extension of the handover protocol between two ATM WLANs could be implemented in a similar way, if the two MAs exchange the messages defined in the WATM-UNI case. Partitioning the address space into mobile and fixed addresses is not necessary, since the MA hides to the rest of the world the nature of the address of a station.

IX. CONCLUSION

ATM WLANs will be a key technology for the provision of multimedia services to mobile and portable terminals by wireless means. This chapter presented the migration path from existing fixed ATM LAN technologies to the new concept of a ATM wireless LAN. The latter inherits the architecture and control protocols of fixed ATM networks and includes the mechanisms that characterize ATM technology and offer QoS to supported services. It was

shown that a cellular approach combined with existing ATM protocols can support mobile users. The dynamic multiple-access scheme based on adaptive TDMA that was proposed is able to multiplex different kinds of applications, enable dynamic bandwidth allocation, and provide QoS to established connections. Moreover, compared to contention-based schemes it offers higher user capacity. Finally it was shown that an ATM WLAN is able to support TCP and alleviate its problem of low throughput, providing wireless access to the Internet.

REFERENCES

1. ATM Forum Technical Committee. Proposal for mobile ATM specification development. ATM Forum, Apr. 1996. Available at http://atmforum.com.
2. Special Issue on Wireless ATM, *IEEE Personal Commun.* 1(4): Aug. 1996.
3. Special Issue on Wireless ATM, *IEEE Commun. Magazine* 35(11): Nov. 1997.
4. Kruys, J. Standardisation of broadband radio access networks in ETSI's BRAN Project. In *proceedings of the EU Wireless Broadband Communications Workshop,* Brussels, Belgium, Sept. 1997.
5. DE/RES-3001-1. Radio Equipment and Systems (RES); Digital European Cordless Telecommunications (DECT) Common Interface; Part 1: Overview, ETSI, ETSI TC-RES, Jan. 1995.
6. IEEE, Draft Standard IEEE 802.11, Wireless LAN, P802.11/D1, Dec. 1994.
7. ETSI PT41 and RES 10. Radio Equipment and Systems (RES); High Performance Radio Local Area Network (HIPERLAN); Functional Specification, ETSI, Jan. 1995.
8. Buitenwerf, E. G., Colombo, H. Mitts, and Wright, P. UMTS: Fixed network issues and design options. *IEEE Personal Commun.* 2(1): 28–37, 1995.
9. Telecommunication Standardization Sector of ITU. Series Q: Switching and Signaling, Intelligent Network, Framework for IMT-2000 Networks, Q.1701, Mar. 1999.
10. Stevens, W. R. *TCP/IP Illustrated, Vol. 1, The Protocols.* Addison-Wesley, Reading, MA, 1994.
11. Romanow, A., and Floyd, S. Dynamics of TCP traffic over ATM networks. *IEEE J. Selected Areas Commun.* 13(4): 633–641, 1995.
12. ATM Forum Technical Committee. Traffic management specification, ver. 4.0. ATM Forum, Apr. 1996. Available at http://www.atmforum.com.
13. Onvural, R. *Asynchronous Transfer Mode Networks: Performance Issues.* Artech House, Norwood, MA, 1994.
14. ATM Forum Technical Committee. UNI 3.1 Specifications. ATM Forum, Sept. 1994. Available at http://www.atmforum.com.
15. Perros, H., and Elsayed, K. Call admission control schemes: A review. *IEEE Commun. Mag.* 82–91, Nov. 1996.
16. ATM Forum Technical Committee. ATM user–network interface signaling specification, ver 4.0. ATM Forum, Mar. 1995. Available at http://www.atmforum.com.
17. Washburn, K., and Evans, J. T. *TCP/IP, Running a Successful Network,* pp. 411–429. Addison–Wesley, Reading, MA, 1994.
18. DE/RES-3001-3. Radio Equipment and Systems (RES); Digital European Cordless Telecommunications (DECT) Common Interface; Part 3: Medium Access Control Layer, ETSI TC-RES, Jan. 1995.
19. Maglaris, B. Performance models of statistical multiplexing in packet video communications. *IEEE Trans. Commun.* 36: 834–844, 1988.
20. Bantz, D. F. Wireless LAN design alternatives. *IEEE Network Mag.* Mar./Apr. 1994.
21. Monogioudis, P. *Spectral Efficiency of CDMA for Personal Communication Networks.* Ph.D. Thesis, University of Surrey, 1994.
22. Pahlavan, K. Wireless intraoffice networks. *ACM Trans. Office Inform. Systems* 6(3): 277–302, 1988.
23. Goodman, D. J. Factors affecting the bandwidth efficiency of packet reservation multiple access. In *Proc. 39th IEEE Vehicular Technology Conference,* 1989.

24. Tanenbaum, A. *Computer Networks*. Prentice–Hall, Englewood Cliffs, NJ, 1988.
25. DTR/SMG-023006U. Universal Mobile Telecommunications System (UMTS); UMTS Terrestrial Radio Access (UTRA); Concept evaluation, ETSI SMG2, Dec. 1997.
26. Proakis, J. G. *Digital Communications*, 3rd ed. McGraw–Hill, Singapore, 1995.
27. Lakshman, T. V. and Madhow, U. The performance of TCP/IP Networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. Networking* 5(3): 336–350, 1997.
28. Goyal, R., Jain, R., Kalyanaraman, S. *et al.* UBR+: Improving performance of TCP over ATM-UBR service. Available http://www.cis.ohio-state.edu/~jain/papers/icc97.ps
29. Balakrishnan, H., Padmanabhan, V., Seshan, S., and Katz, R. A comparison of mechanisms for improving TCP performance over wireless Links. Available at http://daedalus.cs.berkeley.edu.
30. C-Toh, The design and implementation of a hybrid handoff protocol for multimedia wireless LANs. In *First International Conference on Mobile Computing and Networking*, Berkeley, CA, 1995.
31. ATM Forum Technical Committee. BTD-WATM-01.11, Draft Wireless ATM Capability Set 1 Specification, Sept. 1999. Available at http://www.atmforum.com.

25

SUPPORTING HIGH-SPEED APPLICATIONS ON SingAREN[1] ATM NETWORK

NGOH LEK-HENG

SingAREN, Kent Ridge Digital Labs, Singapore 119613

LI HONG-YI

Advanced Wireless Networks, Nortel Research, Nepean, Ontario, Canada K2G 6J8

[1]The Singapore Advanced Research and Education Network.

Database and Data Communication Network Systems, Vol. 3

This chapter focuses on the design of a multicast connection service for asynchronous transfer mode (ATM) networks. In particular, the solution proposed offers what is termed the "premium backbone service" to the backbone connecting nodes of the *Sing*apore *A*dvanced *R*esearch and *E*ducation *N*etwork (*SingAREN*)—a high-speed advanced broadband network serving the R&D community in Singapore and internationally. Main motivation of this work began from the need to provide scalable quality-of-service guarantees on SingAREN backbone to support high-speed applications. Existing ATM signaling solutions proposed by the ATM Forum and others are not suitable in the SingAREN heterogeneous environment with a wide variety of communication requirements. Furthermore, current solutions must be modified extensively in order to support host mobility brought about mainly by wireless access. The work described here addresses these shortcomings by proposing an integrated signaling and connection service architecture, and related algorithms for the setting up of ATM virtual channels across interconnected ATM switches. Some of the important concepts elaborated in the design include the notion of *open signaling,* the use of logical *multicast groups* to handle all connections, *traffic aggregation,* and *seamless support* for host mobility. The proposed design is described here using some details of a prototype implementation currently underway. A distributed routing scheme for ensuring that the proposed design is scalable to large worldwide ATM networks, thus making it suitable for future international trials with SingAREN's international counterparts, is presented. Finally, performance analysis and measurements are applied to routing and connection setup in order to understand their limitations.

I. BACKGROUND

The Singapore Advanced Research and Education Network (SingAREN) [32] is a national initiative to create a very high-speed platform to support R&D and advanced networking technology development, serving users from academia, research organisations, and industry. One of the key objectives for SingAREN is to provide technological support for Singapore ONE—the nation's commercial ATM broadband infrastructure [33]. To help achieve this, SingAREN operates an advanced ATM-based research backbone network linking together the networks in local universities, research organizations, and industry R&D partners. In addition, SingAREN has its international links currently connecting similar networks in the United States, Canada, and Japan. The entire network is used

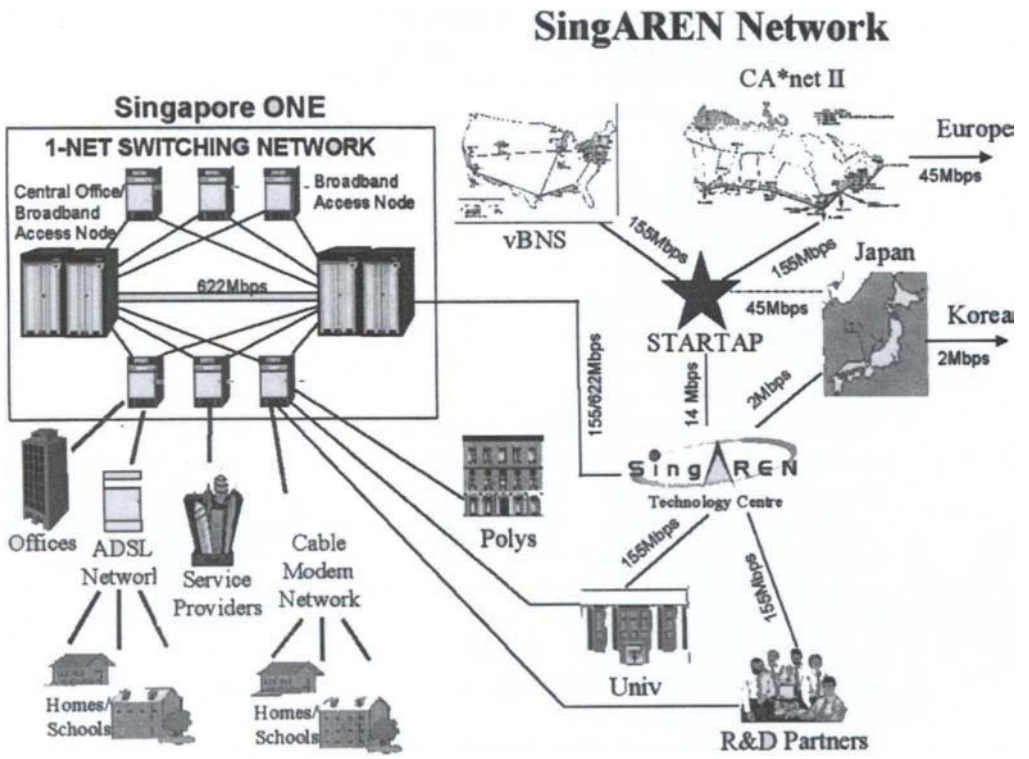


FIGURE 1 SingAREN network infrastructure showing its connections to Singapore ONE and international links.

by the connecting partner sites to conduct a wide range of activities from testing new network software and protocols to international collaborative IP-based application developments such as telemedicine. Figure 1 depicts the SingAREN network infrastructure and its relationship with Singapore ONE.

Similar to many other ATM-based R&D networks elsewhere, the SingAREN backbone network provides a basic IP connectivity to its connecting sites through a statically configured, fully meshed logical network using ATM permanent virtual circuits (PVCs). All routing takes place at the various IP routers, each belonging to individual organizations and that of SingAREN. Standard routing algorithms such as the broader gateway protocol (BGP) are employed. Under this configuration, all IP packets (unicast and multicast) between sites are carried over one or more PVCs, which act as "fat pipes." Under this arrangement, all IP packets to the same destination are carried indiscriminately over the same PVC; therefore, no link-level QoS guarantees can be provided, and multicast capabilities of the backbone switches are also not easily exploited.

II. ADVANCED APPLICATIONS ON SingAREN

Besides networking technologies, another of SingAREN's key objectives is to spur and facilitate the deployment of advanced broadband applications between

Singapore and other countries. SingAREN's international connections permit collaborative research to be carried out between researchers in Singapore and those in countries connected to SingAREN. The motivation for this is to leverage on one another's expertise, and share resources (e.g., data and equipment) and funding, thus helping to raise the level and recognition of R&D activities in Singapore in the international arena. SingAREN also plays a "match-making" role in identifying potential research partners overseas in various domain areas of research and putting researchers in Singapore in contact with these research groups overseas.

There are two main types of collaborative research projects. The first is not concerned with the creation of new applications that run over high-speed network connections, but mainly with using the network as an advanced communications infrastructure. Examples of these are video conferencing, computer-supported cooperative workspaces (CSCW) including shared applications and whiteboards, general 3-D immersive environments, and remote databases. Almost any field of research can benefit from this type of collaborative project, such as the Humanities, Management, and Law, in addition to the scientific, medical, and engineering disciplines. The second type of collaborative research project involves doing something new with networking technology, such as new ways of performing research, visualizing data, or controlling equipment over long distances. This type of collaborative research is especially relevant to the scientific, medical, and engineering disciplines, and will be dealt with in detail in the next subsection.

Although the project teams from each side are expected to secure the required funding and manage these projects themselves, SingAREN seeks to support the networking needs of these projects by providing assistance both financially and technically. Financial assistance can come in the form of subsidies in the cost of subscribing and using SingAREN links. Technical assistance can come in the form of SingAREN engineers working closely with project teams to advise on how to incorporate networking components into their overall project plan and scope, and how to exploit the available high network bandwidths to maximize their scope.

A. Advanced Applications over High-Speed Networks

Advanced applications are characterized by the following networking requirements: (1) high bandwidth, on the order of megabytes per second (Mbps) and (2) quality of service (QOS) in terms of low or bounded latency, delay variation, and data loss. These requirements are satisfied by the SingAREN local and international ATM connections at the ATM level, although bandwidth and QoS at the TCP/IP level require tuning of key network parameters as well as new functionalities in network equipment, and are current networking research areas.

There are several domain areas that can benefit from advanced applications running over high-speed networks, such as bio-informatics, tele-medicine, tele-surgery, remote sensing, weather forecasting, digital libraries and multimedia repositories, distance learning, engineering design and distributed control, tele-manufacturing, tele-architecture, data mining, sciences (physics, chemistry,

biology), and defence. Applications and content constitute the value to users over the physical network infrastructure, which can be considered as the "plumbing."

These advanced applications can also be grouped into several classes based on their characteristics:

- *Multimedia.* These refer to applications such as audio and video conferencing, distribution of multimedia content, and reliable and efficient multicasting of such content.
- *Interactive and collaborative environments.* Although shared workspaces such as CSCW can enable general collaboration as mentioned above, more advanced and domain-specific applications such as the cave automatic virtual environment (CAVE) and distributed interactive simulation (DIS) environment need to be developed to enable new ways of visualizing data and interacting with complex scenarios.
- *Time-sensitive applications.* These refer to applications having a time constraint or delay bound for transmitting and receiving data, such as interactive applications, real-time data delivery, distributed computing, remote sensing and monitoring, and distributed control, e.g., of a scanning electron microscope. For such applications, data that arrive after the specified deadline have significantly reduced value.
- *Sharing of scarce resources.* These refer to new ways of connecting users to resources geographically distant from them, such as databases and specialized equipment, e.g., in manufacturing and supercomputing. As these applications typically do not have strict timing requirements, the key networking requirement is high bandwidth.
- *Networking research and middleware.* These refer to new networking protocols, software, and equipment developed to improve upon the existing software and hardware network infrastructures, such as ATM over long distances, QoS, RSVP, IPv6, multicast, and reliable bulk transfer over high-speed links.

Table 1 shows some examples of international collaborative projects currently supported by SingAREN.

III. ADVANCED BACKBONE NETWORK SERVICES

Within the context of SingAREN, provisions for two broad classes of advanced services have been identified to be crucial besides its basic IP service and high raw bandwidth: multicast and support for future wireless broadband access. Multicast, which advocates the notion of group communications, has been well tested on computer networks and proven to be a highly effective networking service. Recent research work on the Internet has shown that multicast is both a fundamental and important service in supporting real-time applications. This point is especially illustrated by the popularity of IP multicast backbone (MBONE) and its related audio and video tools [1]. Albeit at somewhat slow speed due mainly to the bandwidth limitation of the physical links, MBONE has demonstrated that a virtual worldwide audience can be readily formed through

TABLE 1 SingAREN Projects and Collaborating Parties

Project	Collaborating parties (in Singapore and overseas counterpart order)
Tele-immersion virtual reality	Institute of High Performance Computing and University of Illinois (USA)
X-ray synchrotron technology	Bioinformatics Centre and Stanford University (USA); Bioinformatics Centre and Brookhaven National Labs (USA)
XtalBench technology	Kent Ridge Digital Labs and Scripps Research Institute (USA)
Kleisli and Ptool research	Kent Ridge Digital Labs, University of Pennsylvania and University of Illinois (USA)
Tele-education	School of Computing (NUS) and Boston Children's Hospital, Boston (USA); National University of Singapore & University of Illinois, Chicago
Tele-manufacturing/Tele-design	School of Architecture (NUS), and Massachusetts Institute of Technology, School of Architecture & Planning (USA); School of Architecture (NUS), Temasek Polytechnic, School of Engineering, and Carnegie Mellon University, Architecture & Building Physics (USA); Department of Mechanical & Production Engineering (NUS) and Integrated Manufacturing Technology Institute (Canada)
Tele-medicine	Centre for Information Enhanced Medicine (NUS) and Johns Hopkins University (USA); Medical Informatics Programme (NUS) and National Cancer Centre (Japan)
Digital library technology	Kent Ridge Digital Labs and Virginia Polytechnic Institute and State University (USA)
Web caching research	Centre for Internet Research (NUS) and San Diego Supercomputing Centre (USA)
Data integration and data mining	Kent Ridge Digital Labs and University of Pennsylvania and University if Illinois (USA)
IPv6 testbed	SingAREN, Centre for Internet Research (NUS) and Communications Research Laboratory (Japan)
DS3 network monitoring system	SingAREN, Department of Electrical Engineering (NUS), Centre for Internet Research (NUS) and University of Waikato (New Zealand)
Traffic monitoring in international links	SingAREN and University of British Columbia (Canada)
Tele-seminar over ATM	School of Computing (NUS) and Hanyang University (Korea)
IP multicast over ATM	Network Research Technology Centre (NTRC, NUS) and Asia Pacific Research Laboratory
Satellite technology	School of Electrical & Electronic Engineering (NTU), Communications Research Laboratory (Japan), NASDA (Japan) and Rutherford Laboratory (UK)
Quality of service (QoS) testing	School of Electrical & Electronic Engineering (NTU), and Hiroshima City University (Japan)
Turbo codec for software radio receiver	School of Electrical & Electronic Engineering (NTU), and Kyoto University (Japan)
Application of high-order statistics systems to confirmation radio therapy treatment	School of Electrical & Electronic Engineering (NTU), and San Diego State University

the use of IP multicast and appropriate audio/video tools. Works are currently underway to explore the use of multicast in other areas such as distributed real-time simulation. IP multicast itself is also undergoing further research, which promises to add real-time QOS guarantees and improved scalability [2,3,34].

ATM networks, on the other hand, developed independently of the Internet activities in the late 1980s. Realizing the importance of supporting group communications, all switches support one-to-many (i.e., one sender and many receivers) virtual channels (VCs) through special hardware design at the switch. Unlike the IP multicast case, however, there are severe limitations, which include one-to-many VCs are unidirectional and allow only semi-static setup through the standard signaling procedures [15]. Furthermore, all receiver VCs are expected to have the same QOS throughout the connection lifetime. Support for many-to-many multicast is accomplished by a mesh of one-to-many VCs or a multicast "reflector" approach [4,30], which makes relatively inefficient use of network resources. Another major problem is that support for interswitch ATM multicast routing is still not fully implemented, even though the signaling support for setting up one-to-many connections across multiple switches has been proposed [5,16]. Recently, various techniques that come under the general heading of "IP switching" or "multiprotocol label switching (MPLS)" have also been proposed by different vendors. The key here is to provide a more efficient support for providing IP services (including multicast) over ATM. While these various techniques have proven to be more efficient and are gaining market acceptance, however, they do not interwork well with each other, thus making them unsuitable in a multivendor environment like the SingAREN backbone.

Perhaps an even more important reason for SingAREN to have a new signaling and control solution is the need to support wireless networks in the form of a wireless access network or wireless LANs connecting to the backbone. Broadband wireless communications are becoming more and more important due to the rapid advancements of related hardware and software that resulted in great improvements in the multimedia capabilities of laptop PCs and even cellular phones. Emerging technologies such as wireless ATM [17], wide-band code division multiplex access (WCDMA) [31], and local multipoint distribution systems (LMDS) [31] will no doubt pose new challenges to the wired broadband backbone networks such as SingAREN in the area of interworking and service interoperability. Existing ATM signaling and control solutions described above will have to be extensively modified in order to support issues such as host mobility.

To overcome these problems in supporting multicast in SingAREN's ATM networks, a new scheme in achieving an integrated ATM multicast service called the *premium service* is described here. The proposed solution seeks not only to address the current IP service limitations, but also includes support for mobile hosts. To date, most of the components described in this chapter have been implemented in the SingAREN network consisting of multivendor switches. Further proofs of design scalability in larger networks is provided through an analytical approach. The rest of this chapter is organized as follows. Section IV outlines the design goals of the proposed solution suitable for SingAREN; this is

followed in Section V by a presentation of the key research ideas that shape this work. The details of the proposed multicast service are described in Sections VI and VII. In Section VIII, the issue of sclability of the proposed routing solution is addressed together with an implementation of the signaling network. The solution to how host mobility is supported is described in Section IX. Finally in Section X some conclusions and future directions of this work are outlined.

IV. SingAREN "PREMIUM" NETWORK SERVICE

The basic "fat-pipe" IP service described above is generally adequate for non-time-critical applications such as Web browsing and file transfers amongst SingAREN sites. However, a separate "premium" network transport service that provides QOS-guaranteed with multicasting capability is certainly needed for real-time applications such as IP telephony and high-quality video trials. Another requirement for this premium service is to support a wide range of other incompatible networking solutions which, more often-than-not, are adopted by the individual connecting sites for one reason or another (see Fig. 2). Furthermore, as wireless access networks and host mobility becomes increasingly a reality, the proposed premium service should also adequately address this issue. The work described here will therefore be adopted and proposed as the key technology to realize this premium service in SingAREN.

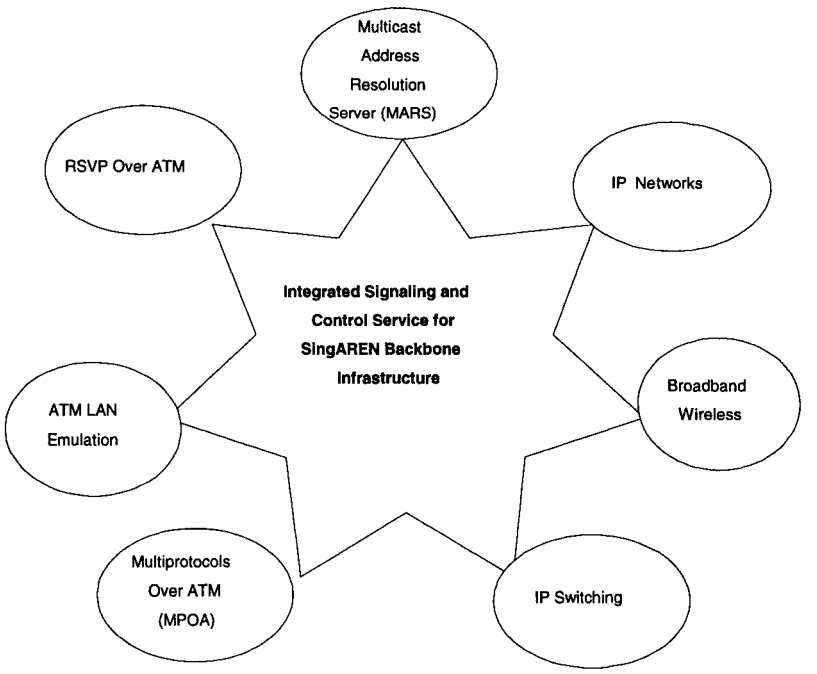


FIGURE 2 Proposed premium service interworks with other connecting SingAREN sites.

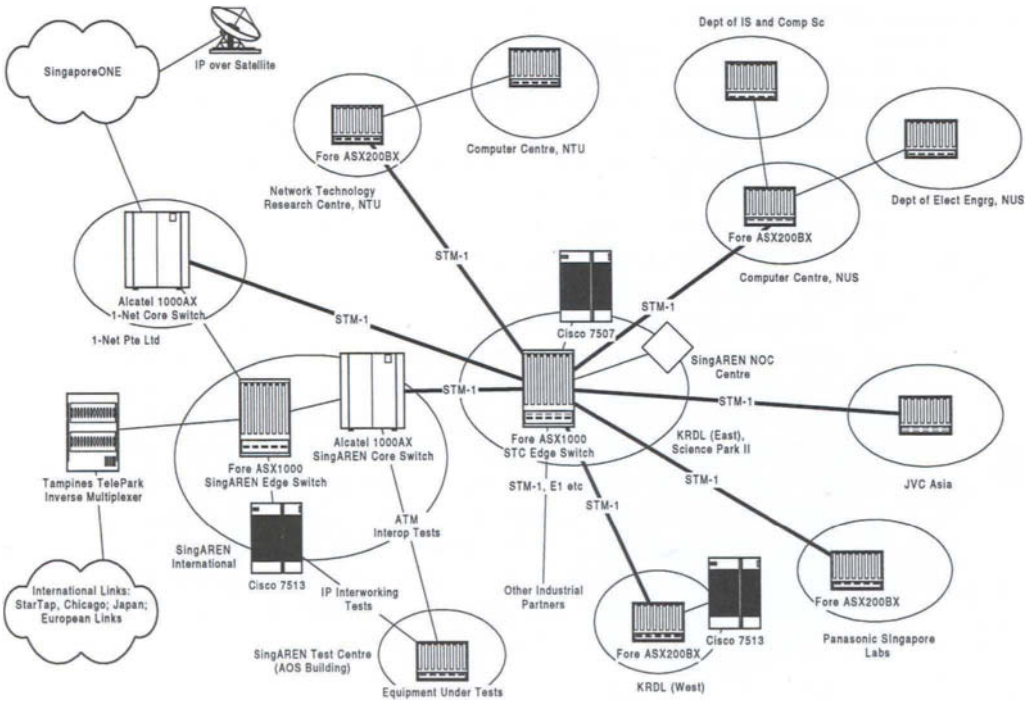


FIGURE 3 SingAREN local backbone ATM network (thick lines) and connecting sites.

A. Design Goals

The long-term objective of the work described here is to become the preferred solution over the standard ATM signaling in providing connection service for hosts and networks attached to SingAREN's ATM networks (see Fig. 3). The various design goals are elaborated next.

B. Desired Multicast Properties

Given that the focus here is to address the issue of multicast at the ATM VC level, it is appropriate to compare it with the Internet IP multicast where the concept of group communications is applied at network host level. The works and operational experience of IP multicast have demonstrated many desirable properties capable of supporting a wide range of applications both now and in the future. Some of these properties to be realized in the proposed ATM multicast service are as follows.

- Many-to-many group communications with heterogeneous QOS guarantees at both sender and receiver ends;
- Dynamic membership with leaf-initiated (senders and receivers) group operations and QOS requests;
- Multicast data filtering and merging; and
- Support simplex VC connections.

C. Support for Host Mobility

Recent breakthroughs in wireless communication technologies have also seen the rapid market penetration of cellular phones and laptop PCs over the past few years. Therefore, wireless extension of the SingAREN network is becoming an attractive option. This would require ATM networks to support mobility of wireless hosts that use the radio link as the communication path as well as wired hosts that can roam freely within the network. A mobility-enabled ATM network should support different scales of mobility such as *roaming* and *handover* that correspond to large- or small-scale user movement, respectively. Support roaming means a user is allowed to travel to another network domain and attach his host to the network at the foreign location, while support handover means that the connections should be maintained when the user moves from one "cell" coverage area to another. The mobility-enabled network protocols are needed such that the ATM networks support mobile hosts to move within the network coverage area while simultaneously maintaining the existing connections and enabling them to process incoming and outgoing calls. It should therefore be made clear that mobility does not always have to involve a wireless (i.e., radio) setting. However, it is realized that by addressing mobility in a wireless environment, a simpler but similar solution should be easily applicable for mobility in a wired environment.

D. Application-Oriented Traffic Aggregation

While the proposed solution could potentially be used to provide premium IP service on a "one-VC-per-flow" basis on the SingAREN backbone, this is clearly not scalable nor always necessary. This is because not all applications would require the one-VC-per-flow support, while some do. An example of this is an application that may only require receiving its real-time data over a dedicated VC, and its other data over a shared VC. Multiplexing of multiple real-time flows into one VC such as that of the multiparty video-conferencing scenario is also desirable. Mechanisms should therefore be provided to allow the application to indicate the mapping between IP flows and respective VCs.

E. Coexist with Other Signaling Solutions

In order to maximize the potential of the proposed solution, the multicast service should be able to be added "unintrusively" into the switches and operate side-by-side with the other ATM signaling solutions supported by default. This is so that while the premium service will be made available SingAREN-wide, other solutions such as the classical IP service should still be available on the SingAREN network to simultaneously support the other research activities that may find these existing signaling solutions adequate.

F. Scalable Design

Last but not least, since ATM is a highly scalable network technology with the ability to interwork with other network technologies, the proposed solutions

should be able to scale well in terms of connection setup performance across various network sizes ranging from a few to even thousands of switches.

V. KEY RESEARCH CONTRIBUTIONS

Before presenting the overall design, the work described here is essentially driven by a few key research ideas, which are elaborated in this section.

A. Open Signaling

Central to the work reported here is the *open signaling* approach (herewith referred to as "opensig") first proposed by the Center for Telecommunication Research (CTR), Columbia University [6–8]. The basic concepts of *opensig* can be summarized as the "design to provide an open programmable environment that facilitates the easy creation of flexible services." This idea comes about because it is envisioned that future multimedia networking will depend greatly on the level of sophistication, degree of flexibility, and speed of creating and deployment of new network multimedia services. Opensig therefore advocates the use of software architecture to achieve these goals. Central to the design of opensig has three components. They can be summarized as follows:

1. Physical network software and hardware elements are modeled using the distributed object-oriented software engineering approach. The main goal to this is to promote openness and programmability of these network components. Applying this principle to the ATM environment, the net result is that multivendor switches now support a common interface.
2. Define and make publicly available the programming interface to each of these components.
3. Define suitable software components and algorithms that result in the corresponding service-oriented *signaling networks* that will interact (program) these physical network components through the defined interface. In this way, new transport services can be created.

While the originators of opensig have focused mainly on the issues of partitioning network resources [9], this idea is extended here in network connection services. The proposed multicast service architecture is therefore modeled as a product distribution process that involves the interactions among *producers, consumers, agents,* and *switches*. The model can be expressed as a quadruple {**P, C, A, S**} where **P** is the set of producers that is the source of the information conveyed in the multicast sessions, such as a video distribution server that sends a video program to a group of receivers. **C** is the set of consumers that are the sinks of the information conveyed from the producers, such as a video-on-demand client that plays the received video programs. **A** is the set of agents responsible for negotiation and implementing a multicast session among the producers and consumers. **S** is the set of switches that transport and relay the data streams from producers to consumers.

Figure 4 shows the proposed ATM multicast service architecture. Central to the design is the *multicast service agent* (MSA), which in turn consists of

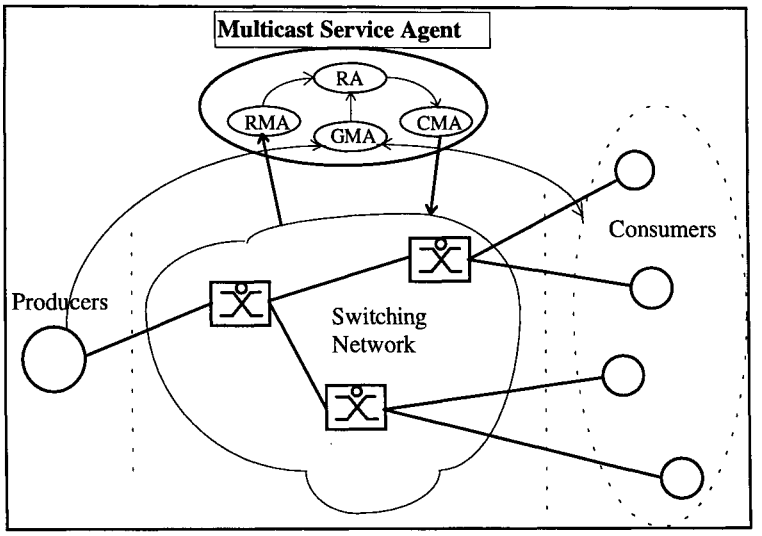


FIGURE 4 Conceptual design of the proposed ATM multicast service.

four subagents called resource management agent (RMA), group management agent (GMA), routing agent (RA), and connection management agent (CMA) respectively. The details of MSA and its respective subagents are described in Section VII.

B. Multicast as a Basis for All Connections

Another significant idea demonstrated by the work described here is to advocate the support multicasting and use of *multicast connections* as the basis for all ATM connections, including those unicast (one-to-one) connections treated as two-member groups. The consistent treatment of connections will have the immediate impact of simplifying the semantics of connections and service flexibility. Furthermore, by treating all connections as potential multicast connections, the ATM network for which this design is implemented becomes a highly "programmable" network environment in that all nodes can be dynamically grouped and connected together at any time. It is true that some computing overheads can result due to this proposed uniform treatment on all connections (e.g., multicast routing is more complex than that of the unicast case); however it can be seen later on supporting mobility. This approach allows a far more elegant approach in providing mobile access. Therefore, the gain far outweighs the potential overheads.

C. Dynamic Logical Multicast Grouping for the Support of Host Mobility

Unlike conventional unicast or the one-to-one connection concept, which often needs the parties involved to specify the exact physical location of one another for a connection to be made, multicast groups are a logical concept with dynamic memberships. What this means is that that member mobility can be supported naturally through the availability of multicast group management

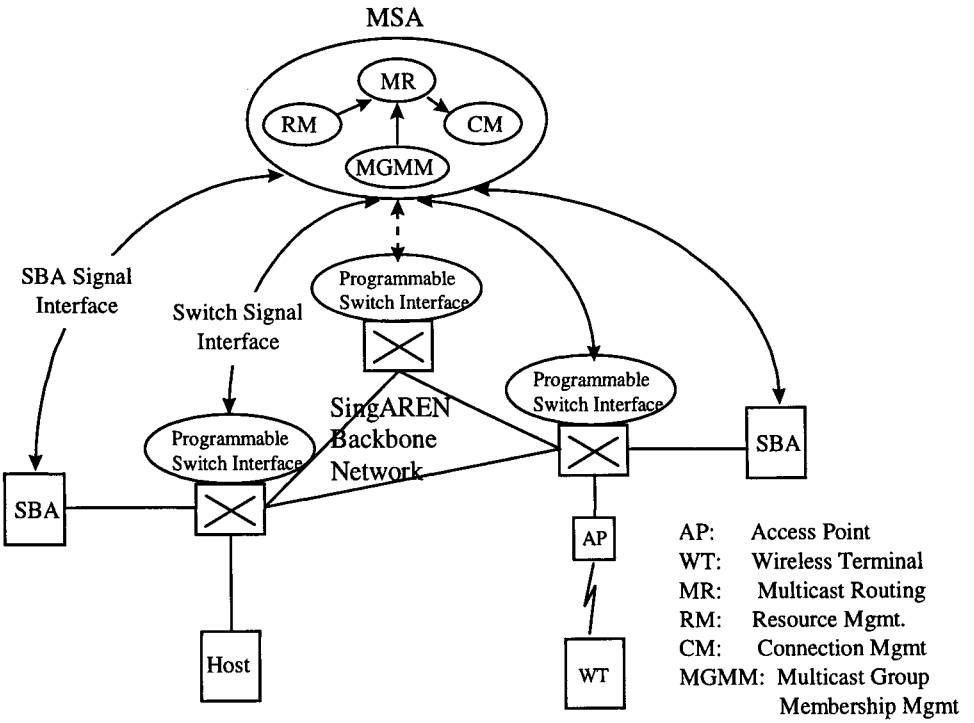


FIGURE 5 Components of the proposed ATM multicast service.

protocol (see Section VII. B). In fact it will be explained later in this chapter that same design, which provides multicast connection service for ATM networks with only "fixed" hosts, can be used to support mobile hosts with virtually no change.

VI. PROPOSED DESIGN

Figure 5 depicts an implementation of the proposed multicast service architecture in a small ATM network consisting of several multivendor switches. Components of the architecture include the MSA, the host-end signaling interface called the *service binding agent* (SBA), and the switch-end signaling agent called the *virtual switch* (VirSwt). The way that these components works is according to the following sequence. To join the group, an application (e.g., video-conferencing application on a host) will first request to join the specific multicast group as either a sender or a receiver, or both, by notifying the MSA through the SBA. The MSA through its subagents will be responsible for adding the host to the multicast group and working out a suitable VC route before returning the assigned VC identifier(s) to the requesting host via the SBA. A similar sequence also applies to the group leave operation.

A prototype of the proposed multicast service over a group of six SingAREN ATM switches supplied by three leading vendors has been implemented. These switches are connecting to both Unix workstations (at

155 Mbits/s, OC-3 multimode fiber) and PCs (at 25 Mbits/s, UTP-5). It has been successfully tested on the setting up of both multicast and unicast connections, as well as running video and audio applications over these premium service connections. Also verified is the feasibility of supporting IP over VC multicast connections. In the rest of this section, the various components of the proposed ATM multicast service are presented. The reader is also referred to related papers [10,11] for a more detailed description of the proposed design.

A. End-User Multicast Signaling Interface—Service Binding Agent (SBA)

As described above, the SBA acts as the interface between the end-users and MSA for the purpose of accessing the multicast premium service. Given the goal of supporting the various properties similar to that of an IP multicast as outlined in Section IV, SBA therefore supports a programmatic interface similar to that of the IETF Resource Reservation Protocol (RSVP) [12,13]. In addition, SBA provides the function of giving the application transparent access to the ATM multicast service (via MSA) and, if necessary, performs service binding for an IP-level multicast (e.g., binding an ATM VC end-point to its corresponding IP address).

The way that SBA is implemented is clearly dependent on the intended platform and environment. For example, SBA can be realized as a set of library calls that can be linked into a brand new networking application, or run as specialized "blackbox" hardware with necessary software and operating system to provide premium service access. With the current SingAREN prototype, a version of SBA implemented and accessible through a set of World Wide Web HTML pages has been done. This allows each SingAREN site to access the service via virtually any host on the network. Using the Web page, the requesting site is able to indicate the required QOS and how the premium service is to apply to IP traffic flows generated at either host level or application level. Once the service has been successfully obtained the subsequent selected traffic will be diverted and carried over the connection(s) setup by the premium service transparently. The same SBA also provides mechanisms to be used to unsubscribe from the premium service.

B. Switch-End Signaling Interface—Virtual Switch

In this section, we concentrate on the software that is added to operate alongside the ATM switches in order to provide the necessary multicast service, using the ATM cloud as shown in Fig. 5. The approach involves adding a software component called a "virtual switch" (VirSwt) in a host attached to each of the switches. This software provides a common interface to switch functions such as setting up and tearing down of VCs, as well as modifying the QOS parameters associated with each VC. To effect the actual change in each switch, however, it is important to note that since different switch vendors have designed their switch interface differently, to perform the above switch-related functions on these various switches requires an appropriate switch-dependent code. Examples of these access mechanisms are the switch-supplied serial-line (RS-232) management software, the Simple Network Management Protocol

(SNMP), and more recently the General Switch Management Protocol (GSMP) [14]. VirSwt's communicate directly with the MSA via the available communication channel (e.g., IP interface) between them.

Recent research activities in the areas of *active network* [35] also proposes the use of highly flexible and programmable software components in the switching nodes. The VirSwt proposed here clearly has the potential of being expanded to incorporate some other functions such as intelligent data filtering and merging of IP flows not well supported in today's ATM switch hardware [21,23,25]. However it is not clear to the authors how these functions should be realized effectively and implemented in VirSwt or even in MSA (see below). For now, VirSwt will simply provide the uniform access to the native ATM switch interfaces.

VII. MULTICAST SERVICE AGENT (MSA)

In this section, details of the various MSA components as shown in Fig. 5 are described. Each MSA is expected to perform the following four functions, namely, *resource management, group management, routing,* and actual physical *connection management.* For clarity, we will be describing each of these functions as independent "agents"; it does not, however, represent the way that they are realized in the actual implementation. It is important to note that an MSA is expected to provide the proposed connection service to a specific group of switches/hosts (called "domain"); multiple MSAs will be required and deployed in a distributed fashion in a large network, which is described separately in Section VIII.

A. Resource Management Agent (RMA)

The RMA of a network domain uses a link state database (LSD) to manage the network resources and provide connectivity information for routing computation. The physical links of a network domain is uniquely identified by the combination of the link's node identifier (ID) and a locally assigned port ID. For example, if the node is a switch, the switch-assigned ATM address is used as node ID; for a host, it may be the host ID (see Section VII.B.1). The RMA manages each link in terms of *link capacity,* which includes the total available bandwidth, total available VPI/VCI pairs, and the propagation delay on the link. In order to coexist with the standard signaling protocols, the RMA defines and utilizes a subset of the available VCI/VPI space on a link (e.g., space reserved for permanent virtual channels (PVCs)). Table 2 lists the VPI/VCI space used on various ATM switches in our implementation.

RMA therefore manages the resource allocation for the connection setup requests from any host of its network "domain" with its LSD reflecting accurately the available resource pattern in the network domain managed by its MSA. The link state information can be used for calculating the expected call admission behavior at each switch in the network domain. Whenever a resource query of a connection request is received by the RMA, it performs a call admission control (CAC) function based on the traffic parameters and requested QOS

TABLE 2 Examples of Multicast VPI/VCI Space in Commercial Switches

Switch type	Virtual path IDs	Virtual channel IDs
FORE-100 FORE-200WG FORE-200BX FORE-1000	32–47	256–1023
Scorpio	32–47	768–1535
ATML	32–47	256–1023

of the connection. The RMA determines whether setting up the new connection violates the QOS guarantee of existing connections in the network domain.

The link state database also records the current connectivity information of the network domain. Therefore, the current network topology can be derived based on the connectivity information of the link state database. The topology of a network can be expressed by a connectivity matrix that varies according to the QOS requirement of a connection request. All the links that cannot satisfy the requested QOS are pruned from the set of all possible paths for the connection. With this reduced set of links, the routing algorithm can compute the shortest paths for any connection request.

B. Group Management Agent (GMA)

The GMA provides an important function of the multicast service by managing the membership status and member joining or leaving activities. GMA keeps a database of multicast group names and the corresponding physical addresses of the existing group members (see later), so that whenever a group operation is being performed, the relevant group information can be looked up or updated accordingly. Other information kept by GMA includes the terms of QOS parameters needing to be guaranteed during the data transmission and the user-specified filters at each VirSwt.

In a network where multiple MSA domains exist (see Section VIII), a group management protocol to allow the respective GMAs to exchange individual domain group membership information is required. One significant observation made when designing a GMA is that it is also providing the required *location management* function, which is needed in a mobile environment. A process often referred to as the *location registration* is carried out whenever a host moves to a new location (see Section IX for more details). This is so that it (the mobile host) can be located and reached by the connection service subsequently. It is observed that this same function is provided in a GMA whenever a host "leaves" a group but "joins" it again at a new physical location, thus, leading to the conclusion that the proposed design can be made to handle mobile hosts with little or no change. With this intention in mind, a suitable group naming and host addressing scheme should therefore be proposed to ensure that a GMA also plays the role of a location manager. This is described next.

1. Host Addressing Scheme & Multicast Group Naming

Any signaling protocol requires an addressing scheme to allow the signaling or routing protocols to identify the source and the destinations of a connection. Most of the current host addressing schemes have dual natures such that an address not only identifies the name but also specifies the location of a host in the network. This type of addressing scheme is generally location sensitive in the way that the exact location of a host is embedded in the address of the host. In a mobile environment, the association of a mobile host name with a location is transient and should therefore be treated separately. In the proposed design, a *name* is an identifier of a host and is assigned for its lifetime while the *physical address* indicates the current location of a host and is bound to a host for a transient period. When a host is attached to the network the association of a host name and the physical address is registered with the GMA. Any connection setup call uses only the name of the destination host and the GMA is responsible for finding out the corresponding physical location. This is a reasonable scheme in a mobility-enabled network because the calling party need only to know the name of the called party and the actual physical address of the called party is resolved by the "network."

Furthermore, beside using individual host names as the group identifiers as described above when creating a multicast connection, other location-independent logical names can also be used to name a group. To facilitate multiple multicast groups in a host, a logical "session identifier" equivalent to the IP port number should also be added to the group identifier. In summary this section outlines the various considerations of a suitable group naming and host addressing scheme; the exact format, management, and assignment of the addressing scheme is beyond the scope of this chapter.

C. Routing Agent (RA)

The Routing Agent computes a multicast tree that connects producer and consumers by using the link state information in RMA, group membership information, and negotiated agreement in GMA. It also offers routing options to the receivers that may emphasize different factors on data transmission, such as reliability, real-time, or cost effective paths. When an RA computes the multicast tree, it selects the strategic switches to invoke user-specified programs such as filters. The results of an RA are a set of contracts for the (virtual) switches involved in the computed multicast tree.

1. QOS-Based Routing

Traditional multicast routing algorithms were designed to support users with homogeneous and simple QOS requirements. With increasing demand for integrated services, the routing algorithms must support diverse, fine-grain, and subjective QOS requirements. In a connection-oriented network such as ATM, the data transferring between end users is accomplished by network routing functions that select and allocate network resources along acceptable paths. In our routing scheme, routing is performed by a centralized RA in the network domain. It is obvious that the merit of this type of routing partly depends on the

accurate information of the network topology and link state. This is similar to the current link state routing protocol such as OSPF where all network nodes can obtain the estimated current state of the entire network [22].

The network can be modeled as a graph of nodes connected by node-to-node links. Each link has parameters, maximum cell transfer delay (MCTD), and residual cell rate (RCR). These values are known to the routing algorithm and we use t, b to represent MCTD, RCR in the distance function. A simplified distance function d_{ij} reflects transfer delay t_{ij}, and residual bandwidth b_{ij} is defined empirically for each direct link $l_{ij} : d_{ij}(t_{ij}, b_{ij}) = \omega_1 t_{ij} + \omega_2(1/b_{ij})^{\alpha}$, where l_{ij} denotes the link from node i to node j. The exponential form of a residual bandwidth term compared with the transfer delay term's linear form shows the emphasis on bandwidth to be exhaustive as a much more important factor affecting the distance function. The w_1, w_2 are weights of the two factors that can be changed by the users in the negotiation phase. Increasing the weight of a factor may change the emphasis of the routing algorithm. For instance, if a receiver wants to receive a real-time data stream, it can increase the value of w_2, which leads the routing algorithm to emphasize more the propagation during the routing computation.

By using the Dijkstra algorithm [19,24], the shortest path from source node to the receiver nodes can be computed. If the paths leading to different receivers travel along the same link, they need to be merged into a single path. After all the common paths are merged, a multicast tree can be established. In the merging process, QOS reservations for different receivers need to be aggregated to form a resource reserved multicast tree. As a result, only the essential (i.e., the minimum superset) common QOS is reserved on a link for a common path. A filter that is split from the common link to facilitate the selective data transmission is assigned with each outgoing path. These filters select the requested data set for the specified receivers according to their QOS requests. Therefore, a logical multicast tree with various QOS reservations on different branches can be established. Note that all the actions described in the above paragraphs, such as routing computation, resource reservation, and filter placement, are performed with the memory of an RA (i.e., we only have a logical multicast tree at this stage). Subsequently, the logical multicast tree will be converted into separate contracts for all the (virtual) switches involved to interpret and implement.

2. Node Joining / Leaving a Multicast Tree

A new member can join an existing multicast group through a joining operation. In the operation, an unicast route is first computed from the source to the joining node by an RA based on the QOS request. Then the joining algorithm will try to merge this new route to the existing routing tree. It first traces the new route from the joining node toward the source until it hits the first intermediate node of the existing multicast tree that has sufficient resources to meet the QOS requirements of the new joining node. This intermediate node is selected as the merging point for the remaining links to the source. In the worst case, the attach point is the source. Figure 6 shows an example of a node R4 joining an existing multicast tree in which the sender S is sending a data stream to receivers R1, R2, R3 via a switch network involving Sw1, Sw2, Sw3,

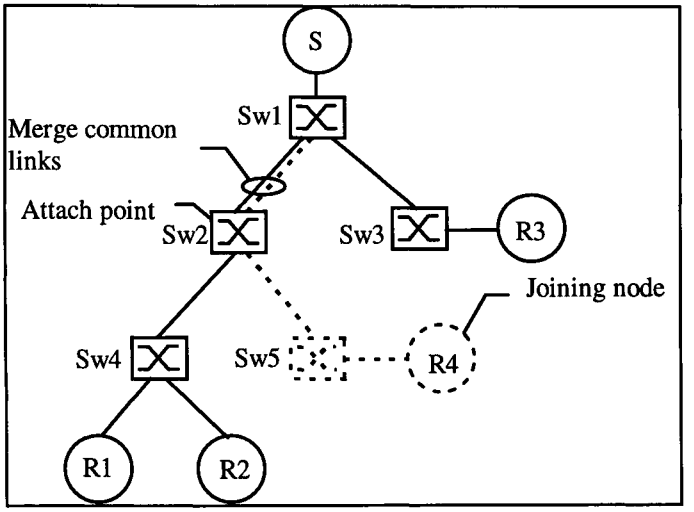


FIGURE 6 Dynamic Member Join/Leave Scheme

Sw4. The unicast route from source to receiver R4 is indicated by the dash lines. The attach point for the joining node is in Sw2. The unicast route from S to R4 has a common link with the existing multicast tree at the link between Sw1 and Sw2. This common path needs to be merged into the same path.

The leave operation removes the leaving node from all the multicast trees currently sending data to it. As in the joining algorithm, the node is disconnected with each multicast tree separately. For instance, if the receiver R4 wants to leave the multicast tree as shown in Fig. 6, the leaving algorithm first disconnects the R4 and then releases the reserved resources for it. The intermediate switch Sw5 checks whether there exists other nodes still attached to itself in the multicast tree. It will remove itself from the multicast tree if there is no group member attached to it. Otherwise, the intermediate node frees the resources that were reserved for the leaving node.

D. Connection Management Agent (CMA)

Once routing contract for each switch is worked out, the CMA is invoked to implement the actual multicast tree setup by sending the routing contracts into the designate switches via the links from the MSA to the switches. A multicast channel in a switch is thought to have one incoming virtual channel and multiple outgoing virtual channels. The routing contract uses the inPort and outPort to specify the switch to be connected to the incoming VPI/VCI and the switches to be connected to the outgoing VCI/VPI. When switches (via VirSwts) receive their routing contracts, each of them can start to setup the upstream and downstream connections for establishing the corresponding VPI/VCIs. The signaling of the switches for establishing upstream and downstream connections is a parallel process in which each switch works on its own contract independently. When the switches complete the contracts designated for them, response messages will

be sent back to the CMA to confirm the success of the connection setup. CMA will forward the response message to the caller after it receives the responses from all the involved switches. In this way, the switches, producers, and receivers can be bound into one multicast session.

E. Performance Analysis for Connection Setup

This section gives performance analysis on connection setup time to the proposed MSA connection setup scheme. It can be proved that the MSA's parallel connections setup scheme uses less time than the widely used hop-by-hop connection setup scheme in setting up nontrivial connections which traverse multiple hops. Since the following analysis uses the terms such as eccentricity, radius, and diameter of a graph, we give the definitions of these terms in the next paragraph.

The topology of a network domain can be represented by a graph $G(V, E)$, which consists of a set V of elements called vertices and a set E of edges that connect the vertices. We define a path from a vertex u to a vertex v as an alternation sequence of vertices and edges $\{v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k\}$, where $v_1 = u, v_k = v$, all the vertices and edges in the sequence are distinct, and successive vertices v_i and v_{i+1} are endpoints of the intermediate edge e_i. We say a pair of vertices u and v in a graph are connected if and only if there is a path from u to v. If every pair of vertices in G is connected, then we say the graph G is connected. In a connected graph, the path of least length from vertex u to a vertex v in G is called the shortest path from u to v, and its length is called the distance from u to v. The eccentricity of a vertex v is defined as the distance from v to the most distant vertex from v. A vertex of minimum eccentricity is called a center of G. The eccentricity of a center of G is called a radius of G denoted by $R(G)$, and the maximum eccentricity among all the vertices of G is called the diameter denoted by $D(G)$. A subgraph S of a graph $G(V, E)$ is defined as a graph $S(V', E')$ such that V' is a subset of V and E' is a subset in E, and the endpoints of an edge in E' are also in V'. We use the $G-V'$ to denote the induced subgraph on $V-V'$. The border node v of a graph $G(V, E)$ is defined as a vertex $v \in V$ that is linked to a vertex v' and $v' \notin V$. Assume V' is the set of border nodes of $G(V, E)$, and the set of interior nodes is denoted by $V-V'$.

The left graph in Fig. 7 shows an example graph where $|V| = 8, |E| = 15$. Four of the vertices are border nodes and each of them has one exterior link. In this graph, border nodes $V' = \{a, c, d, f\}$; interior nodes $V-V' = \{b, e, g, h\}$. The numbers associated with the edges are the length or the distance between the two nodes. The distance is used to represent aggregated effort for transmitting data between two nodes. Based on the distance pattern of the graph, we can calculate the center of the graph as node h with the eccentricity equal to 5, and hence the radius of the graph is $R(G) = 5$. The diameter $D(G) = 9$ is determined by the most distance pair of nodes a and d. The summarized distance information between pairs of border nodes is shown in the right graph of Fig. 7, where the distance between the border nodes is indicated by underlined numbers on the graph. The summarized distance information enables a routing algorithm to compute the paths crossing the network domain.

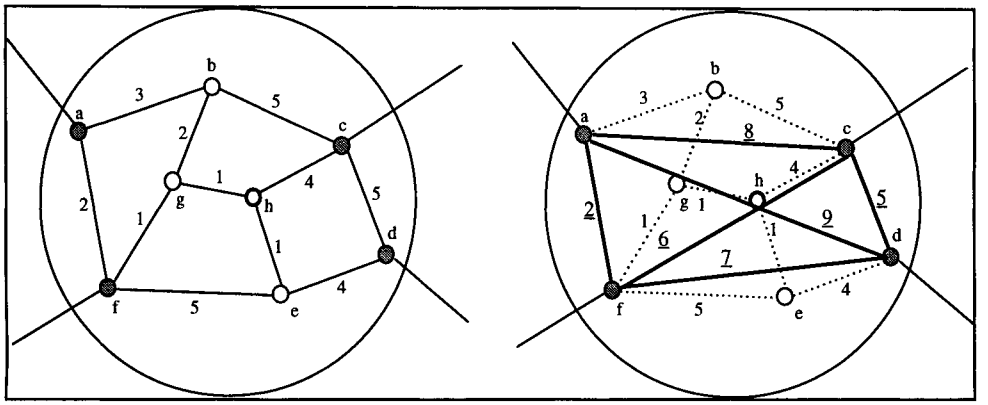


FIGURE 7 (left) A graph representation of a network domain. (right) The corresponding summarized graph.

Ideally, the MSA of a network domain is connected to the center node such that the maximum distance from the MSA to any node in the domain is less than or equal to the radius of the network. Assume the time for sending a signal message is proportional to the distance from the MSA to a node, then the time used for sending a signal message from a MSA to a switch node is $t_m = \bar{\omega}\delta$, where δ is the distance between the MSA and the node; $\bar{\omega}$ is a normalizing factor that results in time scale. Assume the time for setting up a connection in any switch is identical and is denoted by t_s. The time used by the MSA to signal the switch S and to get back the response from S is $2\bar{\omega}\delta$. Therefore, the total time for setting up a connection at switch S is $t^s_{\text{setup}} = 2\bar{\omega}\delta + t_s$. Since MSA is at the center of a network domain and its distance to any node in the domain is less than or equal to $R(G)$, we can conclude that for any node in the network domain, the connection setup time is less than or equal to $2\bar{\omega}R(G) + t_s$. The routing contracts are sent to the switches in a parallel domain by the CMA such that the time used on setting up any connection in the domain depends on the switch that causes the longest setup time delay. As mentioned above, this longest setup time should be less than or equal to $2\bar{\omega}R(G) + t_s$ and hence if the configuration of a network is determined, the time used for setting up any connections within the domain is bounded by $2\bar{\omega}R(G) + t_s$ no matter how many hops the connection traverses.

VIII. SCALING UP TO LARGE NETWORKS WITH MULTIPLE MSAs

The growth of ATM users demands an increasing number of ATM switches to be interconnected to support large numbers of ATM hosts. The current MSA was initially designed for a small ATM network with a few dozens of nodes [11]. As the number of nodes continues to increase, the size of the databases in the MSA and the intensity of the service requests to the MSA will continue to grow. To scale up for the future ATM Internet comprising larger number of switches, the network needs to be partitioned into multiple

domains. This is mainly because single MSA will become a bottleneck when there are large amounts of connection requests. In addition to that, routing algorithms will become inefficient when the number of nodes is large. An ATM network can be partitioned into nonintersection domains, and each domain has its own unique domain ID. Each network domain is assigned an MSA to handle all the connection requests within the domain, and hence we name the MSA as the domain MSA (D-MSA) in large ATM networks. The D-MSA processes all the required routing and connection setup activities within its domain, and it also floods the summarized reachability information to all D-MSAs in other network domains. The reachability information includes the propagation delay and available resource information between each pairs of border nodes in the same domain as described in the right graph of Fig. 7. Summarizing reachability information of the border nodes hides the internal connectivity information from other D-MSAs. The D-MSAs trigger the flooding mechanism in an on-demand fashion (i.e., as long as there is a change in the summarized topological or reachibility information, the flooding mechanism will be triggered). With the flooding mechanism, each D-MSA can obtain detailed topological and link state information of its own domain plus the summarized topological and link state information of all other domains.

Once the link state information is obtained by all D-MSAs, they can use this information to compute routes for connection requests. In a mobility-enabled network, the actual location of a host is determined by the current attachment not traced by the D-MSA at the calling party's domain. Therefore, two phases are required for setting up a connection in our approach: The first phase is to discover the current location of a host, and the second phase is to setup the connections. For setting up a connection, the requests are always sent to the D-MSA of the caller's domain. When a caller's D-MSA receives a connection request with the names {caller, callee}, it checks the name of the callee in the GMA to examine whether it is in the same domain as the caller. If both the caller and the callee are in the same domain, the caller's D-MSA can handle the request with the intradomain routing and call setup procedure as described in Section VII.C. Otherwise, the caller's D-MSA multicasts the request to all other D-MSAs in the network. This multicast message is extended to include the distance information from the caller to each of the border nodes in the caller domain. This distance information can be used for the D-MSA at the callee's domain to perform "destination" routing computation as described in the next section.

A. Routing and Connection Setup in Wide Area ATM Network

One of the most important routing issues is to find the shortest path from the source to the destination. The data transmission could be bidirectional where the exact meaning of source and destination is blurred. For simplicity, we assume source as the *caller* and destination as the *callee* of a connection in later discussions. In most of the hierarchically organized network routing protocols, source-based routing algorithms such as hierarchical OSPF [22] for the Internet and PNNI [16,20] for ATM networks are used. In these protocols, the source router or switch normally has the detailed topological information

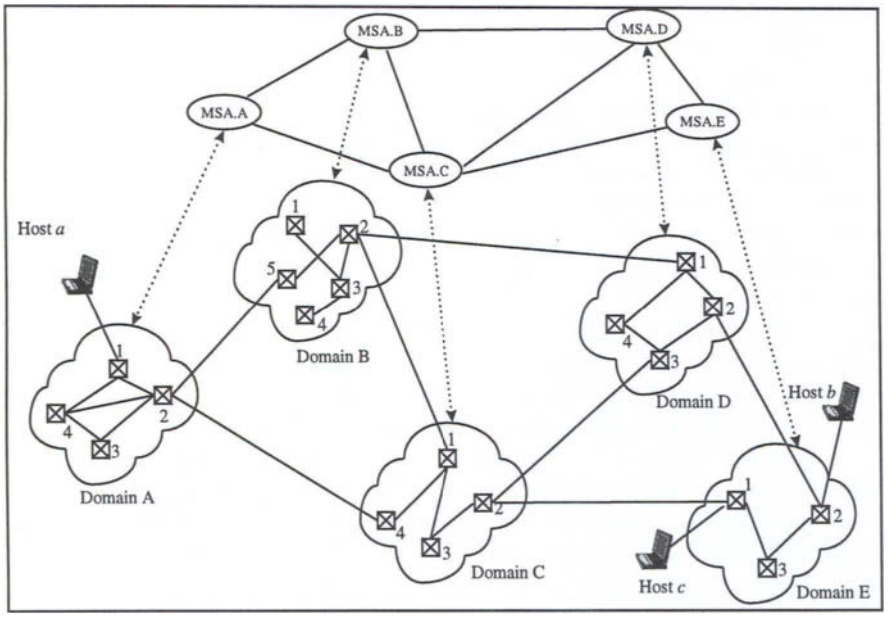


FIGURE 8 Proposed solution deployed over a large ATM network.

of its own domain and the summarized information of other domains. The nonoptimal path problem arises when the source does not know the detailed topology in the destination domain. The reason is that the shortest paths from a source to different hosts in another domain may not necessarily travel to the same border node at the destination domain. Figure 8 shows a multiple-domain ATM network, where the capital letters. *A–E* denote domain names, the digits 1–5 denote switch ID, and the lower case letters *a–c* denote host names. For instance, A_n means switch n at domain A, D-MSA.X represents the D-MSA at domain X. Assume the source host a is in domain A, and the two destination hosts b and c are in domain E. If we use the number of hops as the distance of a path from a source to a destination, the shortest path between host a and host b is $\overline{ab} = \{A_1, A_2, B_5, B_2, D_1, D_2, E_2\}$ and the distance between host a and host b is $|ab| = 6$. The shortest path for route $\overline{ac} = \{A_1, A_2, C_4, C_1, C_3, C_2, E_1\}$ travels through domain C instead of domain B and D used for path $\overline{ab}$. The distance between host a and host c is $|ac| = 6$. From this example one can find out that if the internal topology of domain E is not known by the MSA.A, there is no way for MSA.A to determine how to find a shortest path to an internal host in domain E.

B. "Destination" Routing and Connection Setup

To solve the suboptimal route problem, we propose using a "destination" routing scheme, where the D-MSA at the destination domain is responsible for setting up the connection from the source to the destination. The process for setting up a connection has two phases, namely, the host discovery phase and

the connection setup phase. The procedures of the two phases are described as follows:

- Host discovery phase has four steps:

(1) Connection request is sent to the D-MSA of the source domain.

(2) The source D-MSA looks for the destination host in the GMA of the source domain. If the source and the destination are in the same domain, the D-MSA of the source is responsible for setting up the connection by using intradomain routing and connection setup protocol described in Section VII.C. Otherwise go to step 3.

(3) The D-MSA sends an extended connection *setup*(S, D) message to all the D-MSAs in the network. S and D stand for source and destination respectively. The *setup*(S, D) message has the distance information from the source to each of the border nodes in the source domain.

(4) When the *setup*(S, D) message reaches the destination domain, the D-MSA at the destination domain obtains the distance information from the source host to the border nodes at the source domain, the detailed topology information of its own domain, and the summarized distance information of the intermediate domains. With all the information mentioned above, the destination D-MSA is able to compute the shortest path from the source to the destination.

- Connection setup phase has four steps:

(1) The D-MSA of the destination domain computes the shortest path (SP) from the source to the destination by using the Dijkstra algorithm and constructs a set of routing contracts for each of the D-MSAs in the domains that the SP traverses. The routing contracts specify the requirements for setting up the partial paths within the involved domains.

(2) The D-MSA of the destination domain sends the set of routing contracts to all the D-MSAs of the domains over which the SP traverses; the message is called *setup(contract)*. The contracts are encapsulated into a signaling packet and the packet is transmitted from the destination D-MSA to the source D-MSA along the signaling path that connects all the involved D-MSAs in the domains that the SP crosses through. Each D-MSA takes the contract designated for its own domain and forwards the rest to its upstream D-MSA.

(3) Upon receiving the routing contracts,

- The D-MSA of the destination domain sets up the connection from the border node to the destination host and sets up the interdomain connection to the border node of its upstream domain on the SP.
- All the intermediate D-MSAs set up partial paths from the upstream border node to the downstream border node on the SP in their domains. They also set up the path from the downstream border nodes at their upstream domains to the upstream border node at their own domain.
- The source D-MSA is responsible for setting up connections from the source to the border node on the SP at the source domain.

(4) A *response* message is sent from the destination D-MSA just after the *setup(contract)* message. The *response* message travels along the same route as the *setup(contract)* message. The function of the *response* message is to wait for each of the D-MSAs completing the tasks specified in

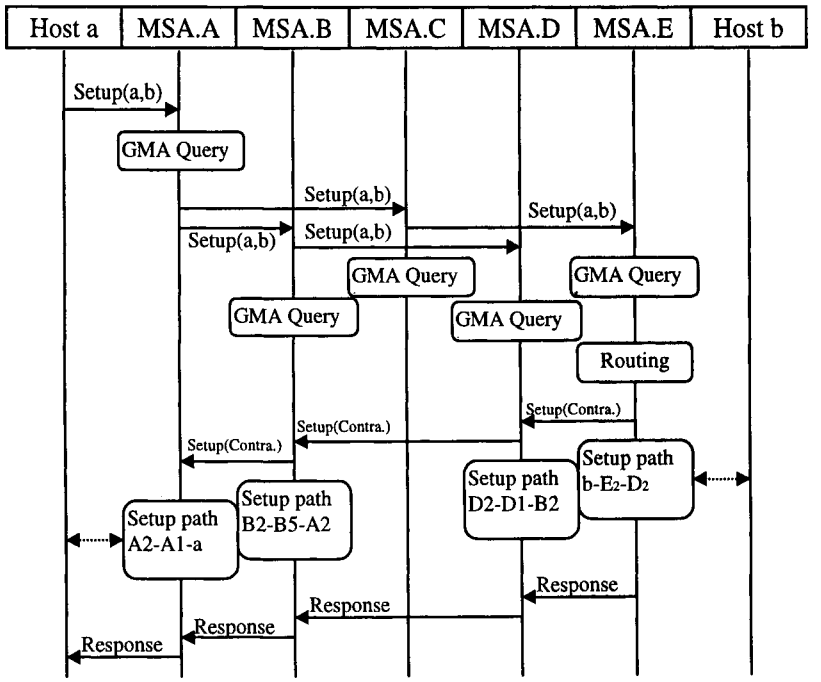


FIGURE 9 Interaction graph for connection setup in a multiple-domain network.

the contract. When the *response* message reaches the source D-MSA, it will be forwarded to the source host to confirm the success of the connection request.

Let's describe the connection setup process by using an example that sets up a connection from host a to host c in the multiple-domain network in Fig. 8. In this network, there are five domains (i.e., domains A–E) and each of them has a D-MSA that controls the connection setup in its domain. Each domain has a couple of switches and host a is linked to switch A_1 of domain A and host b is linked to switch E_2 in domain E. We describe this connection setup by using an interaction diagram with signaling descriptions.

As shown in Fig. 9, host a sends a *setup(a, b)* message to the MSA.A. MSA.A look up host b in the GMA of its domain. Since host b is not in domain A, a multicast message will be sent to MSA.B, MSA.C, MSA.D, and MSA.E. Then MSA.E finds out that host b is in domain E after querying the GMA. MSA.E computes the shortest path from a to b and constructs the routing contracts for MSA.D, MSA.B, and MSA.A. The *setup(contract)* message is sent along the signaling path {MSA.E, MSA.D, MSA.B, MSA.A, a}. When MSA.E completes setting up the partial path {b, E_2, D_2}, it will send out the *response* message toward host a along the signaling path {MSA.E, MSA.D, MSA.B, MSA.A, a}. The *response* message waits for each MSA on the route to complete its contract and then is forwarded to the upstream MSA. When host a receives the *response* message, it can start to transmit data to host b.

C. Performance Analysis for "Destination" Routing and Connection Setup

The time for setting up a connection in "destination" routing and connection setup scheme is determined by the time interval between sending out the *setup* message and receiving the *response* message. The signaling path for setting up a connection from a source to a destination can be represented by a chain {MSA.0, MSA.2,..., MSA.n}, where MSA.0 is at the source domain and MSA.n is at the destination domain. The time for propagating the setup message to the destination MSA is $t_{\text{setup}} = \sum_{i=1}^{n} \bar{\omega}\delta_i$, where the δ_i is the distance between MSA.i and MSA.$i-1$. The time for the MSA.n to look up the GMA and compute the routing is represented by t_{route}. The setup contract message travels along the path {MSA.m, MSA.$m-1$,..., MSA.0} where MSA.m is in the destination domain and the MSA.0 is in the source domain. Note that m may not be equal to n because some domains on the data path may not necessarily be on the signaling path for the *setup* signal. The time for the ith D-MSA to complete its contract is represented by t^i_{contract}. Assume the maximum time used for completing the setup contract among all the D-MSAs is $t_{\text{max-contract}} = \max_i\{t^i_{\text{contract}} \mid 0 \le i \le m\}$, which is incurred by one of D-MSAs in the *setup(contract)* message chain. The time for propagating the response message is $t_{\text{response}} = \sum_{i=1}^{m} \bar{\omega}\delta_i$, where the δ_i is the distance between MSA.i and MSA.$i-1$. We can add up these time periods into the total connection setup time t_{total}, which consists of four time periods: the propagation time for the *setup* message to travel from the source to the destination; the time for the destination D-MSA to look up the destination host and compute the route; the maximum time used by a D-MSA on the computed route to complete its contract; and finally the time used for propagating the *response* message. Therefore the total time for setting up a connection crossing different domains is

$$\begin{aligned} t_{\text{total}} &= t_{\text{setup}} + t_{\text{route}} + t_{\text{max-contract}} + t_{\text{response}} \\ &= \sum_{i=1}^{n} \bar{\omega}\delta_i + t_{\text{route}} + t_{\text{max-contract}} + \sum_{j=1}^{m} \bar{\omega}\delta_j. \end{aligned}$$

If the maximum contract implement delay is incurred by the MSA.L that is on the chain for *setup(contract)* message, we can replace the third term $t_{\text{maxcontract}}$ with the upper limit time for an intradomain connection setup such that

$$t_{\text{total}} \le \sum_{i=1}^{n} \bar{\omega}\delta_i + t_{\text{route}} + 2\bar{\omega}(R(G_l) + \delta_l) + t_s + \sum_{j=1}^{m} \bar{\omega}\delta_j,$$

where $R(G_l)$ is the radius of the domain in which the D-MSA causes the longest delay for implementing the *setup(contract)*, δ_l is the distance from the upstream border node of domain l to the downstream border node at the domain l's upstream domain, and t_s is the time for setting up a connection in a single switch.

The "destination" routing and connection setup scheme described here is different from other ATM network routing protocols such as the PNNI [16]. For a start, this scheme integrates multicast group management, routing, and connection setup functions into one. It also guarantees to find the "shortest" route for setting up connections in a wide area ATM network—something

which PNNI cannot achieve [20]. Furthermore, since the proposed routing and connection setup scheme uses D-MSAs to exchange link state information and to compute routes, it effectively reduces the burden on the switches, which were not designed for performing heavy routing computation and storing large amounts of link state information. One may argue that using multicast messages among D-MSAs for connection setup is too heavy in the proposed routing and connection setup protocol. However, further optimization techniques such as caching can be applied to the design so that the *setup* message can be sent directly to the destination DMSAs of popular destinations, without relying on multicasting a "query" message first.

D. Multicast Connection Setup

The procedure for setting up multicast connections is similar to the unicast connection setup case. A multicast session consists of one data source and a number of receivers that form a group. Each multicast session is assigned a network-wide unique connection identifier that is used to identify this multicast connection over the whole network. When a data source requests to send data to a group, the source D-MSA assigns a new connection_id to the multicast session. The source D-MSA multicasts a setup message to all the D-MSAs in the network. This setup message contains the group name and the unique connection_id. After receiving the setup message, each D-MSA looks up the GMA for the group members. If there are group members in its domain, the D-MSA performs a "destination" routing and connection setup procedure separately for each of the group members. When any intermediate D-MSA receives setup contract message for its domain, it compares the connection_id of the new connection with the connection_id of the existing connections node by node from the downstream border node to the upstream border node in its domain. If the new connection_id is the same as an existing connection_id, and the upstream border nodes for the new connection are coincident with an existing connection with the same connection_id, then this D-MSA attempts to identify the common node for the two connections. The paths of the two connections can be merged from the common node to the upstream common border node such that only one upstream subpath is used for the common path of the two connections. If all the involved D-MSAs detect common nodes for the new connections and merge a subpath with the existing ones whenever it is possible, then multicast trees can be formed for multicast sessions. Figure 10 shows the merging process for an existing connections $C_1 = \{c, h, g, f\}$ and a new connection $C_2 = \{e, h, g, f\}$ that have the node h as their common node and f as their common upstream border node. The subpath $\{h, g, f\}$ of the new connection is merged with the existing one.

E. MAS Signaling Network

It should be obvious by now that the proposed solution assumes the existence of signaling and control channels in which communications between an MSA and other components such as the SBA and switches are under control. These channels, which are referred to here as the "signaling network", can be provided

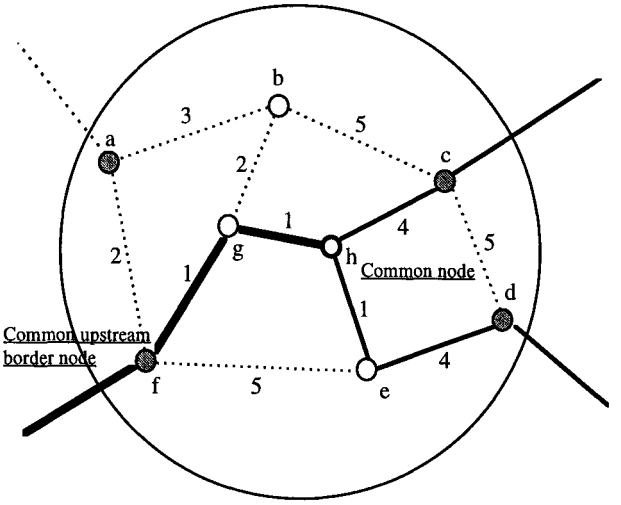


FIGURE 10 Merged common paths from node *h* to node *f*.

in a variety of ways. However, given that SingAREN already supports a best-effort IP service to all hosts and switches concerned, it is used here to provide the required communications. Clearly the performance (i.e., QOS) of this network will also impact the overall performance of the proposed signaling solution. Given that this arrangement is adequate for the current implementation, the real impact is left for future studies.

The initial tests reveal that normally it takes between 1 and 2 for a requested service to be completed on the SingAREN network over a maximum of four interconnected switches. This typically involves the sending of a service request via the SBA to the MSA, and for the MSA to calculate and effect a connection to be setup via the virtual switch(es) before returning an acknowledgement to the requesting host. More studies are needed to verify it with the analytical results presented earlier. No tests involving wireless network have been conducted so far.

IX. HOST MOBILITY SUPPORT

Three basic issues need to be addressed in the mobility-enabled network protocols: location management, connection management, and handover management. Location management is a generic capability required in both the fixed network and the mobility-enabled network (i.e., wired or wireless network). Location management provides an association between a unique host identifier and a network access point, which is used to locate the current network point to which the host is attached. Handover management deals with dynamic host migration that requires signaling and network control for dynamic rerouting a set of existing connections from one access point to another. In general, this process involves a host or network initiation of handover for several connections that may be connected to different fixed host or mobile hosts (or terminals (MT)).

Although the wireless ATM is still in its infancy stage, many proposals are being worked out to address the three basic mobility management issues [28,29]. For the location management, an obvious way is to adopt methods similar to those used in cellular/PCS systems, which use the Home Location Registration (HLR) and Foreign Location Registration (FLR). Some value-added techniques to be applied to the HLR/FLR approach for improving the performance on address resolution such as caching and hierarchically organizing the address databases have also been proposed [27]. Another approach for location management is to extend current PNNI signaling mechanisms to integrate location discovery of a mobile terminal with the connection setup. In both approaches a network is responsible for translating the original destination address (i.e., mobile terminals home address) used by a caller into a topologically significant address and then reroute the call to the mobile's current location. The major differences lie in where the address translation is performed in these schemes. In the extended PNNI approach, address translation is performed by looking at an additional translation table at the mobile-enhanced ATM switches in the mobile's home domain while the HLR/FLR approach requires external services for address translation to be deployed in the ATM network. There are several shortcomings in these proposed location management schemes: Firstly, a mobility tracking mechanism is required for the network to trace the movement of each mobile terminal, which means additional tracking and messaging protocols need to be deployed in the network. Secondly, the signals for setting up connections always travel to the home location of the mobile terminal to find out the current location of the callee and then reroute the signal to its current location. Furthermore, both of the proposals need the mobility-enhanced ATM switch to interpret the connection setup messages, which makes the signaling software even more complex.

A. Proposed Approach

An integrated approach as proposed in this chapter, which deals with both mobile terminals and fixed terminals in a similar way, would be more appropriate for future mobility-enabled ATM networks with large numbers of mobile terminals. It should be clear by now that the proposed multicast service solution has some important features that make it suitable to support host mobility. For clarity, these are further elaborated here.

(1) By using a logical multicast group identifier to associate any connection, hosts can move and be connected freely anywhere in the network.

(2) The leaf-initiated connection request supported via the SBA for both the sender and receiver ends allow maximum flexibility for each party to move independently of each other. More importantly (not mentioned previously), SBA can further support the concept of "advance-join," which allows join connection requests on a new location to be initiated in advance. Using this feature in the mobile environment, a mobile host is able to request in advance a new connection to be negotiated and setup in its next location before the actual wireless handover takes place.

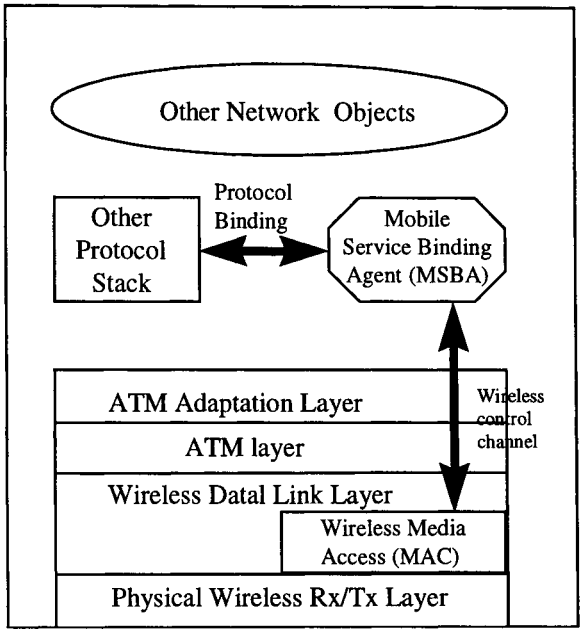


FIGURE 11 Wireless Mobile Service Binding Agent (MSBA)

(3) The multicast group management, which allows for multicast group member belonging to a same group to discover each other, fits nicely into the role of location management in a mobile environment. This means that while other proposals call for the addition of the location management function to be added to the overall system design, the proposed multicast service has already an efficient location management built in.

To understand how mobility can be achieved, let's use a simple example of setting up a point-to-point connection between two mobile terminals followed by a subsequent movement of one of them, which involves a handover. The reader should have no problem constructing a multicast example using the information provided here.

To begin, each MT will register (i.e., power-up, determine the strongest signal) via its mobile version of SBA (see Fig. 11, the mobile service binding agent (MSBA)) with its respective MSA by issuing a join (or create) operation to a logical group name, which also identifies itself uniquely. Each MSA will in turn distribute this information to the other MSAs in the network via the *group and location management agent* (details described in Section VII.B). Whenever a connection is to be made from a MT to another, the logical group name of the called party and the logical name of the calling party will be made known to their respective MSAs. Other information to be supplied to MSA via the MSBA include the current physical location of the MT and required QOS parameters. As soon as a suitable route is determined by the routing function, a physical VC will be setup to connect the two parties, and as soon as the connection is made, the respective MSBAs will be notified and data can flow. Let's further

assume that one of the two MTs has been moving and that a stronger signal "beacon" is detected by the wireless physical layer and triggers a handover. This handover information, together with the address of the new wireless access port (AP), will be made known to the MSBA of the MT in question via the wireless control channel (WCNTL). Once received, the MSBA will initiate an advance-join request on the new AP. The result of this is that a new multicast route will attempt to be set up from the crossover switch to the new AP. Assuming that the new VC can be set up, the requesting MSBA will be notified, which will in turn initiate a wireless handover via the MSBA WCNTL.

B. The Mobile Service Binding Agent (MSBA)

It should be clear from the above description that the only difference between a fixed host and a wireless mobile host in the proposed design is in the MSBA (Fig. 11). To begin, the MSBA will have an interface to the underlying radio frequency (RF) layers. In contrast to a fixed SBA, MSBA will perform all the function of an SBA i.e., in formulating group connection requests on behalf of the application and communicating with its MSA, but the MSBA will also include the following additional functionalities, which due to space constraints are explained briefly here.

(1) It will have an interface (via the WCNTL) to the underlying air interface so that it can be told to initiate a handover as and when the conditions are satisfied. Furthermore, it should also be able to inform the air interface the outcome of the higher-level handover and allow the air interface to take appropriate actions such as to do a switch over to the new AP. MSBA should also be notified when the wireless handover has been completed.

(2) Given that the VCI/VPI space is managed by the respective MSAs, this means that whenever a handover takes place the new VCs may be assigned VCI/VPI pairs different from the old ones. MSBA is therefore required to provide the necessary solutions to ensure that these newly assigned VCI/VPI pairs are used by the applications nonintrusively.

(3) Given that data loss could occur during the handover, MSBA is also responsible for providing the necessary solutions such as a data buffering scheme in order to maintain the connection QOS. Interested readers are referred to [36] for a description on such a scheme.

X. CONCLUSIONS AND FUTURE DIRECTIONS

The SingAREN network represents one typical next-generation broadband backbone infrastructure that will be adopted widely for day-to-day use. To meet the challenges of diversity of technologies in this network and the advanced applications it supports, this paper presents an integrated signaling design for the setting up of ATM connections as the basis of the SingAREN premium service. The proposed design supports and exploits some key ideas on multicast and programmability, which leads to many advantages over other solutions proposed. Some of these advantages include a common treatment of

both multipoints and one-to-one ATM connections, and seamless integration of mobile hosts. It is argued in various places of the paper that the proposed design can also support a more efficient yet simpler switch-to-switch routing protocol and connection setup. The proposed design also results in a new, and (arguably) more efficient, platform for the support of Internet protocols. Other less obvious advantages of the proposed design are:

- It encourages the use of "external" high-performance machines in performing route computation and processing connection requests. This can dramatically reduce the overall running cost by reducing the complexity of the network switch hardware and software.
- The proposed routing approach exchanges the link state information only among the MSAs, which effectively eliminate the need for all the switches in the network to flood their reachability and link state information. This greatly improves the overall routing scalability.

In the near future, the actual performance of the prototype under development will be addressed in greater detail. Other issues to be examined include simulating and optimizing the performance of the proposed group-management and routing scheme in multiple MSAs (Section VIII), the criteria of defining a MSA's domain boundaries, and the handover data buffering scheme of the MSBA in relation to higher level transport protocols [18]. International service trials of the proposed solution over a larger network are also planned.

ACKNOWLEDGMENTS

The authors acknowledge the generous funding support for SingAREN from the National Science and Technology Board (NSTB) and Telecommunication Authority of Singapore (TAS). Our special thanks to Dr. Tham Chen Khong, application manager of SingAREN, for providing the main text in Section II. Some content of this chapter first appeared in a paper [37] by the authors.

REFERENCES

1. The MBONE Information Page: http://www.mbone.com/.
2. Borden, M., *et al.* Integration of real-time Services in an IP-ATM network architecture. IETF RFC 1821, Aug. 1995.
3. Braden, R., *et al.* Integrated services in the Internet, architecture: An overview. RFC 1633, Sept 1994.
4. Laubach, M. Classical IP and ARP over ATM. IETF RFC 1577, Jan. 1994.
5. Armitage, G. J. IP multicasting over ATM networks. *IEEE J. Selected Areas Commun.* 15(3): 445–457, 1997.
6. Lazar, A. A., Bhonsle, S., and Lim, K. S. A binding architecture for multimedia networks. In *Proceedings of the COST-237 Conference on Multimedia Transport and Teleservices,* Vienna, Austria, Nov. 14–15, 1994.
7. Lazar, A. A., Lim, K. S., and Marconcini, F. Binding model: Motivation and description, CTR Technical Report 411-95-17. Centre for Telecommunications Research, Columbia University, New York, Oct. 1995.
8. Lazar, A. A., Lim, K. S., and Marconcini, F. Realizing a foundation for programmability of ATM networks with the binding architecture. *IEEE J. Selected Areas Commun.*

Special Issues on Distributed Multimedia Systems, 1214–1227, Sept. 1996. Also see http://comet.ctr.columbia.edu/xbind/.
9. Chan, M. C., Huard, J. F., Lazar, A. A., and Lim, K. Service creation, renegotiation and adaptive transport for multimedia networking. In *Proceedings of The Third COST 237 Workshop on Multimedia Telecommunications and Applications,* Barcelona Spain, November 25–27, 1996.
10. Ngoh, L. H., Li, H. Y., and Pung, H. K. A direct ATM multicast service with quality of service guarantees. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems,* June 1996, pp. 54–61.
11. Li, H. Y., Pung, H. K., & Ngoh, L. H. A QoS guaranteed ATM multicast service supporting selective multimedia data transmission. In *Proceedings of the International Conference on Computer Communications and Networks (IC3N),* Oct. 1996.
12. Zhang, L., *et al.* RSVP: A new resource reservation protocol. *IEEE Network* Sept. 1993. Available at http://www.isi.edu/div7/rsvp/rsvp.html.
13. Braden, R., and Hoffman, D. RAPI—RSVP application programming interface—Version 5. IETF Internet Draft, draft-ietf-rsvp-rapi-00.ps, May 1997.
14. Newman, P., Hinden, R., Liaw, F. C., Kyon, T., and Minshall, G. "General Switch Management Protocol Specification." Ipsilon Networks, Inc.
15. ATM Forum. ATM user–network interface specification version 4.0, af-sig-0061.000, July 1996.
16. ATM Forum. Private network-to-network interface specification 1.0 (PNNI 1.0), af-pnni-0055.000, Mar. 1996.
17. Caceres, R., and Iftode, L. Improving performance of reliable transport protocols in mobile computing environments. *IEEE J. Selected Areas Commun.* 13(5): 850–857, 1995.
18. Raychaudhuri D., and Wilson, N. D. ATM-based transport architecture for multiservices wireless personal communication networks. *IEEE. J. Selected Areas Commun.* 12(8): 1401–1413, 1994.
19. Murty, K. G. *Network Programming.* Prentice–Hall, Englewood Cliffs, NJ, 1992.
20. Alles, A. *ATM Internetworking,* white paper. Cisco Systems, Inc., Mar. 1995.
21. Shacham, N. Multipoint communication by hierarchically encoded data. In *IEEE Proceedings of INFOCOM '92,* 1992, pp. 2107–2114.
22. Semeria, C., and Maufer, T. Introduction to IP multicast routing. Internet Draft, 1996.
23. Buford, J. K. *Multimedia Systems.* Addison-Wesley, Reading, MA, 1994.
24. Widyono, R. The design and evaluation of routing algorithms for real-time channels. Tenet Group TR-94-024. University of California—Berkeley, 1994.
25. Yeadon, N., Garcia, F., Hutchison, D., and Shepherd, D. Continuous media filters for heterogeneous internetworking. In *IS & T/SPIE International Symposium on Electronic Imaging Special Session on Multimedia Computing and Networking,* San Jose, California, Jan. 1996.
26. Bell, T. E., Adam, J. A., and Lowe, S. J. Technology 1996—Communications. *IEEE Spectrum* 30–41, Jan 1996.
27. Cho, G., and Marshall, L. F. An efficient location and routing scheme for mobile computing environment. *IEEE J. Selected Areas Commun.* 13(5): 868–879, 1995.
28. Raychaudhuri, D. Wireless ATM networks: Architecture, system design and prototyping. *IEEE Personal Commun.* 42–29, Aug. 1996.
29. Ayanoglu, E., Eng, K. Y., and Karol, M. J. Wireless ATM: Limits, challenges, and proposals. *IEEE Personal Commun.* Aug. 1996.
30. ATM Forum. LAN emulation servers management specification 1.0, af-lane-0057.000, Mar. 1996.
31. Honcharenko, W., *et al.* Broadband wireless access. *IEEE Commun. Mag.* 20–26, Jan. 1997.
32. Ngoh L. H., *et al.* SingAREN—The Singapore advanced research and education network. *IEEE Commun. Mag.* 38(11): 74–82, 1998. Special issue on Advanced Telecommunication Infrastructures in Asia. See also http://www.singaren.net.sg.
33. Singapore ONE: http://www.s-one.gov.sg.
34. Deering, S., and Hinden, R. Internet protocol version 6 (IPv6) specification. RFC 1883, Dec. 1995.

35. Tennenhouse, D. L., and Wetherall, D. J. Towards an active network architecture. ACM *Comput. Commun. Rev.* 5–18, 1997.
36. Li, H. Y., et al. Support soft hand-over in wireless ATM networks using mobility enhanced multicast service. In *International Workshop on Wireless ATM,* China, Feb. 1997. Also available from the authors.
37. Ngoh, L. H., et al. An integrated multicast connection management solution for wired and wireless ATM networks. *IEEE Commun. Mag.* 35(11): 52–59, 1997.

INDEX