Lecture Notes in Computer Science 3181

Yahiko Kambayashi Mukesh Mohania
Wolfram Wöß (Eds.)

Data Warehousing and Knowledge Discovery

6th International Conference, DaWaK 2004
Zaragoza, Spain, September 1-3, 2004
Proceedings

Springer

Volume Editors

Yahiko Kambayashi
Kyoto University
Graduate School of Informatics
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
E-mail: kambayashi@db.soc.i.kyoto-u.ac.jp

Mukesh Mohania
IBM India Research Lab
Block No. 1, I.I.T., Hauz Khas, New Delhi 110016, India
E-mail: mkmukesh@in.ibm.com

Wolfram Wöß
University of Linz, FAW Institute for Applied Knowledge Processing
Altenberger Str. 69, 4040 Linz, Austria
E-mail: wwoess@faw.uni-linz.ac.at

Library of Congress Control Number: 2004110973

CR Subject Classification (1998): H.2, H.3, H.4, C.2, H.5, I.2

ISSN 0302-9743
ISBN 3-540-22937-X Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11312215 06/3142 5 4 3 2 1 0

In Memoriam

Yahiko Kambayashi

(1943–2004)

In great sadness we received notice of the sudden and untimely passing away of Prof. Yahiko Kambayashi on Friday, February 6, 2004.

Prof. Kambayashi, one of the leading international pioneers of database research, completed his Ph.D. at Kyoto University in 1970. In his early academic career, Prof. Kambayashi's research topics were logic circuits design, switching theory, and automata theory. From 1971 to 1973 he was a Visiting Research Associate at the University of Illinois, Urbana, where he developed a logic design method called the transduction method, which is now widely used by major US and Japanese logic design software vendors. After his return to Kyoto University as a faculty member in 1973, Prof. Kambayashi started to focus his research on databases. His major research results include an algorithm for the calculation of key dependencies, new inference rules for embedded multivalued dependencies, processing methods for cyclic queries, and new concurrency control mechanisms. In 1984, he became a professor at Kyushu University, where he extended his research area to geographic information systems, schema design in network data models, and concurrency control. In 1990, he was appointed as a professor at Kyoto University, where he conducted several very important practical research projects including studies of applications of database technologies to groupware and distance education systems. From April 2003, he served as Dean of the Graduate School of Informatics at Kyoto University.

Prof. Kambayashi published numerous articles in major journals and conferences. He also was the author and editor of many books and conference proceedings.

Prof. Kambayashi was also a great educator. A record number of Japanese and foreign students received M.S. and Ph.D. degrees under his supervision at Kyoto University and Kyushu University. Many of them are now serving as faculty members at universities in Japan and other countries. Prof. Kambayashi also taught courses at McGill University (1979), Kuwait University (1982) and Wuhan University (1984) as a visiting professor.

Prof. Kambayashi was an IEEE fellow, a trustee of the VLDB Endowment, a member of the SIGMOD Advisory Committee, a vice-chair of the ACM Tokyo/Japan Chapter, chair of the DASFAA Steering Committee, co-chair of the WISE Society and WISE Steering Committee, a member of the CODAS Steering Committee, a member of the ER Steering Committee, a member of the RIDE Steering Committee, a co-editor-in-chief of the World Wide Web Journal, an associate editor of ACM TODS, and a member of the editorial boards of several international journals. He was a winner of the ACM SIGMOD Contribution Award in 1995 for his many professional services in Japan and internationally.

Prof. Kambayashi helped to found the DEXA series of conferences and was one of the initiators of the DaWaK conference for which he served as General Chair from the very beginning in 1999.

Those who knew Prof. Kambayashi remember the energy and stamina with which he not only tackled his own research issues but also supported his colleagues, staff, students, collaborators and guests. Not only profound insights and expertise, but also his friendship and generous hospitality attracted many researchers and students.

Prof. Kambayashi is survived by his wife and two sons. His sudden leave is not only a tragic loss to his family but also a great loss to the whole international research community. Many of us will remember him as a friend, a mentor, a leader, an educator, and as our source of inspiration. We express our heartfelt condolence and our deepest sympathy to his family.

PC Chairs of DaWaK 2004
Mukesh Mohania (IBM India Research Lab, India)
Wolfram Wöß (FAW, Johannes Kepler University of Linz, Austria)

DEXA President and Vice-President
Roland Wagner (FAW, Johannes Kepler University of Linz, Austria)
A Min Tjoa (Vienna University of Technology, Austria)

Former PC chairs of DaWaK
Werner Winiwarter (University of Vienna, Austria)
Masatoshi Arikawa (University of Tokyo, Japan)
A Min Tjoa (Vienna University of Technology, Austria)

June 2004

Preface

Within the last few years, data warehousing and knowledge discovery technology has established itself as a key technology for enterprises that wish to improve the quality of the results obtained from data analysis, decision support, and the automatic extraction of knowledge from data.

The 6th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2004) continued a series of successful conferences dedicated to this topic. Its main objective was to bring together researchers and practitioners to discuss research issues and experience in developing and deploying data warehousing and knowledge discovery systems, applications, and solutions.

The conference focused on the logical and physical design of data warehousing and knowledge discovery systems. The scope of the papers covers the most recent and relevant topics in the areas of data cubes and queries, multidimensional data models, XML data mining, data semantics and clustering, association rules, data mining techniques, data analysis and discovery, query optimization, data cleansing, data warehouse design and maintenance, and applications. These proceedings contain the technical papers selected for presentation at the conference.

We received more than 100 papers, including 12 industrial papers, from over 33 countries, and the program committee finally selected 40 papers. The conference program included an invited talk by Kazuo Iwano, IBM Tokyo Research Lab, Japan.

We would like to thank the DEXA 2004 Workshop General Chairs (Prof. Roland Wagner, Prof. A Min Tjoa) and the Organizing Committee of the 15th International Conference on Database and Expert Systems Applications (DEXA 2004) for their support and their cooperation. Many thanks go to Ms. Gabriela Wagner for providing a great deal of assistance as well as to Mr. Raimund Angleitner-Flotzinger and Mr. Andreas Dreiling for administering the conference management software. We are very indebted to all the Program Committee members and external reviewers who reviewed the papers very carefully and in a timely manner. We would also like to thank all the authors who submitted their papers to DaWaK 2004; they provided us with an excellent technical program.

September 2004

Mukesh Mohania
Wolfram Wöß
Pankaj Garg

Program Committee

General Chairperson

Yahiko Kambayashi, Kyoto University, Japan

Conference Program Chairpersons

Mukesh Mohania, IBM India Research Lab, India
Wolfram Wöß, Johannes Kepler University of Linz, Austria
Industrial Chair: Pankaj Garg, Intel, Japan

Program Committee Members

Tatsuya Akutsu, Kyoto University, Japan
Marcelo Arenas, University of Toronto, Canada
Hiroki Arimura, Kyushu University, Japan
Mahmooh Awan, Dubai University College, Dubai
Mike Bain, University of New South Wales, Australia
Elena Baralis, Politecnico di Torino, Italy
Denilson Barbosa, University of Toronto, Canada
Ladjel Bellatreche, ENSMA, Poitiers University, France
Jorge Bernardino, Coimbra Polytechnic, Portugal
Sourav S. Bhowmick, Nanyang Technological University, Singapore
Christian Böhm, UMIT Tirol, Austria
Janez Brank, Jozef Stefan Institute, Ljubljana, Slovenia
Damien Bright, University of South Australia, Australia
Alun Butler, University of Greenwich, London, UK
Luca Cabibbo, Università degli Studi Roma Tre, Italy
Tiziana Catarci, Università degli Studi di Roma La Sapienza, Italy
Chee-Yong Chan, National University of Singapore, Singapore
Elizabeth Chang, Curtin University of Technology, Bentley, Perth, Australia
Arbee L.P. Chen, National Dong Hwa University, Taiwan
Sunil Choenni, Nyenrode University/University of Twente, The Netherlands
Saso Dzeroski, Jozef Stefan Institute, Slovenia
Vladimir Estivill-Castro, Griffith University, Brisbane, Australia
Wenfei Fan, Temple University, USA
Ling Feng, University of Twente, The Netherlands
Jean-Gabriel Ganascia, Université Paris VI, France
Minos Garofalakis, Bell Laboratories, USA
Ananth Grama, Purdue University, USA
Jerzy Grzymala-Busse, University of Kansas, USA

Rajeev Gupta, IBM India Research Lab, India
S.K. Gupta, IIT Delhi, India
Marc Gyssens, University of Limburg, Belgium
Mirsad Hadzikadic, University of North Carolina at Charlotte, USA
Tu Bao Ho, Japan Advanced Institute of Science and Technology, Japan
Se June Hong, IBM T.J. Watson Research Center, USA
Andreas Hotho, University of Karlsruhe, Germany
Samuel Kaski, University of Helsinki, Finland
Hiroyuki Kawano, Kyoto University, Japan
Larry Kerschberg, George Mason University, USA
Masaru Kitsuregawa, University of Tokyo, Japan
Guanling Lee, National Dong Hwa University, Taiwan
Yue-Shi Lee, Ming Chuan University, Taiwan
Leszek Lilien, Purdue University, USA
Tok Wang Ling, National University of Singapore, Singapore
Jixue Liu, University of South Australia, Australia
Zhang Long, IBM China Research Laboratory, India
Sanjay Kumar Madria, University of Missouri-Rolla, USA
Mukesh Mohania, IBM India Research Lab, India
Anirban Mondal, University of Tokyo, Japan
Wee Keong Ng, Nanyang Technological University, Singapore
Dimitris Papadias, Hong Kong University of Science and Technology, China
Stefano Paraboschi, Politecnico di Milano, Italy
Torben Bach Pedersen, Aalborg University, Denmark
Jian Pei, State University of New York, USA
Clara Pizzuti, ISI-CNR, Italy
David Powers, Flinders University of South Australia, Australia
Zbigniew W. Ras, University of North Carolina, USA
Rajeev Rastogi, Bell Laboratories, USA
Prasan Roy, IBM India Research Lab, India
Elke A. Rundensteiner, Worcester Polytecnic Institute, USA
Domenico Saccà, Università della Calabria, Italy
Carmem Satie Hara, Universidade Federal do Parana, Brazil
Michael Schrefl, Johannes Kepler University of Linz, Austria
Il-Yeol Song, Drexel University, USA
Bala Srinivasan, Monash University, Australia
Biplav Srivastava, IBM India Research Lab, India
Gerd Stumme, University of Karlsruhe, Germany
Einoshin Suzuki, Yokohama National University, Japan
Ernest Teniente, Universitat Politecnica de Catalunya, Spain
A Min Tjoa, Vienna University of Technology, Austria
Riccardo Torlone, Università Roma Tre, Italy
Sérgio Viademonte da Rosa, Monash University, Australia
Millist Vincent, University of South Australia, Australia
Werner Winiwarter, University of Vienna, Austria

Wolfram Wöß, Johannes Kepler University of Linz, Austria
Xindong Wu, University of Vermont, USA
Yiyu Yao, University of Regina, Canada
Show-Jane Yen, Fu Jen Catholic University, Taiwan
Kazumasa Yokota, Okayama Prefectural University, Japan

External Reviewers

Periklis Andritsos
Vasudha Bhatnagar
Mario Cannataro
Ling Chen
Silvia Chiusano
Byron Choi
Agnieszka Dardzinska
Filippo Furfaro
Paolo Garza
Martin Goller
John Horner
Arto Klami
Kuriakos Mouratidis
Dung Nguyen Xuan
Riccardo Ortale
Günter Preuner
Giovanni Quattrone
Janne Sinkkonen
Min Song
Jarkko Venna
Qiankun Zhao

Table of Contents

Data Warehousing Design

Knowledge Discovery Framework and XML Data Minig

Data Cubes and Queries

Multidimensional Schema and Data Aggregation

Inductive Databases and Temporal Rules

Industrial Track

Data Clustering

Data Visualization and Exploration

Data Classification, Extraction and Interpretation

Data Semantics

Association Rule Mining

Mining Event Sequences

Pattern Mining

Conceptual Design of XML Document Warehouses

Vicky Nassis[1], R. Rajugan[2], Tharam S. Dillon[2], and Wenny Rahayu[1]

[1] Dept. of CS-CE, La Trobe University, Melbourne, Australia
{vnassis,wenny}@cs.latrobe.edu.au
[2] Faculty of Information Technology, University of Technology, Sydney, Australia
{tharam3,rajugan}@it.uts.edu.au

Abstract. EXtensible Markup Language (XML) has emerged as the dominant standard in describing and exchanging data among heterogeneous data sources. XML with its self-describing hierarchical structure and its associated XML Schema (XSD) provides the flexibility and the manipulative power needed to accommodate complex, disconnected, heterogeneous data. The issue of large volume of data appearing deserves investigating XML Document Warehouses. But due to XML's non-scalar, set-based semi-structured nature, traditional data design models lack the ability to represent XML design level constructs in an abstract and implementation-independent form which are crucial for designing complex domains such as data marts and data warehouses, but also during their operational and maintenance phase. We utilize Object Oriented (OO) concepts to develop a conceptual model for XML Document Warehouses. In this paper we propose a conceptual design formalism to build meaningful XML Document Warehouses (XDW). Our focus includes; (1) conceptually design and build meaningful XML (warehouse) repository (xFACT) using OO concepts in integration with XML Schema constructs, (2) conceptually model and design *virtual dimensions* using *XML conceptual views* [10a] [10b] to satisfy warehouse end-user requirements and (3) use UML *package* diagrams to help logically group and build hierarchical *conceptual views* to enhance semantics and expressiveness of the XDW.

1 Introduction

Data Warehousing (DW) has been an approach adopted for handling large volumes of historical data for detailed analysis and management support. Transactional data in different databases is cleaned, aligned and combined to produce good data warehouses. Since its introduction in 1996, eXtensible Markup Language (XML) has become the *defacto* standard for storing and manipulating self-describing information (meta-data), which creates vocabularies in assisting information exchange between heterogenous data sources over the web [22]. Due to this, there is considerable work to be achieved in order to allow electronic document handling, electronic storage, retrieval and exchange. It is envisaged that XML will also be used for logically encoding documents for many domains. Hence it is likely that a large number of XML documents will populate the would-be repository and several disparate transactional databases.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 1–14, 2004.

The concern of managing large amounts of XML document data arises the need to explore the data warehouse approach through the use of XML document marts and XML document warehouses.

Since the introduction of dimensional modeling which, evolves around facts and dimensions, several design techniques have been proposed to capture multidimensional data (MD) at the conceptual level. Ralph Kimball's Star Schema [11] proved most popular, from which well-known conceptual models SnowFlake and StarFlake were derived. More recent comprehensive data warehouse design models are built using Object-Oriented concepts (structural relationships, Object cubes or data cubes) on the foundation of Star Schema. In [7] [8a] [8b] and [18] two different OO modeling approaches are demonstrated where a data cube is transformed into an OO model integrating class hierarchies. The Object-Relational Star schema (O-R Star) model [9] aims to envisage data models and their object features, focusing on hierarchical dimension presentation, differentiation and their various sorts of embedded hierarchies.

These models both object and relational have a number of drawbacks namely they are; (a) data-oriented without sufficient emphasis or capturing user requirements, (b) extensions of semantically poor relational models (star, snowflake models), (c) original conceptual semantics are lost before building data warehouses as the operational data source is relational, (d) further loss of semantics resulting from oversimplified dimensional modeling, (e) time consuming if additional data semantics are required to satisfy evolving user requirements and (f) complex query design and processing is needed, therefore maintenance is troublesome. When applying these approaches to the design of XML document warehouses, it is important to consider XML's non-scalar, set-based and semi-structured nature. Traditional design models lack the ability to utilise or represent XML design level constructs in a well-defined abstract and implementation-independent form.

One of the early XML data warehouse implementation includes the Xyleme Project [16]. The Xyleme project was successful and it was made into a commercial product in 2002. It has well defined implementation architecture and proven techniques to collect and archive web XML documents into an XML warehouse for further analysis. Another approach by Fankhauser et al. [5] explores some of the changes and challenges of a document centric (such as EDI messages, eBooks) XML warehouse. Both these approaches offer some powerful implementation and architectural design insights into XML warehouses but, coupling them with a well defined conceptual and logical design methodology may help future design of such XML warehouse for large-scale XML systems. For these given reasons, in this paper we propose a conceptual modelling approach to the development of an XML Document Warehouse (XDW).

1.1 Our Work

UML, a widely adopted standard for Object-Oriented (OO) conceptual models is the foundation to build this conceptual model for XML document warehousing, considering that its graphical notation complies with the user(s) and domain expert(s) understanding. Our proposed methodology provides; (a) a conceptual design, using UML, for the XDW meaningful XML FACT repository (xFACT), (b) conceptually model and design *virtual dimensions* using *XML conceptual views* [10a] [10b] to satisfy warehouse end-user requirements and (c) use UML *package* diagrams to logically group and build hierarchical *conceptual views* to enhance semantics and expressiveness of the XDW.

The main aspects to our methodology are as follows; (1) *User requirements (UR):* Assist in determining different perspectives of the document warehouse rather than data/documents, (2) *XML Document structure* [13]: Using XML document capability in accommodating and explicitly describing heterogeneous data accompanied with their inter-relationships semantics (unlike flat-relational data), (3) *XML Schema:* Describes, validates and provides semantics for its corresponding instance document (XML document) [15]. Also, it can capture all OO concepts and relationships [4] [6] as well as intuitive XML specific constructs, such as ordering explained in Section 1.2 and (4) *Conceptual Views* [10a][10b]: A *conceptual view* describes how a collection of XML tags relates to the direct use of a domain user at the conceptual/abstract level. A typical XML domain may contain XML documents ranging from few to many thousands of semantically related clusters of XML documents depending on the real world requirement. Only a subset of the XML tags cluster, its specification and their data values (information), which collectively form a conceptual view, may be of interest to the domain user at a point in time.

The rest of the paper is organized as follows; In Section 1.2, we outline some unique XML Schema concepts that are captured and modeled using UML, while Section 1.3 provides a brief description of the example case study used in this paper, of which, we will be gradually building XML warehouse model (using UML). Section 2 is dedicated to in-depth discussion of the XDW conceptual model (with examples), followed by the conclusion in Section 3.

1.2 XML Schema and OO Concepts

A widely used approach for conceptual modeling to developing XML Schemas is the OO conceptual model, frequently expressed in UML [6]. In order to understand the semantics that must be captured in abstract models of XML Data Warehouses, two significant issues are raised (1) ordered composition and (2) homogeneous composition. An illustration of each case is provided in the sections that follow using the proposed mapping techniques in [6] for the transformation of the OO diagrams to XML Schema. The examples are extracted from the complete UML diagram in Figure 5.

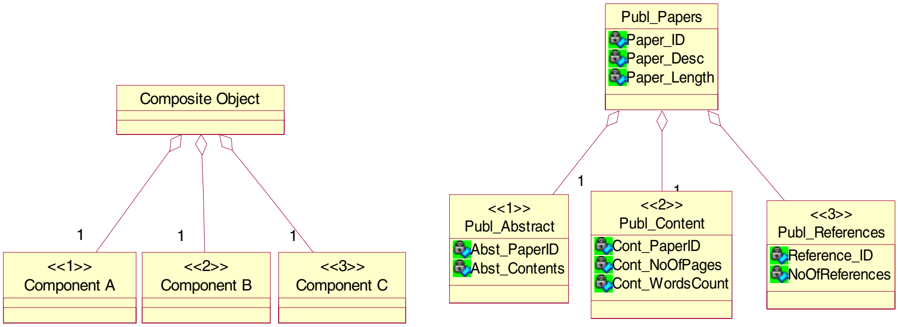
1.2.1 Ordered Composition

Consider the following XML Schema fragment:

```
<xs:element name="Publ_Papers" type=" Publ_PaperType"/>
<xs:complexType name="Publ_PaperType">
          <xs:sequence>
               <xs:element name="Paper_ID" type="xs:ID"/>
               <xs:element name="Paper_Desc" type="xs:string"/>
               <xs:element name="Paper_Length" type="xs:short"/>
               <xs:element name="Publ_Abstract" type="Publ_AbstractType"/>
```

```
            <xs:element name="Publ_Content" type="Publ_ContentType"/>
            <xs:element name="Publ_Reference" type="Publ_ReferenceType"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_AbstractType">
        <xs:sequence>
            <xs:element name="Abst_PaperID" type="xs:IDREF"/>
            <xs:element name="Abst_Contents" type="xs:string”/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_ContentType">
        <xs:sequence>
            <xs:element name="Cont_PaperID" type="xs:IDREF"/>
            <xs:element name="Cont_NoOfPages" type="xs:short"/>
            <xs:element name="Cont_WordsCount" type="xs:short"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_ReferenceType">
        <xs:sequence>
            <xs:element name="reference_ID" type="xs:ID" nillable="false"/>
            <xs:element name="NoOfReferences" type="xs:string" nillable="false"/>
        </xs:sequence>
</xs:complexType>
```

The composite element `Publ_Papers` is an aggregation of the sub-elements namely `Publ_Title, Publ_Abstract` and `Publ_Content`. Interpreting this XML Schema section we observe that the tag `<xs:sequence>` signifies that the embedded elements are not only a simple assortment of components but these have a specific ordering. Hence we add to UML an annotation that allows capturing of the ordered composition as shown in Figure 1, utilizing stereotypes to specify the objects' order of occurrence such as <<1>>, <<2>>, <<3>>, …. ,<<n>>. Figure 1b shows the XML Schema segment above modeled applying this UML notation.



a. UML Stereotype for an Ordered Composition

b. The XML Schema Fragment for an Ordered Composition shown in UML

Fig. 1. Figures 1(a) and 1(b): Ordered composition example

1.2.2 Homogeneous Composition

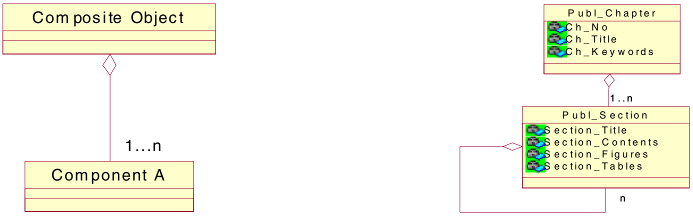
In a homogeneous aggregation, one "whole" object consists of "part" objects, which are of the same type [19]. Two important cases are considered when applied to the homogenous composition, which are as follows:

1.2.2.1 Case of One-to-Many Relationship

Consider the XML schema segment below of the relationship between the two elements `Publ_Chapter` and `Publ_Section`.

```
<xs:element name="Publ_Chapter" type="Publ_ChapterType"/>
<xs:complexType name=" Publ_ChapterType">
        <xs:sequence>
            <xs:element name="Ch_No" type="xs:short"/>
            <xs:element name="Ch_Title" type="xs:string"/>
            <xs:element name="Ch_Keywords" type="xs:string" maxOccurs="unbounded"/>
            <xs:element name="Publ_Section" type="Publ_SectionType" minOccurs="1"
            maxOcurs ="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_SectionType">
        <xs:sequence>
            <xs:element name="Section_Title" type="xs:string"/>
            <xs:element name="Section_Contents" type="xs:byte"/>
            <xs:element name="Section_Figures" type="xs:anyType"/>
            <xs:element name="Section_Tables" type="xs:byte"/>
            <xs:element name="Section_SubSection" type="Publ_SectionType" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
```

It is declared that the object `Publ_Chapter` can consist of one or more `Publ_Sections`. We specify the **maxOccurs** value of `Publ_Section` to "unbounded" (default value is '1') enabling the element `Publ_Section` to occur from one to many times within `Publ_Chapter`. We consider the assumption that a given section **cannot** be contained in any other chapter from the papers submitted. This characteristic is shown in Figure 2a using the proposed UML notation while Figure 2b shows the model corresponding to the XML Schema fragment below.



a. UML Notation for a Homogeneous Composition featuring One to Many relationship

b. Homogeneous Composition example

Fig. 2. Figures 2(a) and 2(b): Homogeneous composition (Case 1)

1.2.2.2. Case of Many-to- Many Relationships

Consider the XML schema segment below of the relationship between `Publ_Year` and `Publ_Month` elements:

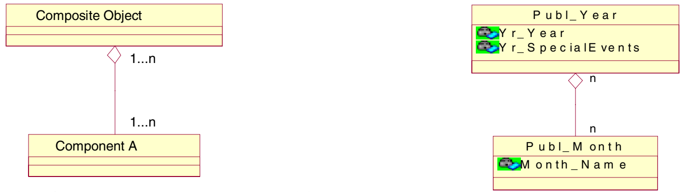
```
<xs:element name="Publ_Year" type="Publ_YearType" maxOccurs="unbounded"/>
<xs:complexType name="Publ_YearType">
        <xs:sequence>
            <xs:element name="Yr_Year"/>
```

```
                <xs:element name="Yr_SpecialEvents"/>
                <xs:element name="Publ_Month" type="Publ_MonthType" nillable="false"
                maxOccurs="unbounded"/>
            </xs:sequence>
</xs:complexType>
<xs:complexType name="Publ_MonthType">
            <xs:sequence>
                <xs:element name="Month_Name"/>
            </xs:sequence>
</xs:complexType>
```

In this category a composite object (`Publ_Year`) may have many components (`Publ_Months`) and each component may belong to many composite objects. The **maxOccurs** value equaling to “unbounded” in both elements forms the many-to-many relationship between `Publ_Year` and `Publ_Month` elements. The UML notation illustrating this case is shown in Figure 3a while Figure 3b shows the model corresponding to the XML Schema fragment below.

1.3 An Example Case Study

As a motivating example for this paper, we intent to gradually build a conceptual model of an XDW based on a simplified (academic) conference publication system (CPSys). To illustrate the purpose of this paper we highlight only a few, simplified warehouse user requirements. These requirements include (a) chronological listing of author details, sorted by conferences, (b) alphabetical listing of all proceedings arranged according to the conference date and year. Each listing should include proceedings title, ISBN, conference, editor listing and the table of contents, (c) alphabetical listing of all proceedings together with abstract lists with cross-reference to author/(s) details, (d) chronological listing of conferences and workshops listed with their associated publications, and (e) chronological listing of conferences and workshops listed with their associated abstract list, together with their authors.



a. UML Notation for a Homogeneous Composition featuring Many to Many relationship

b. Homogeneous Composition example (n:m)

Fig. 3. Figures 3(a) and 3(b): Homogeneous composition (Case 2)

Due to page limitations, in this paper, we provide only a very brief discussion on capturing and modeling user requirements, as it can be found in many published literatures in this area such as [21]. In the next section, we outline our proposed XDW conceptual with embedded examples to highlight some important model concepts.

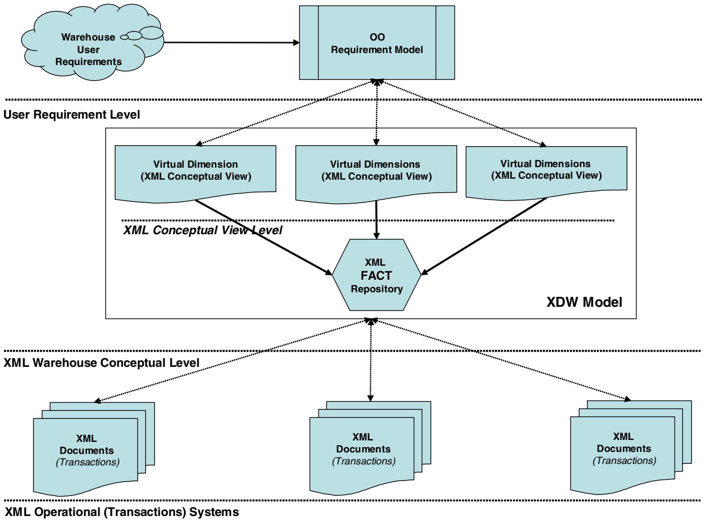

Warehouse User Requirements
OO Requirement Model
User Requirement Level
Virtual Dimension (XML Conceptual View)
Virtual Dimensions (XML Conceptual View)
Virtual Dimensions (XML Conceptual View)
XML Conceptual View Level
XML FACT Repository
XDW Model
XML Warehouse Conceptual Level
XML Documents (Transactions)
XML Documents (Transactions)
XML Documents (Transactions)
XML Operational (Transactions) Systems

Fig. 4. XDW Context Diagram

2 XML Document Warehouse Model

The XDW model outlined below, to our knowledge, is unique in its kind as it is utilizes XML itself (together with XML Schema) to provide; (a) structural constructs, (b) metadata, (c) validity and (d) expressiveness (via refined granularity and class decompositions). The proposed model is composed of three levels (1) User Requirement Level, (2) XML Warehouse Conceptual Level and (3) XML Schema Level. A context diagram of this model is given in Figure 4, below. The first level, the user requirement level composes of two elements; (a) warehouse user requirement document and (b) OO requirement model which includes UML use-case diagrams and descriptions. The second level of the XDW model is composed of the XML FACT repository (xFACT, discussed in Section 2.4 below) and the dimensions that satisfy warehouse user requirements captured. The (XML) Schema level of the model involves transformation of the conceptual model into XML Schema.

2.2 User Requirement Level

The first level of the XDW model captures the warehouse end-user requirements. As opposed to the classical data warehouse models, this requirement model does not consider the transactional data as the focal point in the design of the warehouse. The XDW model is designed based on the user requirements. The transactional data/documents are considered only to refine existing user requirements, if such a need arises. This level is further divided into two more sub-components, which are discussed below.

2.2.1 Warehouse User Requirements

The warehouse user requirements correspond to written, non-technical outline of the XML warehouse. These are usually the typical or the predictable results expected of an XDW. Also, these user requirements can be further refined and/or enhanced by the participation of domain experts or the operators of the transactional system in question. In addition, further refinement can be added to this model by using approaches that generate user requirement documents such as analysing frequent user query patterns [21] or other automated approaches.

2.2.2 OO Requirement Model

The OO requirement model transforms all non-technical user requirements into technical, software model specific concepts using UML (actors, use-case, and objects) to represent. Again, the OO requirement model can be further refined through the involvement of domain experts, system analysts, programmers or the operators of the transactional system.

2.3 XML Warehouse Conceptual Level in UML

The process of deriving the XDW conceptual model involves taking the formalized user requirements expressed in UML and then validating them against the XML transactional system to check for data availability. In the case of no or lack of transactional data highlights ambiguous the user requirements (which has to be re-defined and/or modified) for future warehouse requirements. At a later stage during the transactional system maintenance, these will then be considered as the future user requirements.

The XDW conceptual model is composed of; (1) an XML FACT repository (xFACT) and (2) a collection of logically grouped ***conceptual views***. The xFACT is a snapshot of the underlying transactional system/(s) for a given *context*. A *context* is more than a measure [1] [7] or an item that is of interest for the organization as a whole. In classical data warehouse models, a context is normally modeled as an ID packed FACT and associated data perspectives as dimensions. Usually, due to constraints of the relational model, a FACT will be collapsed to a single table, with IDs of its dimension/(s), thus emulating (with combination of one or more dimension/(s)) a data cube (or dimensional data). A complex set of queries is needed to extract information from the FACT-Dimension model. But, in regards to XML, a context is more than a flattened FACT (or simply referred to as *meaningless* FACT) with embedded semantics such as those explained Section 1.2 as well as with non-relational constructs such as set, list, and bag. Therefore, we argue that, a *meaningless* FACT does not provide semantic constructs that are needed to accommodate an XML *context*.

The role of conceptual views is to provide perspectives to the document hierarchy stored in xFACT repository. Since conceptual views can be grouped into logical groups, each group is very similar to that of a ***subject area*** (or class categories) [2] [20] in Object-Oriented conceptual modeling techniques. Each subject-area in the

XDW model is referred to as *Virtual Dimension* (VDim) to keep in accordance with dimensional models. VDim is called *virtual*; since it is modeled using XML *conceptual views* [10a] (which is an *imaginary* XML document) and behaves as a dimension in the given perspective. The following sections discuss in detail the modeling of VDims, xFACT repository and the issues associated with them.

2.4 XML FACT Repository (xFACT)

XML FACT repository (xFACT) is composed of an XML Schema, which is constructed from the XDW conceptual model and its XML document (formed from the transactional XML document sources). It is a snapshot of the underlying transactional system/(s) for a given *context*. A context is a meaningful collection of classes and relationships, which is designed to provide perspectives (with the help of one or more *conceptual views*) for business intelligence. Due to page limitation, we only presented some fragments (*see* Section 1.2) of the xFACT (XML) Schema, which provides the metadata and validity for the xFACT repository.

2.4.1 Meaningful FACT

In building the xFACT, we vary from modeling traditional data warehouse flat FACT tables such that, we add meaning. That is to say, the xFACT is more semantically descriptive due to its interconnected relationships and decomposed hierarchical structures (Figure 5). The relationships considered in xFACT are not restricted to association (1:1, 1:m, n:m) relationships, but also homogenous composition (with shared aggregation with cardinality 1:n and n:m) specialization/generalization and ordering as discussed in Section 1.2. We utilize XML schema definition language to describe the xFACT semantics (classes, relationships, ordering and constraints). Therefore, modeling of the xFACT is constrained only by the availability of XML schema elements and constructs.

Figure 5 shows the complete model of the xFACT designed for our case study, where the real-world object `Publications` is hierarchically de-composed into `Publ_Papers` and `Publ_Conference`. `Publ_Papers` is further decomposed (with ordering) into `Publ_Abstract`, `Publ_Contents` and `Publ_References`. Such decompositions are necessary to provide granularity to a real-world object and if needed, additional semantics can be added at the design time at different levels of hierarchy. For example, the `Publ_Conference` class hierarchy is decomposed with additional semantics such as `Publ_Region`, `Publ_Country` and `Publ_City`. Another example is the relationship between the parent class `Publ_Person` and the derived sub-classes `Publ_Referee` and `Publ_Author`, which form a generalization relationship (ISA).

Since the flexibility of XML Schema in handling complex structures is greater than any other model, it provides extra intuition to make the xFACT table as expressive as possible. In addition, the process of iterative refinement of user requirements conducted at the design stage helps in modeling such a FACT. Building the xFACT provides the well-defined repository for designing/constructing of expressive VDims.

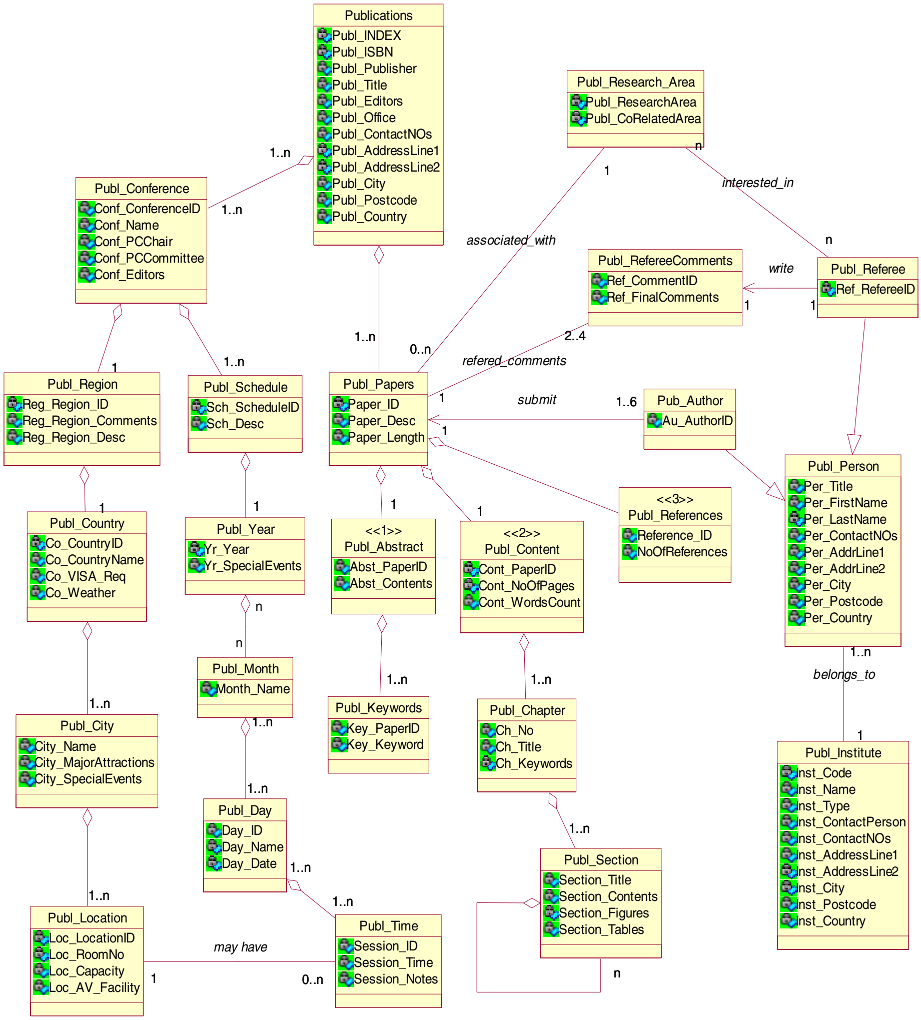


Fig. 5. The complete xFACT of the example case study

2.5 Virtual Dimensions

A user requirement, which is captured in the OO Requirement Model is transformed into one or more *Conceptual Views* [10a] in association with the xFACT. These are typically *aggregate views* or *perspectives* of the underlying stored documents of the transaction system/(s). In addition, if required, further constructs (such as computed attributes or joint values) can be added to the conceptual view/(s) to satisfy the underlying user requirement. A valid user requirement is such that, it can be satisfied by one or more XML conceptual views for a given context (i.e. xFACT). But a typical situation may arise where, a user requirement exists for which there is no transactional document or data fragment exists to satisfy it. If this does occur, further enhancements are made to the user requirement to make it feasible to model with the existing xFACT. Such situation indicates that there is no transactional document or document fragment to satisfy the end user requisite. Therefore modeling of VDim is an iterative process where user requirements are validated against the xFACT in conjunction with the transactional system.

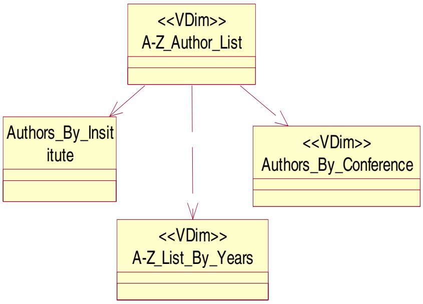


Fig. 6. A conceptual view hierarchy with <<construct>> stereotype

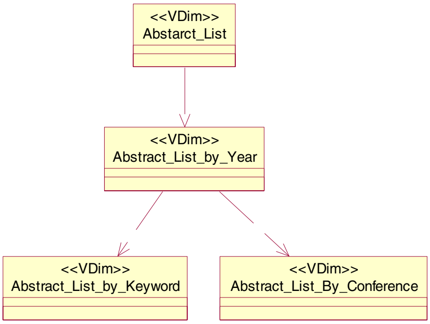


Fig. 7. VDim "*Abstract*" Package Contents

A VDim is modeled at the XDW conceptual level using a new UML stereotype called `<<VDim>>`. This stereotype is similar to a UML class notation with a defined set of attribute and methods. The method set can be either constructors (to construct a VDim) or manipulators (to manipulate the VDim attribute set). Also note that we model the relationship between an xFACT and a VDim with a dashed, directed line, denoting the `<<construct>>` stereotype (shown in Figure 6-7). Though VDims can have additional semantic relationships such as generalization, aggregation, association [10a] [10b], such relationships can be shown using standard UML notations. In addition to this, two VDims can also have `<<construct>>` relationships depending on dependencies (e.g. Figure 7, between VDim `Abstract_List_by_Year` and VDim `Abstract_List_by_Keyword`).

Definition 1: *One Virtual Dimension composes of one (or more logically grouped) conceptual view, thus satisfying one (or more logically related) user document warehouse requirement/(s).*

2.5.1 UML Packages as Dimensions

Earlier, we stated that semantically related conceptual views could be logically grouped together as grouping classes into a subject area. Further, a new view-hierarchy and/or constructs can be added to include additional semantics for a given user requirement. In the XDW conceptual model, when a collection of similar or related *conceptual views* are logically grouped together, we called it aggregated or grouped *Virtual Dimension* (Figures 6-7), implying that it satisfies one or more logically related user requirement/(s). In addition, we can also construct additional conceptual view hierarchies such as shown in Figures 6-7. These hierarchies may form additional structural or dependency relationships with existing conceptual views or view hierarchies (grouped VDims) as shown in Figures 8-9. Thus it is possible that a cluster of dimensional hierarchy/(ies) can be used to model a certain set of user requirement/(s). Therefore we argue that, this aggregate aspect can give us enough abstraction and flexibility to design a user-centered XDW model.

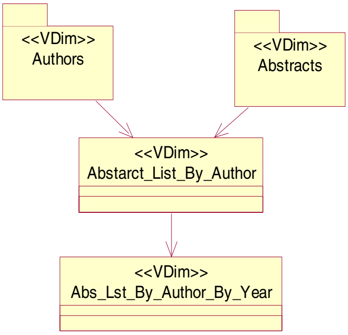


Fig. 8. A VDim hierarchy (derived from grouped VDims)

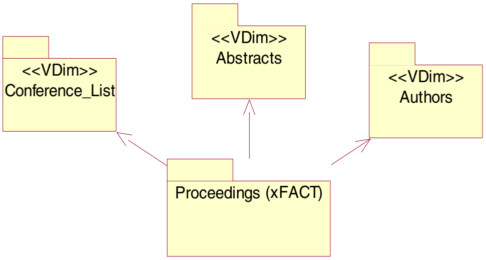


Fig. 9. "Authors", "Abstracts" & "Conference_List" package

In order to model an XML conceptual view hierarchy or VDim and capture the logical grouping in among them, we utilize the ***package*** construct in UML. According to OMG specification;

"*A package is a grouping of model elements. Packages themselves may be nested within other packages. A package may contain subordinate packages as well as other kinds of model elements. All kinds of UML model elements can be organized into packages.*" [12].

This in practice describes our logical grouping of XML conceptual views and their hierarchies. Thus we utilize packages to model our connected dimensions (Figure 8). Following the similar arguments above, we can show that, the xFACT (shown in Figure 5) can be grouped into one logical construct and can be shown in UML as one package. In Figures 9 & 10, we show our case study XDW model with xFACT and VDims connected via `<<construct>>` stereotype.

In summary, we gradually introduced the XDW conceptual model in UML and fragments from sections 2.3 – 2.5. First the complete xFACT model is shown in Figure 5 while the xFACT stereotype of ordering is shown in Figure 1. The xFACT semantic relationships are shown in Figures 2, 3 & 4. Then we introduced virtual dimensions (VDim, shown in figures 7,8 & 9) with the `<<construct>>` stereotype. Later the complete XDW model (similar to that of the Star Schema for relational model) is shown in figures 9, & 10 using the UML *package* construct. Please note that, due to space limitations, the attributes set of the VDim and that of some xFACT classes are not shown. Also, in the xFACT class hierarchy, the cardinality of the composition and the association relationships should be treated as 1, unless specified.

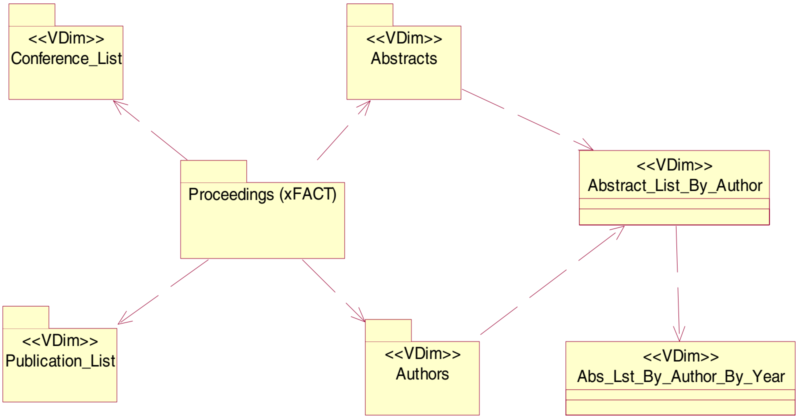


Fig. 10. XDW Conceptual Model (in UML)

3 Conclusion and Future Work

XML has become an increasingly important data format for storing structured and semi-structured text intended for dissemination and ultimate publication in a variety of media. It is a markup language that supports user-defined tags and encourages the separation of document content from presentation. In this article, we present a coherent way to integrate a conceptual design methodology to build a native XML document warehouse. The proposed XML design methodology consists of user requirement level, warehouse conceptual level and XML Schema level, which capture warehouse user requirements and the warehouse model respectively. The only model we did not discuss here in detail is the OO user requirement model, as many literatures exist for that purpose.

For future work, a lot of issues deserve investigation. The first subject matter is the mapping of the conceptual model in XML Schema level. This includes converting the xFACT and VDim into XML Schema using the transformations discussed in the papers [6] and [23]. In the case of VDims, this is actually the transformation of between the *conceptual views* into *XML View* [10a] [10b] schemas. Other steps in our future work include; (a) the techniques to map operational XML documents to xFACT (the equivalent of the classical data warehouse Extract-Load-Transform process), (b) access the data warehouse including XML queries for OLAP processing and (c) aggregate functions.

References

1. M Golfarelli, D Maio, S Rizzi; *"The Dimensional Fact Model: A Conceptual Model for Data Warehouses"*; Int. Journal of Cooperative Information Systems (Invited Proceeding), 7-2-3, 1998.
2. T S Dillon, P.L Tan; "*Object-Oriented Conceptual Modeling*"; Prentice Hall, Australia, 1993.

3. Ramez Elmasri, Shamkant B Navathe; "*Fundamentals of Database Systems*";3rd Ed, Addison-Wesley '00.
4. Ling Feng, Tharam Dillon & E Chang; "*A Semantic Network Based Design Methodology for XML Documents*"; ACM Trans. on Info. Sys., Volume 20, No. 4, August 2002; pp 390-421.
5. Peter Fankhauser, Thomas Klement; "*XML for Data Warehousing Changes & Challenges*"; Proc of DaWaK 2003 Prague, Sept, pp 1-3.
6. Ling Feng, Elizabeth Chang, Tharam Dillon; "*Schemata Transformation of Object-Oriented Conceptual Models to XML*"; Int. Jou. of Com. Sys Sci & Eng., Vol 18 No. 1 Jan 2003. pp. 45-60.
7. Juan Trujillo, Manuel Palomar, Jaime Gomez, Il-Yeol Song; "*Designing Data Warehouses with OO Conceptual Models*"; "Computer", IEEE Computer Society (12): 66-75. December 2001.

8a. Sergio Lujan-Mora, Juan Trujillo, Il-Yeol Song; "*Multidimensional Modeling with UML Package Diagrams*"; ER 2002: pp 199-213.

8b. Sergio Lujan-Mora, Juan Trujillo, Il-Yeol Song; "*Extending the UML for Multidimensional Modeling*"; UML 2002: pp 290-304.

9. Rahayu, J.W., Dillon, T.S., Mohammad, S., and Taniar, D., "*Object-Relational Star Schemas*", Proc. of the 13th IASTED Int. Conf. Parallel and Distributed Comp. & Sys. (PDCS 2001), 2001.

10a. R Rajugan, E Chang Thram S Dillon & Ling Feng; "*XML Views: Part I*"; 14th Int. Conf. on DB & Expert Systems Applications (DEXA 2003), Prague, Czech Republic, September 1-5, 2003.

10b. R Rajugan, E Chang Thram S Dillon & Ling Feng; "*XML Views, Part II: Modeling Conceptual Views Using XMSemantic Nets*"; (*to be submitted*).

11. Ralph Kimbal, Margy Ross; "*The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*"; 2nd Edition, Wiley Computer Publishing, 2002.
12. OMG; "*Unified Modeling Language Specification*", March 2003, Ver 1.5 formal/03-03-01; http://www.omg.org/technology/documents/formal/uml.htm
13. W3C Consortium; "*EXtensible Markup Languagee*s"; http://www.w3.org/XML/; 2000.
14. W3C Consortium; "*XML Query (XQuery) Requirements*"; http://www.w3.org/TR/xquery-requirements/ ; Nov 2003.
15. W3C Consortium; "*XML Schema*"; http://www.w3c.org/XML/Schema; 2 May 2001.
16. Xyleme Project; Publications; http://www.xyleme.com/
17. OMG; "Data Warehousing, CWM™ and MOF™ Resource Page" http://www.omg.org/cwm/
18. Alberto Abelló, Josè Samos, and Fèlix Saltor "*Understanding Facts in a Multidimensional Object-Oriented Model*"; 4th Int. Wrks.on DW and OLAP (DOLAP '01) Atlanta, Nov '01, pp 32-39.
19. Rahayu, W. S., E. S. Chang, et al.; *"Aggregation versus Association in Object Modeling and Databases"*; ACS Conf. on IS, Australian Computer Society Inc. 2: pp 521 - 532.
20. Coad P., Yourdon, E. "*Object-oriented Analysis*"; Prentice Hall, 1991.
21. Ji Zhang, Tok Wang Ling, Robert M. Bruckner & A Min Tjoa; "*Building XML Data Warehouse Based on Frequent Patterns in User Queries*"; Proc of DaWaK 2003 Prague, Sept. '03; pp 99-108.
22. Jaroslav Pokorny; *"XML Data Warehouse: Modelling and Querying";* Kluwer Academic Publishers, The Netherlands, 2002, pp 67-80.
23. Renguo Xiaou, Tharam S. Dillon, Elizabeth Chang, Ling Feng; "*Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema*"; DEXA 2001, pp 795-804.

Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses

Ladjel Bellatreche[1], Michel Schneider[2], Hervé Lorinquer[2], and Mukesh Mohania[3]

[1] LISI/ENSMA, Futuroscope, France
bellatreche@ensma.fr
[2] LIMOS, Blaise Pascal University, France
michel.schneider@isima.fr
[3] I.B.M. India Research Lab, India
mkmukesh@in.ibm.com

Abstract. There has been a lot of work to optimize the performance of relational data warehouses. Three major techniques can be used for this objective: enhanced index schemes (join indexes, bitmap indexes), materialized views, and data partitioning. The existing research prototypes or products use materialized views alone or indexes alone or combination of them, but none of the prototypes use all three techniques together for optimizing the performance of the relational data warehouses. In this paper we show by a *systematic experiment evaluation* that the combination of these three techniques reduces the query processing cost and the maintenance overhead significantly. We conduct several experiments and analyse the situations where the data partitioning gives better performance than the materialized views and indexes. Based on rigorous experiments, we recommend the tuning parameters for better utilization of data partitioning, join indexes and materialized views to optimize the total cost.

1 Introduction

Data warehousing technology uses the relational data schema for modeling the underlying data in a warehouse. The warehouse data can be modeled either using the star schema or the snowflake schema. In this context, OLAP queries require extensive join operations between the fact table and dimension tables [11,15]. To improve the query performance, several optimization techniques were proposed; we can cite materialized views [1,2,3,10,16], advanced indexing techniques including bitmapped indexes, join indexes (for supporting star queries), bit-sliced indexes, projection indexes [7, 8, 9, 14, 16] and data partitioning [4, 5, 12]. The data table can be fragmented into three ways: vertically, horizontally or mixed. In the context of relational warehouses, the previous studies show that horizontal partitioning is more suitable. Commercial RDBMSs like Oracle9i offer various options to use horizontal partitioning: Range, Hash, and Hybrid. This type of horizontal partitioning is called *primary horizontal partitioning* and it can be applied to dimension tables. Another type of horizontal partitioning is called derived horizontal partitioning [4]. It consists in decomposing a table based on the fragmentation schema of another table. For example, let us consider a star schema with three

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 15–25, 2004.

dimension tables (Customer, Time and Product) and a fact table Sales. The former table can be decomposed into two fact fragments Sales_1 and Sales_2 that represent all sales activities for only the male customers and all sales activities for only the female customers, respectively. This means that the dimension table Customer is virtually partitioned using range partitioning on Gender column.

In this paper, we conduct experiments to show the effect of the combination of the three major optimization techniques by using the APB1 benchmark [13] under Oracle 9i. Along this study, the effect of updates (append and delete operations) is considered. Since the derived horizontal partitioning is not directly supported by commercial RDBMSs like Oracle9i, we present an implementation to make it operational (in this paper, we use fragmentation and partitioning interchangeably).

The paper is organized as follows: in Section 2, we introduce the necessary background and we present our explicit solution to implement the derived horizontal partitioning in commercial systems; in Section 3, we present experiments for comparing joins indexes and partitioning and we summarize the main tuning recommendations when using the three techniques; in Section 4 we present experiments for exploring the combination of the three techniques; Section 5 concludes and points some perspective.

2 Background

2.1 Benchmark

For our study, we use the dataset from the APB1 benchmark [13]. The star schema of this benchmark has one fact table Actvars and four dimension tables:

Actvars(Product_level, Customer_level, Time_level, Channel_level, UnitsSold, DollarSales, DollarCost) (24 786 000 tuples)

Prodlevel(Code_level, Class_level, Group_level, Family_level, Line_level, Division_level) (9 000 tuples)

Custlevel(Store_level, Retailer_level) (900 tuples)

Timelevel(Tid, Year_level, Quarter_level, Month_level, Week_level, Day_level) (24 tuples)

Chanlevel (Base_level, All_level) (9 tuples)

Two new attributes, week_level and day_level have been added to Timelevel table to facilitate an adequate management of updates (see section 2.3). This warehouse has been populated using the generation module of APB1. This warehouse has been installed under ORACLE 9i on a Pentium IV 1,5 Ghz microcomputer (with a memory of 256 Mo and two 7200 rps 60 Go disks) running under Windows 2000 Pro.

2.2 Workloads

The workloads used for our experiments focus on star queries. Each one has local restrictions defined in the involved dimension tables. We consider restrictions defined with predicates having only equality operator: ***A* = *value***, where ***A*** is an attribute name of a dimension table and ***value*** $\in$ domain(A). When a query Q_i of a workload involves such a restriction, the workload that we consider will have n_i potential queries, where n_i represents the cardinality of domain(A). In other words, there is a potential query for each different value of domain(A). Q_i is called a *parameterized query*, and $Q_i(n_i)$

denotes its set of potential queries. For example, if we consider the parameterized query involving the predicate gender = "M", the workload will have 2 potential queries: one with the previous predicate and another with gender = "F".

2.3 Update Operations

Since the materialized views and indexes are redundant data structures, they should be maintained periodically to reflect the interactive nature of the data warehouse. The cost of maintaining materialized views, indexes and even fragments should be taken into account when combining these three techniques in order to reduce the maintenance overhead. In this study, we incorporate updates into the workload. We suppose that they occur at *regular intervals* and a certain number of queries are executed between two updates. Based on the periodicity of updates, we consider two scenarios (that keep the size of the data warehouse constant):

1. *UD (that means that Updates occur each Day).* It consists in deleting the N oldest tuples of the fact table and inserting N new tuples (N is equal to the average activity of a day). Since our warehouse memorizes 24 786 000 tuples in the fact table for a period of 517 days, the activity of one day corresponds to an average of 47 942 tuples. So, with this scenario, 47 942 old tuples (of the fact table) are deleted and 47 942 new tuples are inserted.
2. *UW (that means that Updates occur each Week).* This scenario is similar to the previous one, except that the periodicity of the update operations is a week. Therefore 335 594 old tuples are deleted and 335 594 new ones are inserted in the fact table.

For these scenarios, we measure the time that Oracle uses to execute all operations (updating the raw data, the materialized views, the indexes and the fragments).

2.4 Cost Models for Calculating the Total Execution Time

Let S be a set of parameterized queries $\{Q_1, Q_2, \ldots, Q_m\}$, where each Q_i has a set of potential queries $(Q_i(n_i))$. To calculate the total time to execute S and one update, we use two simple cost models called Independent_Total_Time(TI) and Proportional_ Total_Time(TP). In TI, we suppose that the frequencies of the queries are equal and independent of the number of potential queries $Q_i(n_i)$. In the second one, the frequency of each Q_i is proportional to its n_i. Each model (TI and TP) will be used under the scenarios UD (where we consider TID and TPD) and UW (by considering TIW and TPW). Let t(O) be the execution time of an operation O (that can be a query, an UD or an UW). We define four cost models according to each models: TID, TIW, TPD, and TPW and defined as follows:

$$\mathrm{TID}(\alpha) = \alpha\left(\sum_{i=1..m} t(Q_i)\right) + t(\mathrm{UD}) \tag{1}$$

$$\mathrm{TIW}(\alpha) = \alpha\left(\sum_{i=1..m} t(Q_i)\right) + t(\mathrm{UW}), \text{ where } \alpha \text{ is a positive integer,} \tag{2}$$

$$\mathrm{TPD}(\beta\%) = 0.01 * \beta * \left(\sum_{i=1..m} n_i * t(Q_i)\right) + t(\mathrm{UD}) \tag{3}$$

$$\mathrm{TPW}(\beta\%) = 0.01 * \beta * \left(\sum_{i=1..m} n_i * t(Q_i)\right) + t(\mathrm{UW}) \tag{4}$$

where β is an integer taken in the interval [0, 100]. In other words, TID(α) and TIW(α) give the total time needed to execute α times each query plus an update. TPD(α%) and TPW(α%) give the total time required to execute β% of the potential queries plus an update.

2.5 An Implementation of the Derived Horizontal Partitioning

To partition dimension tables, Oracle provides several techniques: range, hash and hybrid partitioning. The attributes used for partitioning are called*fragmentation attributes*. Oracle and the existing commercial systems do not allow partitioning of the fact table using a fragmentation attributes belonging to a dimension table, as we will illustrate in the following example.

Suppose that we want to partition the fact table Actvars based on the virtual fragmentation schema of the dimension table ProdLevel (we assume that this former is partitioned into 4 *disjoint* segments using the attribute class_level). This will accelerate OLAP queries having restriction predicates on Class_Level. Based on this fragmentation, each tuple of a segment of the fact table will be connected to *only one* fragment of dimension ProdLevel. To achieve this goal (fragmenting the fact table based on the fragmentation schema of ProdLevel), we partition the fact table using one of the different partitioning modes (Range, Hash, Hybrid) available in the commercial systems based on the foreign key (Product_level)[1]. This fragmentation will generate 900 fact segments instead of 4 segments[2]. This example motivates the need of a mechanism that implements the derived fragmentation of the fact table. To do so, we propose the following procedure:

```
1- Let $A = \{A_1, \ldots, A_p\}$ be the set of fragmentation attributes.
2- For each $A_i (1 \leq i \leq p)$ do
    2.1- Add a new column (attribute) called connect$_i$ (whose domain is an integer) in the fact table. %This column gives the corresponding segment of each tuple of the fact table. For example, if the value of connect$_i$ of a given column is 1; this means that this tuple is connected (joined) to the segment 1 of the dimension table used to partition the fact table%
    2.2- For each tuple of the fact table, instanciate the value of connect$_i$.
3- Specify the fragmentation of the fact table by using the attribute connect$_i$ with one of the partitioning modes (range, hash, hybrid, etc.).
```

To take into account the effect of data partitioning, the queries must be rewritten using the attribute connect$_i$. This implementation needs extra space (for storing the attribute(s) connect$_i$). It requires also an extra time for the update operations.

[1] The foreign key is the single attribute that connects the fact table and the dimension table.

[2] The number of fragments of the fact table is equal the number of fragment of the dimension table.

3 Comparing Derived Partitioning and Join Indexes

3.1 Queries

For this comparison, we consider separately eight queries Q1 to Q8. The queries Q1 to Q5 have one join operation and one restriction predicate. The queries Q6 to Q8 have two joins operations and two restriction predicates. Each restriction predicate has a selectivity factor. The workload and the star schema used in our experiments are given in [6] (due to the space constraint).

3.2 Experimental Results

To identify the situations where the use of derived partitioning is interesting, we have conducted three series of experiments: (1) without optimization techniques; (2) only data partitioning is used (and depends on each query), and (3) only join indexes are used. When the data partitioning is used, we have considered for each query, a number of partitions equal the number of different values of its restriction attribute. Based on this number, we use the range mode (R) when it is small, otherwise, the hash mode (H) is used. The hybrid mode is used when queries have two join operations and two restriction predicates.

Table 1. The results for the first serie (without optimization techniques)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Query time (s)	106	61	63	53	54	61	59	56
UD (s)	68	68	68	68	68	68	68	68
UW (s)	75	75	75	75	75	75	75	75
TID(1)	174	129	131	121	122	129	127	124
TID(10)	1128	678	698	598	608	678	658	628
TPD(5%) (s)	89	105	118	269	878	214	493	2588
TPD(25%) (s)	174	251	320	1075	4118	800	2192	12668
TIW(1)	181	136	138	128	129	136	134	131
TIW(10)	1135	685	705	605	615	685	665	635
TPW(5%) (s)	96	112	125	276	885	221	500	2595
TPW(25%) (s)	181	258	327	1082	4125	807	2199	12675

When the join indexes are used, we select one index for the queries with one join and two indexes for those with two joins. For each query, we report the extra space used either by the partitioning or by the indexes, the query time, the update time for UD and UW, TID(α) and TIW(α) for two values of α(1 and 10), the values of TPD(β%) and TPW(β%)) for two values of β (5 and 25). The results are reported in the three tables Table 1, Table 2, Table 3 (one for each series).

3.3 Comments

Based on these results, the following comments are issued:

Query Time: We observe that partitioning gives a profit even for a low selectivity. The profit is very important with a high selectivity (when the number of different values for a restriction attribute is greater than 50). Join indexes give also a profit as soon as the selectivity is sufficiently high (more than 10 different values for the selection attribute). But partitioning gives better results compared to join indexes.

Update Time: Join indexes are in general much more efficient than partitioning. Partitioning performs as well as indexes only for low selectivity and for daily updates. With partitioning, it is important to limit the number of partitions (less than 100 for our benchmark), otherwise the update time becomes very high. Therefore, we need to partition the fact table into *a reasonable number* of segments.

TI and TP Model: It appears that partitioning is in general much more interesting than join indexes. Join indexes give better results only for high selectivity and small values of α and β (this means that the query frequency is almost the same as the update frequency).

These series of experiments summarize the following tuning recommendations when optimizing a parameterized query:

Rule 1: Data partitioning is recommended when (1) the selectivity factor of restriction predicate used in the query is low **or** (2) when the frequency of the update operation is low compare to the query frequency.

Rule 2: Join indexes are recommended when (1) the selectivity is high **or** (2) when the frequency of the update is similar to the query frequency.

In addition it is important to note that partitioning requires an additional space more significant than those required by indexes. But it remains acceptable (10% of the total space occupied by the warehouse if the number of partitions is limited to 100). This additional space is justified by the fact of adding a new column (connect) in the fact table (see Section 2.3).

Table 2. Results for the second serie (data partitioning) (the best score for the three situations is represented in bold)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Number of partitions	4 (4R)	12 (12R)	15 (15R)	75 (75H)	300 (300H)	48 (4R*12H)	144 (12R*12H)	900 (12R*75H)
Extra space (Mo)	74	90	123	170	194	156	174	431
Query time (s)	53	9	13	4	1	11	2	1
UD (s)	69	70	70	88	112	86	105	135
UW (s)	105	92	121	154	199	144	164	220
TID(1)	**122**	**79**	**83**	**92**	113	97	107	136
TID(10)	**599**	**160**	**200**	**128**	**122**	**196**	**125**	145
TPD(5%) (s)	**80**	**75**	**80**	**103**	**127**	**112**	**119**	**180**
TPD(25%) (s)	**122**	**97**	**122**	**164**	**187**	**218**	**177**	**360**
TIW(1)	**158**	**101**	**134**	158	200	155	166	221
TIW(10)	**635**	**182**	**251**	**194**	209	**254**	**184**	230
TPW(5%) (s)	**116**	**97**	**131**	**169**	**214**	170	178	265
TPW(25%) (s)	**158**	**119**	**173**	**230**	**274**	**276**	**236**	**445**

Table 3. Results for the third serie (join indexes) (the best score for the three situations is represented in bold)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Join indexes	JI1	JI2	JI3	JI4	JI5	JI1+JI2	JI2+JI3	JI2+JI4
Extra space (Mo)	11	8	20	35	59	19	28	43
Query time (s)	152	24	57	26	8	19	9	3
UD (s)	69	69	71	69	68	68	71	69
UW (s)	98	99	101	101	104	109	110	110
TID(1)	221	93	128	95	**76**	**87**	**80**	**72**
TID(10)	1589	309	641	329	148	258	161	**99**
TPD(5%) (s)	99	83	117	168	188	114	136	204
TPD(25%) (s)	221	141	299	563	668	296	395	744
TIW(1)	250	123	158	**127**	**112**	**128**	**119**	**113**
TIW(10)	1618	339	671	361	**184**	299	200	**140**
TPW(5%) (s)	128	113	147	200	224	**155**	**175**	**245**
TPW(25%) (s)	250	171	329	595	704	337	434	785

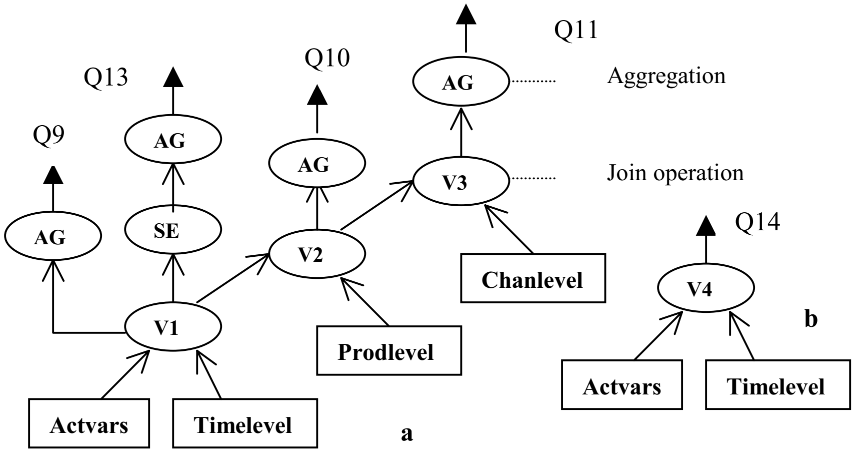


Fig. 1. The materialized views V1 to V4 used in case 2 and in case 3

4 Combining the Three Techniques

4.1 The Queries and the Cases

To evaluate the result of combining the three techniques, we conduct experiments using six SJA (Select, Join, Aggregation) queries noted Q9 to Q14. All queries are parameterized, except the query Q14. To capture the effect of the data partitioning, the number of restriction predicates in each query (except the query Q14) is equal the number of join operations.

Since a fragment is a table, the three techniques can be combined in various ways: selecting indexes and/or views on a partitioning, installing indexes and/or partitions on views, selecting views and indexes separately. We consider the following cases:

Case 0 (for comparison purpose): None optimization technique is considered;
Case 1 (to test separately the ability of a set of joins indexes): The four join indexes JI1 on actvars (timelevel.month_level), JI2 on actvars(prodlevel family_level), JI3 on actvars(prodlevel.grouplevel), JI4 on actvars(chanlevel.all_level);
Case 2 (to situate separately the performances of a set nested materialized views): The three nested views of figure 1a;
Case 3 (idem as the previous one with one view more): The views of figure 1a and figure 1b;
Case 4 (to show the interest of associating a star transformation with join indexes): The star transformation[3] with the four join indexes of case 1, plus the bitmap index incorporating the join with Timelevel, plus the view V4 of figure 1b;
Case 5 (to evaluate the partitioning): The derived partitioning into 96 partitions using month_level (12R) + all_level (8H) (R and H mean Range and Hash, respectively);
Case 6 (to test partitioning): The derived partitioning into 72 partitions using all_level (9R) + family_level (8H);
Case 7 (to test partitioning): The derived partitioning into 144 partitions using all_level (9R) + family_level (16H);
Case 8 (to test partitioning in association with indexes): The derived partitioning into 72 partitions using all_level (9 R) + family_level (8H), plus the bitmap index on actvars(derived attribute of month_level), plus the join index JI3;
Case 9 (to test partitioning): The derived partitioning into 96 partitions using month_level (12R) + H(family_level (8H);
Case 10 (to test partitioning in assocition with index): The derived partitioning into 96 partitions using month_level (12R) + H(family_level (8H) plus the join index JI3
Case 11 (to test partitioning in association with index): The derived partitioning into 96 partitions using month_level (12R) + H(family_level (8H), plus the join JI3, plus the view V4 of figure 5b.

4.2 Experimental Results and Comments

The results obtained for these 12 cases are reported in table 4 (ES means extra space). The execution times are given in seconds. In table 4 we found also the extra space which is needed to install the different objects (indexes, views, partitions), the values of TID(α) and TIW(β) for two values of α (1 and 10), the values of TPD(β%) and TPW(β%)) for two values of β (1 and 10).

When partitioning is used, the number of partitions should be limited to a reasonable value (less than 100 for our benchmark). With a number of partitions greater than 100, the time used for the updates becomes very high. Moreover some partitioning can disadvantage queries (for example case 6 for Q9 and Q13). Update times are highest when materialized views are used. This limits seriously the interest of the views in this kind of situations. However some configurations can give good global results despite

[3] The star transformation is a powerful optimization technique that relies upon implicitly rewriting (or transforming) the SQL of the original star query. The end user never needs to know any of the details about the star transformation. Oracle's cost-based optimizer automatically chooses the star transformation where appropriate.

Table 4. The results of the experiments for studying the combination of the three techniques (best score in bold, second score in grey)

Case	ES	Q9	Q10	Q11	Q12	Q13	Q14	UD	UW	TID (1)	TID (10)	TIW (1)	TIW (10)	TPD (1%)	TPD (10%)	TPW (1%)	TPW (10%)
0		61	59	65	63	58	102	65	74	473	4145	482	4154	8383	81216	8392	81225
1	158	28	5	3	29	1	102	76	138	244	1756	306	1818	600	5587	662	5649
2	301	12	1	1	77	10	103	131	292	335	2171	496	2332	-	-	-	-
3	302	12	1	1	77	10	1	147	390	249	1167	492	1410	-	-	-	-
4	167	29	5	2	20	1	1	113	236	**171**	693	**294**	**816**	494	4805	617	4928
5	279	15	10	4	31	36	97	90	145	283	2020	338	2075	2012	19445	2067	19500
6	268	78	12	1	1	71	97	98	185	358	2698	445	2785	2860	29879	2947	29966
7	297	97	6	1	1	91	126	161	217	483	3381	539	3437	3592	36628	3648	36684
8	303	46	12	1	2	1	97	122	258	281	1712	417	1848	367	4664	503	4800
9	279	16	2	1	12	35	97	90	164	253	1720	327	1794	1533	15936	1607	16010
10	305	15	2	1	12	1	97	96	176	224	1376	304	1456	**315**	**3701**	**395**	**3781**
11	305	16	2	2	12	1	1	169	485	203	**509**	519	825	468	4576	784	4892

high update times and views can be profitable in association with others techniques. Materialized views are profitable for parameterized queries if we select a view for each value of the parameter. In general, this is not acceptable due to the huge storage and update time that we should allocated to them (materialized views). We observe also that the extra space needed to install the objects (indexes, views, partitions) remains less than 300 Mo (i.e. about 15% of the total space occupied by the warehouse).

The important observation is that join indexes alone, or materialized views alone, or partitioning alone do not provide the best results for TI or for TP. Best results are obtained when two or three techniques are combined: case 4 which combines through a star transformation the join indexes, a bitmap index and a view; case 10 which combines partitioning and a join index; case 11 which at more associates a view. Case 4 is well suited for TI but not for TP. Case 10 and 11 give good results both for TI and for TP. Combinations involving partitioning are recommended when the database administrator wants to optimize parameterized queries first.

We observe also that some configurations are good for several situations. It would be very interesting to determine such robust configurations since they remain valid after some changes in the use of the data warehouse (changes in the frequencies of queries, changes of queries, ...).

5 Conclusion

The objective of this paper was to explore the possibilities of combining materialized views, indexes and partitioning in order to optimize the performances of relational data warehouses. Firstly, we have proposed an implementation of derived horizontal partitioning that allows the use of different modes of partitioning available (like range, hash and hybrid). We have compared join indexes and horizontal derived partitioning. Our results show that partitioning offers better performance (for query processing time), especially when the selectivity of the restriction predicates is low. With regard to the updates, it is less interesting, primarily when the number of partitions is high. When updates and queries interleave, a partitioning on an attribute A with n different values is advantageous as soon as a parameterized query on A is executed more than 0.05*n times between two updates. This work shows that the two techniques are rather complementary. There is thus interest to use them jointly as it had been already underlined through a theoretical study [5].

We have further compared different configurations mixing the three techniques to optimize a given set of queries. It appears that each technique used alone is not able to give the best result. Materialized views contribute in optimizing parameterized queries, but they require huge amount of storage (but they remain a good candidate for optimizing non parameterized queries). Along these experiments, two performance scenarios are distinguished: the first one is based on a star transformation with join indexes with complementary structures such as bitmap indexes or views; another one based on partitioning with complementary structures such as join indexes or views. We have noticed that the second scenario is robust since it gives good results for different situations.

Our experiments do not cover all the various uses of a warehouse. Different points should be explored in the future such as: the consideration of other types of queries (those having restrictions with OR operations, nested queries, etc.); the influence of other kinds of updates. Nevertheless, these results allow us to list some recommendations for better tuning the warehouse: (1) the horizontal derived partitioning can play an important role in optimizing queries and the maintenance overhead, (2) incorporation of updates into the workloads may influence the selection of materialized views, indexes and data partitioning, (3) partition the fact table into a reasonable number of fragments rather having a huge segments. We think that these recommendations open the way for new algorithms for selecting simultaneously fragments, indexes and views in order to accelerate queries and optimize the maintenance overhead.

References

1. S. Agrawal, S. Chaudhuri, V.R. Narasayya, “Automated selection of materialized views and indexes in SQL databases”, in Proc. 26th Int. Conf. on Very Large Data Bases (VLDB), pp. 496-505, 2000.
2. E. Baralis, S. Paraboschi, and E. Teniente, “Materialized view selection in a multidimensional database,” in Proc. 23rd Int. Conf. on Very Large Data Base (VLDB), pp. 156-165, 1997.
3. L. Bellatreche, K. Karlapalem, and Q. Li, “Evaluation of indexing materialized views in data warehousing environments”, in Proc. Int. Conf. on Data Warehousing and Knowledge Discovery (DAWAK), pp. 57-66, 2000.
4. L. Bellatreche, K. Karlapalem, M. Schneider and M. Mohania, “What can partitioning do for your data warehouses and data marts”, in Proc. Int. Database Engineering and Application Symposium (IDEAS), pp. 437-445, 2000.
5. L. Bellatreche, M. Schneider, M. Mohania, and B. Bhargava, “Partjoin : an efficient storage and query execution design strategy for data warehousing”, Proc. Int. Conf. on Data Warehousing and Knowledge Discovery (DAWAK), pp. 296-306, 2002.
6. L. Bellatreche, M. Schneider, Lorinquer, H. and M. Mohania, “Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses,” extented version available at http://www.lisi.ensma.fr/publications.php
7. S. Chaudhuri and V. Narasayya., “An efficient cost-driven index selection tool for microsoft sql server”, in Proc. Int. Conf. on Very Large Databases (VLDB), 1997, pp. 146-155.

8. C. Chee-Yong, “Indexing techniques in decision support Systems”, Ph.D. Thesis, University of Wisconsin, Madison, 1999.
9. H. Gupta et al., “Index selection for olap,” in Proc. Int. Conf. on Data Engineering (ICDE), pp. 208-219, 1997.
10. H. Gupta and I. S. Mumick, “Selection of views to materialize under a maintenance cost constraint,” in Proc. 8th Int. Conf. on Database Theory (ICDT), pp. 453-470, 1999.
11. Informix Corporation, “Informix-online extended parallel server and informix-universal server: A new generation of decision-support indexing for enterprise data warehouses”, White Paper, 1997.
12. Nicola, M. ”Storage Layout and I/O Performance Tuning for IBM Red Brick Data Warehouse”, IBM DB2 Developer Domain, Informix Zone, October 2002.
13. OLAP Council, “APB-1 olap benchmark, release II”, http://www.olapcouncil.org/research/bmarkly.htm.
14. P. O’Neil and D. Quass., “Improved query performance with variant indexes”, in Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 38-49, 1997.
15. Red Brick Systems, “Star schema processing for complex queries”, White Paper, July 1997.
16. A. Sanjay, G. Surajit, and V. R. Narasayya, “Automated selection of materialized views and indexes in microsoft sql server”, in Proc. Int. Conf. on Very Large Databases (VLDB), pp. 496-505, 2000.

GeoDWFrame: A Framework for Guiding the Design of Geographical Dimensional Schemas

Robson N. Fidalgo[1,2], Valéria C. Times[1], Joel Silva[1], and Fernando F. Souza[1]

[1] Center of Informatics – Federal University of Pernambuco
p.o. box 7851-50732-970 - Recife - PE – Brazil
{rdnf,vct,js,fdfd}@cin.ufpe.br
[2] Curso de Sistemas de Informação - Faculdade Integrada do Recife
Av. Eng. Abdias de Carvalho, 1678 - 50720-635 - Recife - PE – Brazil
rfidalgo@fir.br

Abstract. Data Warehouse (*DW*) is a dimensional database for providing decision support by means of on-line analytical processing (*OLAP*) techniques. Another technology also used to provide decision support is Geographical Information System (*GIS*). Much research aims at integrating these technologies, but there are still some open questions, particularly regarding the design of a geographical dimensional data schema. This paper discusses some related work and proposes *GeoDWFrame* that is a framework based on the star schema and has been specified as guidance for designing geographical dimensional schemas. Some experimental results are also given.

1 Introduction

Data Warehouse (*DW*) [1,2,3,4] and On-Line Analytical Processing (*OLAP*) [4,5] are traditional technologies for decision support. *DW* is a dimensional database that is organised over two types of tables: 1) dimensions – address descriptive data and 2) facts – address measuring data. According to Kimball [2,3], most useful facts are numerical, additive, and continuously valued and the dimensional data are intrinsically desnormalized and normally organized in hierarchical structures to support different levels of aggregation. Another kind of dimensional database is a Data Mart (*DM*) [1,2,3,4], which can be defined as a specific or departmental *DW*[1]. Regarding *OLAP*, it is a specific software category for providing strategic and multidimensional queries over the *DW* data.

Another technology for decision support, but specifically inside of a spatial context, is the Geographical Information System (*GIS*) [6,7]. This system helps in acquiring, manipulating, examining and presenting geo-objects. Each one of these objects is composed of its descriptive information (conventional data) and its geographical reference (geometrical data). These objects are viewed as maps that can be combined, one above the other, in order to provide layers of geo-information.

[1] In this paper, we do not make a rigorous distinction between *DW* and *DM*.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 26–37, 2004.

Concerning the integration among *DW*, *OLAP* and *GIS*, some research has already been done (see section 3). However, these approaches do not deal with open and extensible solutions and do not converge on some points of the dimensional scheme of a Geographical *DW* (*GDW*). In order to address the first problem, the Geographical On-Line Analytical Processing Architecture (*GOLAPA*) [8,9] was proposed. Regarding the second problem, we propose the framework *GeoDWFrame*, which is based on the star schema [1,2,3,4] and guides the design of *GDW* schemas.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of geographical decision support. Section 3, discusses some related work. Then, section 4, proposes the framework *GeoDWFrame*. Following this, some experimental results are given in section 5 and section 6 presents our conclusions and future work.

2 Geographical Decision Support

Decision support can be defined as a field of information technology that permits the processing of large database in order to extract information to help in understanding the behaviour of the business data of an organization. *DW*, *OLAP* and *GIS* are tools to provide decision support, but different to *DW* and *OLAP*, *GIS* are essentially transaction-oriented tools. That is, it allows, with few restrictions to execute the traditional transactional operations (insert, delete, update and query) over its data. To the contrary, *DW*/*OLAP* essentially just permits to load and query of its data.

Although a detailed study about the requirements of an Analytical *GIS*[2] (*AGIS*) has not yet been completely done, we understand that the use of a *GDW* for this kind of system could bring a lot of benefits. With a *GDW*, an *AGIS* takes the advantage of the dimensional approach and of the *DW* intrinsic concepts [1] (subject-orientated, integrated, time-variant and non-volatile) and moreover, the *GDW* explicitly will separate the transactional environment data of the decision environment data, which will prevent that any operation that occurs in the transactional environment will not be reflected in the decision environment.

Some research (see section 3) has been addressed on the *GDW*, but some points still need to be further discussed, especially the design of its dimensional schema. However, this research agrees in one point: that the *GDW* schema is an extension of a start schema. That is, it extends this schema to insert geographical data, which are normally defined in a dimension known as *Spatial* or *Geographical*. We highlight that a *GDW* should keep the intrinsic concepts of a *DW* and moreover provide support to store, index, aggregate and analyse (in table or map) geo-data. Realize, as a *DW*, a *GDW* requires a data staging processing [3] to address spatial data heterogeneities. In other words, before loading these data into a *GDW*, they must be cleaned and integrated, for example, they must have correct granularity, representation and topology and the spatial data must have the same format, scale, projection and spatial reference system.

[2] We understand an *AGIS* as a *GIS* that basically just allows load and query of data.

With regard to geo-granularity, we highlight that a *GDW* must support queries over features that are not known in advance (e.g. an ad-hoc area or intersected areas). In order to address this, the finest geographical granularity of *GDW* must be defined by features that symbolize points-objects. These features represent the finest geo-fact and where these occur, and these also can be successively aggregated to compose another fact of a feature defined on running time. For example: "*sales in 2003 by state and stores where the stores are located X distance from a point Y*". In this example we process the *OLAP* query (e.g. *"sales in 2003 by state and stores*"), then we process the *GIS* query (e.g. *"where the stores are located X distance from a point Y"*) and finally we process the intersection of last queries. Figure 1 depicts this example. Note, if regions of store (polygons-objects) were the finest geo-granularity, we would not have enough information for processing this query, because we need the facts occurred in locations of stores (points-objects) to answer this query.

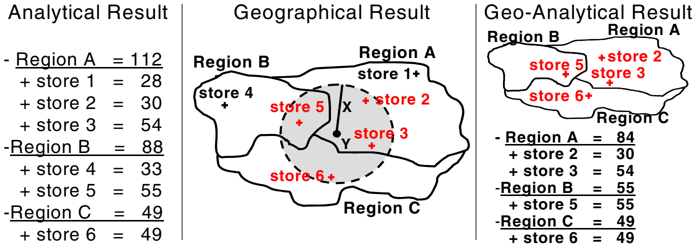


Fig. 1. Finest geo-granularity vs. queries over ad-hoc areas

3 Related Work

Much research [10–19] addresses the use of *DW* with geographical data, but most of them do not propose a true *GDW*, because in some cases [18,19] the descriptive data are handled in *DW* while the spatial data are handled in *GIS*, which stores its data in proprietary files (e.g. .SHP and .MIF).

From previous research, the work of Han et al [10,11] is the more relevant, because this also proposes a framework to address a true *GDW*, that is, both the descriptive and geometrical data are stored in *GDW*. The framework of Han et al. is based on a star schema and considers three types of dimensions and two types of measures. The dimensions are: non-geometrical (all levels contain only descriptive data), geometrical-to-non-geometrical (the finest granularity level contains only geometrical data and the others only descriptive data) and fully-geometrical (all levels contain only geometrical data). With regard to the measures, these are: numerical (only additive data) and spatial (a collection of pointers to spatial objects). Figure 2 shows an example of a *GDW* based on this framework. In the Figure 2, *Temperature*, *Precipitation* and *Time* are non-geometrical dimensions, *Region-name* is a fully-geometrical dimension, *area* and *count* are numerical measures and *region-map* is a spatial measure.

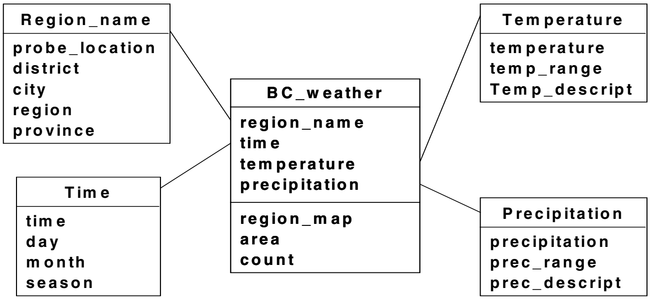


Fig. 2. *GDW* model of Han et. al [10]

With regard to the research of Han et al, there are four issues that need to be further discussed. These issues motivated the proposal of the *GeoDWFrame*. The first debates the use of spatial measures. In any *DW*, a fact (or measure) is a quantitative value and not a collection of pointers. For this, we understand that a spatial measure will be defined as a textual field, which needs first to be parsed for just afterwards to be processed, probably increasing the cost and complexity of computing. Another point about spatial measures is: if spatial dimensions already address the spatial data, why not process this data to achieve an equivalent result to spatial measure. This would be more natural, because dimension intrinsically deals with textual fields and these data would not need to be parsed. Based on Figure 2, Figure 3 exemplifies the use of a spatial measure, where *Collection_of_Spatia_id* is a spatial measure. Figure 4 shows how to achieve an equivalent result to Figure 3, without applying the spatial measure. In Figure 4, *Time* and *Spatial* are dimensions and *Temperature* and *Precipitation* are numerical measures. Note that despite the Figures showing the information in different forms, the Figures have equivalent information.

Time	Temperature	Precipitation	Collection of Spatial_ids
March	cold	0.1 to 0.3	{AL04, AM03, … XN87}
March	cold	0.3 to 1.0	{AM10, AN05, … YP90}
. . .	. . .	. . .	. . .

Fig. 3. An example applying spatial measure [10]

Time	Spatial	Temperature (^{O}C)	Precipitation
March	AL04	2.5	0.1
March	AM03	3.0	0.2
March	. . .	. . .	. . .
March	XN87	1.5	0.3
March	AM10	3.5	0.4
March	AN05	2.0	0.7
March	. . .	. . .	. . .
March	YP90	1.5	0.9
. . .	. . .	. . .	. . .

Fig. 4. The previous example (Fig. 3) without applying spatial measure

The second ponders the intrinsic redundancy of dimensional data [1,2,3] and its impact over the high cost of storing geometrical data in common dimensions. The *GDW* model in discussion does not present an approach to minimize this issue, but a *GDW* can have more than one dimension with the same spatial levels (e.g. the *Store* and *Customer* dimensions have the same levels: *Counties, Regions, States* and *Cities*) or even if a *GDW* has just one dimension with spatial levels, there will still be the intrinsic dimensional redundancy of geometrical data to be addressed.

The third argues that a geometrical-to-non-geometrical dimension just allows geometrical data in the finest granularity level. This fact may limit its use, because in practice, it is relevant that a *GDW* can permit geometrical data at any level. Otherwise, just the finest granularity level could be drawn on the map. And finally, the fourth questions where the descriptive data of geographical objects are. These are useful for *OLAP* queries, but the dimensions in discussion just handle geometrical data.

4 GeoDWFrame

Kimball [2,3] defined several techniques and guidelines to build a conventional *DW*. From Kimball's work and the discussion above, we propose the framework *GeoDWFrame*. From the evaluation of four issues previously discussed, we decided that *GeoDWFrame,* respectively: 1) does not apply spatial measure, 2) normalizes the geometrical data, 3) provides geographical data in any dimensional level and 4) stores the descriptive data of geo-objects. In order to provide support for these issues, *GeoDWFrame* proposes two types of dimensions, namely: geographical and hybrid. The first one is classified into primitive and composed, while the second one is classified into micro, macro and joint. The primitive and composed dimensions have at all levels (or fields) just geographical data (e.g. *client addresses* and its geo-references), while the others deal with geographical and conventional data (e.g. *client addresses* and its geo-references plus *ages* and *genders*). Besides these dimensions, the *GeoDWFrame*, as in any *DW*, also supports dimensions with just conventional data (e.g. a *product* dimension). In the sequence, the differences between primitive and composed dimensions are given.

A primitive geographical dimension represents the geo-coding data (geometries) used to handle spatial objects. Basically, it can be implemented using two approaches: 1) a relational database with long field (e.g. *Text, Long* or *BLOB*) or 2) a pos-relational database that supports a geometry abstract type (e.g. *Oracle SDO_GEOMETRY* or *DB2 ST_GEOMETRY*). With regard to the first approach, it allows the storage of geo-coding data in a textual or binary format, but the spatial processing (e.g. query and indexing) is performed by geographical software and not by the database system. Regarding the second approach, the database system performs the spatial processing, but the spatial extensions have not been standardized yet (e.g. divergence of spatial operators and spatial data type), which results in a proprietary solution.

Figure 5 shows an example of how the last approaches can model polygons. In the Figure 5-a, the fields *GeoPK, Xmim, Ymim, Xmax* and *Ymax* are conventional (e.g. *Varchar2* or *Number*) while the *Geo* is a long field. In Figure 5-b, the *GeoPK* field is also conventional, but the *Geo* is a geometry abstract type. These fields represent the identification of an object (*GeoPK*), its Minimum Bounding Rectangle (*MBR*) (*Xmim, Ymim, Xmax* and *Ymax*) and its geo-coding (*Geo*). Realize that a primitive geographical dimension just deals with geometrical data. For this, it is not appropriate to *OLAP* queries. However, this is essential to: 1) support geo-operations, 2) draw the features and 3) keep the historic of the geo-objects applying slowly changing dimensions [2,3].

GeoPK	: Conventional
Xmim	: Conventional
Ymim	: Conventional
Xmax	: Conventional
Ymax	: Conventional
Geo	: Long Field

a) Relational approach

GeoPK	: Conventional
Geo	: Geometry Abstract Type

b) Pos-relational approach

Fig. 5. Examples of how representing a primitive geographical dimension

The quantity of fields in a primitive geographical dimension may change according to: 1) the type of the spatial object (e.g. a point does not need a *MBR*), 2) the geographical software that will be used (e.g. *TerraLib* [20] needs more fields to handle multi-polygons) and 3) the chosen approach (relational or pos-relational). However, basically we can focus on two fields: one to store the identification of the spatial object (*GeoPK*) and another to store its geographical representation (*Geo*).

In *GOLAPA*, in order to be as open and extensible as possible, the primitive geographical dimension is implemented using the relational approach with the Geography Mark-up Language (*GML*) [21]. Note, in the future, when the spatial extensions of pos-relational database are completely standardized, these will be able to be used as well.

Concerning the composed geographical dimension (Figure 6), it contains fields to represents its primary key, the description of the geo-objects and its foreign keys to primitive geographical dimensions. In Figure 6, the geographical hierarchy is known in advance (*Region→Property→Lot*), which allows it to aggregate the levels of this hierarchy using conventional methods [22]. However, when the geographical hierarchy is unknown in advance (at design time), it is necessary, during the *GDW* building process, to apply spatial indexing methods [23] to define the order existing among the geographical objects. Realize that without the primitive geographical dimensions a dimension with spatial data would have to store their descriptions and geo-references together. Then due to the: 1) intrinsic redundancy of dimensional data and 2) high costs of storing the geo-references (compared to foreign keys to primitive dimension), this approach would result in a high cost of storage.

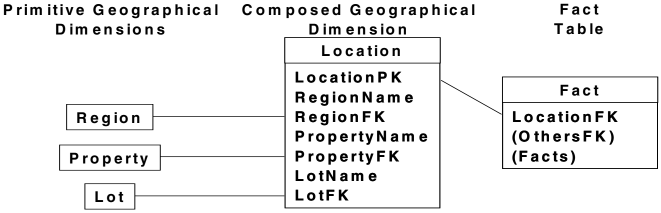


Fig. 6. An example of a composed geographical dimension

With regard to hybrid dimensions, that is, micro, macro and joint, these are described as follows. A micro hybrid dimension handles conventional and geographical data, but the geographical ones represent the finest geo-granularity of the dimension, which must represent points-objects (e.g. *Addresses* and *Lots*). Note that due to the small spatial granularity, other dimensions rarely share these geo-referenced data. A micro hybrid dimension (Figure 7) has conventional fields and two geographical fields (the descriptive and the foreign key for its primitive geographical dimension).

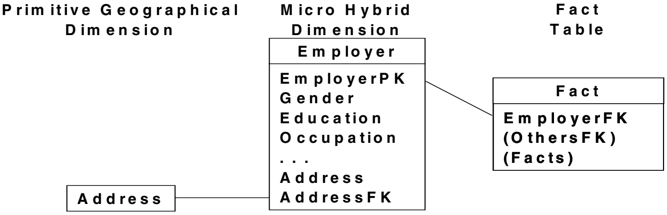


Fig. 7. An example of a micro hybrid dimension

A macro hybrid dimension, different to a micro hybrid dimension, handles geographical data that are usually shared (e.g. *Countries, Regions, States* and *Cities*). The schema of a macro hybrid dimension can be defined based on two approaches: 1) conventional fields plus one foreign key to a composed geographical dimension or 2) conventional fields plus one foreign key to each primitive geographical dimension that corresponds to each descriptive geographical field. Note that in the first one, the composed geographical dimension minimizes the number of foreign keys in the macro hybrid dimension and also can work as a mini-dimension or a role-playing dimension [2,3]. Figures 8 and 9 show these approaches, which can be applied in Figure 10 as well. In Figures 8 and 9, we assume *Lot* as the finest geo-granularity, for this, its geo-data must be defined as points-objects. However, in Figure 10, we assume *Region*, *Property* and *Lot* as polygons-objects and the *Address* as the finest geo-granularity, that is, points-objects.

A joint hybrid dimension merges the micro and macro approaches in just one dimension. Figure 10 draws an example of this dimension and Figure 11 shows the *GDW* of Figure 2 designed according to the *GeoDWFrame*. In Figure 11, the *Temperature* and *Precipitation* dimensions are useful to answer queries, such as, "*what is the area where the temperature description is cold and the precipitation range is 0.1 to 0.3*".

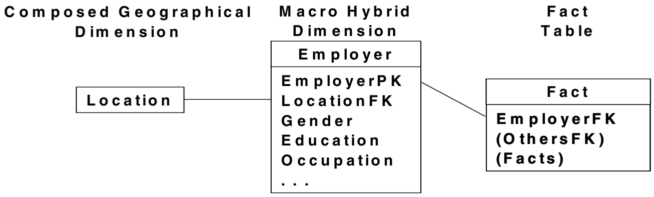


Fig. 8. An example of a macro hybrid dimension

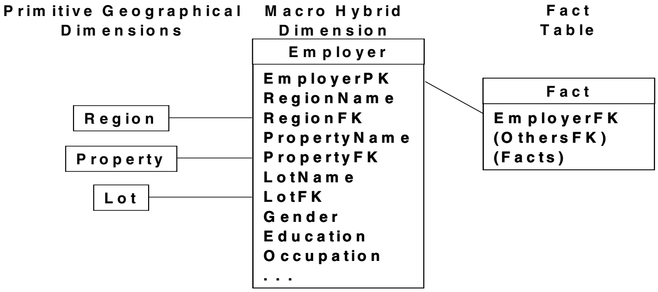


Fig. 9. Another example of a macro hybrid dimension

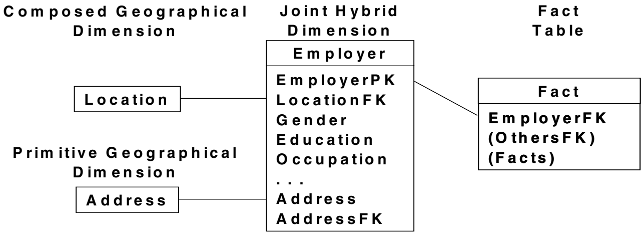


Fig. 10. An example of a joint hybrid dimension

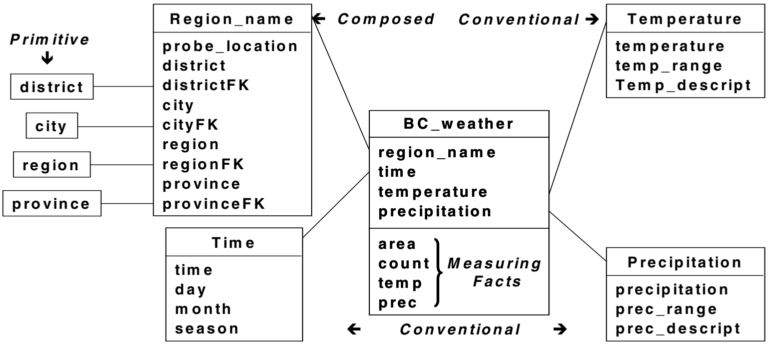


Fig. 11. The *GDW* of Figure 2 according to *GeoDWFrame*

In Figure 11, all measures are numerical facts. *Region_name* is a composed geographical dimension with its foreign keys to each primitive geographical dimension, while *Temperature, Precipitation* and *Time* are conventional dimensions.

5 Experimental Results

In order to gather some experimental results, we have extended a conventional Data Mart (*DM*) by some adding geographical properties according to the *GeoDWFrame*. Figure 12 presents the *Sales_Fact_1997 DM*, as our experimental dimensional schema.

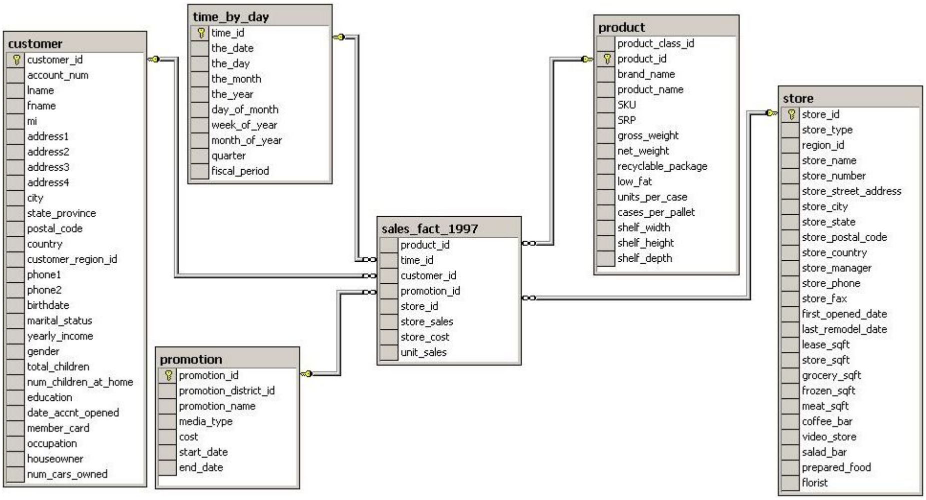


Fig. 12. The dimensional schema of *Sales_Fact_1997*

We have used the *Sales_Fact_1997 DM* of *Food-Mart DW*, because it can be easily acquired from the *SQL Server CD* installation [24]. To build the geographical *Sales_Fact_1997 DM* according to *GeodwFrame* we basically have: 1) updated its schema by: 1.1) defining the dimensions and its geo-levels (*County, State* and *City* of *Store* and *Customer*), 1.2) identifying the finest geo-granularity (*City*), 1.3) creating the dimensions (*Store* and *Customer* as macro hybrid dimensions, *Shared* as a composed dimension and *County, State* and *City* as primitive dimensions); 2) coded the geo-data into *GML* – we have get *SHP* files and use the tool *Degree Viewer/Converter* [25] to generate the *GML* data; 3) loaded the *GML* geometries in the *County, State* and *City* dimensions; 4) loaded the name of geo-objects and the foreign keys for the *County, State* and *City* dimensions into the *Shared* dimension; 5) added into *Store* and *Customer* dimensions the foreign key to *Shared* dimension; and finally, 6) removed the *County, State* and *City* levels of *Store* and *Customer* dimensions, because these now compose the *Shared* dimension. Figure 13 shows the *Sales_Fact_1997* dimensional schema based on *GeoDWFrame*.

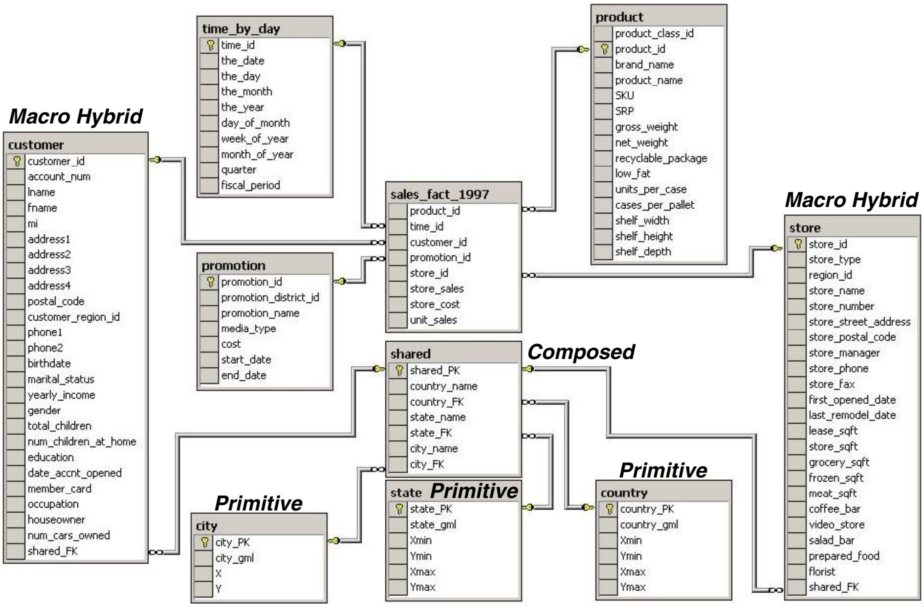


Fig. 13. The dimensional schema of *Sales_Fact_1997* based on *GeoDWFrame*

Figure 13 shows that the applied updates have kept the essence of *Sales_Fact_1997* schema. That is, the *Customer* and *Store* dimensions have been lightly updated while the remained dimensions and fact table have been completed preserved. It demonstrates the *GeoDWFrame* applicability and shows that its use guides the design of geographical dimensional schemas in order to reduce costs (the schema is lightly updated and free of geometry redundancy). Also, it does not apply spatial measure and allows the use of geometrical data at any level. Thus, *GeoDWFrame* provides an important contribution in the *GDW* design issue.

6 Conclusions and Future Work

Much research addresses the use of *DW* with *GIS*, but still lacks a consensus about the dimensional model of *GDW*. Aiming to address this, we analysed some related work (identifying its convergent and discussion points) to propose the framework *GeoDWFrame*. It is proposed to guide the design of geographical dimensional schemas that must support analytical, geographical or analytical-geographical operations. To provide this support, *GeoDWFrame* proposes two types of dimensions: geographical (primitive and composed) and hybrid (*micro, macro* and *joint*). The use of these dimensions aims to minimize the geo-data redundancy and moreover provide a natural support to multidimensional and/or geographical tools, because the *GDW* will handle descriptive and geometrical data and will not apply spatial measure. Note that according to *GOLAPA*, the *GeoDWFrame* uses *GML* to define geo-data, but the

GeoDWFrame is not limited to *GOLAPA* and, for this reason, it can use geometry abstract types to handle geo-data. Applying the *GeoDWFrame* in large *GDW* (with vector or raster data) and investigating the challenges to extract, transform and load geo-data into a *GDW* (based on the *GeoDWFrame*), composes our future work.

References

1. W. H. Inmon. Building the Data Warehouse. 2nd edition. John Wiley & Sons (1997)
2. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses, John Wiley & Sons, Inc. (1996)
3. Kimball R., Reeves L., Ross M., Thornthwaite W.: The Data Warehouse Lifecycle Toolkit. Expert Methods for Designing,Developing and Deploying Data Warehouses, Wiley Computer Publishing,New York (1998).
4. Chaudhuri S., Dayal U.: An overview of Data Warehousing and OLAP Technology. SIGMOD Record 26(1). 1997.
5. OMG: Common Warehouse Metamodel (CWM) Specification. (2002).
6. Longley, P. A., Goodchild, M. F., Maguire, D. J.. Geographical Information Systems: Principles, Techiniques, Applications and Managemente. 2nd Edition, John Wiley and Sons. (1999)
7. Demers, M. N.: Fundamentals of Geographic Information Systems. 2nd ed. John Wiley & Sons. (2000)
8. Fidalgo R. N., Times, V. C., Souza, F.F.:GOLAPA: An Open and Extensible Architecture for Integration between OLAP and GIS. Proc. Brazilian Workshop on Geoinformatics (GeoInfo). (2001) – In Portuguese.
9. GOLAPA Project: GOLAPA official home page – http://www.cin.ufpe.br/~golapa/architecture (2004)
10. Han, J., Stefanovic, N., Koperski, K.: Selective materialization: An efficient method for spatial data cube construction. Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD). (1998)
11. Bédard, Y., Merrett, T., Han, J.. Fundamentals of spatial data warehousing for geographic knowledge discovery. H. J. Miller and J. Han (Editors) Geographic Data Mining and Knowledge Discovery. (2001)
12. Rivest, S., Bédard, Y. and Marchand, P.: Towards better support for spatial decision-making: Defining the characteristics of Spatial On-Line Analytical Processing (SOLAP). Geomatica, 55(4). (2001)
13. Shekhar, S., Lu, C.T., Tan, X., Chawla, S., Vatsavai., R. R.: Map Cube: A Visualization Tool for Spatial Data Warehouses. In: H. Miller and J. Han (Editors), Geographic data mining and knowledge discovery. (2001)
14. Papadias, D., Kalnis, P., Zhang, J., Tao, Y.: Efficient OLAP Operations in Spatial Data Warehouses, Proc. Int. Symp.on Spatial and Temporal Databases, (SSTD). (2001).
15. Papadias, D., Tao, Y., Kalnis, P., Zhang, J.: Indexing SpatioTemporal Data Warehouses. Proc. Int. Conf. on Data Engineering (ICDE). (2002)
16. Zhang, L., Li, Y., Rao, F,. Yu, X., Chen, Y.: Spatial Hierarchy and OLAP-Favored Search in Spatial Data Warehouse, Proc. Int. Workshop on Data Warehousing and OLAP (DOLAP). (2003).
17. Gonzales, M.L.. Spatial *OLAP*: Conquering Geography. DB2 Magazine. (1999)
18. Kouba, Z., Matousek, K., Miksovsky, P.: On Data Warehouse and GIS integration. Proc. Int. Conf. on Database and Expert Systems Applications (DEXA). (2000)

19. Ferreira, A.C., Campos, M. L., Tanaka, A. K.: An Architecture for Spatial and Dimensional Analysis Integration. Proc. World Multiconference on Systemics, Cybernetics and Informatics (SCI). (2001)
20. TerraLib official home page, (http://www.terralib.org/documentation.html) (2004)
21. OpenGIS. Geography Markup Language Implementation Specification, version 2.1.2. (2002)
22. V. Harinarayan, A. Rajaraman, J. Ullman. Implementing Data Cubes Efficiently. Proc. ACM SIGMOD Int. Conf. on Management of Data. (1996)
23. Guttman, A.: R-Tree: A dynamic Index Structure for Spatial Searching. Proc. ACM SIGMOD Int. Conf. on Management of Data. (1984)
24. Microsoft SQL Server official home page (http://www.microsoft.com/sql/) (2004)
25. Deegree Web Feature Service official home page (http://deegree.sourceforge.net/) (2004)

Workload-Based Placement and Join Processing in Node-Partitioned Data Warehouses

Pedro Furtado

Centro de Informática e Sistemas
Engenharia Informática, Univ Coimbra (DEI /CISUC)
pnf@dei.uc.pt

Abstract. Data warehouses (DW) with enormous quantities of data put major performance and scalability challenges. The Node-Partitioned Data Warehouse (NPDW) divides the DW into cheap computer nodes for scalability. Partitioning and data placement strategies are relevant to the performance of complex queries on the NPDW. In this paper we propose a partitioning placement and join processing strategy to boost the performance of costly joins in NPDW, compare alternative strategies using the performance evaluation benchmark TPC-H and draw conclusions.

1 Introduction

Data warehouses (DW) are large repositories of historical data used mainly for analysis. The applications that allow users to analyze the data (e.g. iterative exploration) have to deal with Giga or Terabytes of data, while users require fast answers. As return-on-investment considerations become crucial, it is desirable that scalability be obtained dynamically and at reasonable cost without throwing away low cost platforms.

Over the past there has been an large amount of work on related and intertwined issues of data allocation [1, 4, 6, 11, 16] and query processing strategies in parallel and distributed databases [3, 8, 9, 13]. Data allocation in parallel and distributed databases has been studied extensively, typically in heavy transactional environments. In an early work [10] compares full partitioning with clustering the relations on a single disk, concluding that partitioning is consistently better for multi-user workloads but can lead to serious performance overhead on complex queries involving joins, with high communication overheads. [4] proposes a solution to data placement using variable partitioning. In that work the degree of partitioning (nº of nodes over which to fragment a relation) is a function of the size and access frequency of the relation. Again, experimental results in [4] show that partitioning increases throughput for short transactions but complex transactions involving several large joins result in reduced throughput with increased partitioning.

Heavy query environments such as a data warehouse with complex query patterns can exhibit such a high communication overhead, as complex joins are executed over several relations. Whether the data warehouse is a set of "pure" star sub-schemas, with a set of dimensions describing business elements and very large facts with business-related measures, or a less structured warehouse schema such as TPC-H [17], the common features are some very big relations, typically facts, with tens to hundreds of

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 38–47, 2004.

million rows and heavy join activity over those big relations and other smaller ones. This is exactly the kind of environment where speedup and scale-up can be hampered by heavy communication costs. We focus on join processing costs in this paper.

One of the crucial issues with partitioning in the presence of costly joins, which we review more profoundly in the next section, is the need to exchange significant amounts of data between nodes, which is especially costly given slow interconnects. Parallel hash-join algorithms [5, 7] (PHJ) provide an efficient way to process heavy joins in such environments. Data placement algorithms on the other hand typically do not bother with the join strategy and its relationship with the query workload.

In this paper we propose the use of workload-based placement and PHJ processing of joins in the specialized structure and context of the data warehouse to speedup complex joins. We show why the use of PHJ is relevant in the NPDW and take into consideration workload access patterns in a simple but new way when determining data placement or reorganization of the data in NPDW.

Related work on partitioning and processing strategies includes other join processing strategies such as PRS [15] or placement dependency [3, 9]. PRS partitions one relation and replicates the other ones and placement dependency co-locates relation partitions exhibiting dependency [3, 9]. Once again, these strategies assume data placement to be done independently from join processing. In [18] an automated partitioning and placement strategy is proposed for generic databases and in [11] an automated partitioning and placement strategy is designed for DB2 which uses the query optimizer to evaluate costs of alternative allocations. Our work is related to these ones but focuses on a simple allocation strategy targeted at data warehouses and which does not require any tight integration with the query optimizer as in [11].

The paper is organized as follows: section 2 discusses the partitioning issue. Section 3 describes data placement generically in NPDW. Section 4 discusses relevant processing costs in NPDW. Section 5 discusses our solutions to data placement and processing that result in small overheads. Finally, section 6 analyzes some experimental performance results using the TPC-H decision support benchmark [17]. Section 7 contains concluding remarks.

2 The Partitioning Issue

The basic partitioning problem is well described in [9]. Assuming all queries of the form Q={ target | qualification}, where target is a list of projected attributes and qualification is a list of equi-joined attributes, let a query be $Q=\{R_1.A,R_2.B \mid R_1.A=R_2.A \ \Lambda\ R_2.B=R_3.B\}$ and assume that R_1 and R_2 reside in different nodes shown in Figure 1. Then, not all of the three relations can be partitioned to process the join, since the first join predicate requires that R_2 be partitioned on A and the second join predicate requires that R_2 be partitioned on B. We may choose to partition R_1 and R_2 on A and replicate R_3 or to partition R_2 and R_3 on B and replicate R_1. The problem is then to decide which set of relations to partition.

Relation	Tuples	Site 1	Site 2
R1	12000	R1	
R2	10000		R2

Fig. 1. Example Layout of R1 and R2

The partition and replicate strategy (PRS) [15] involves partitioning R_1 into two fragments F_{11}, F_{12} and sending F_{12} to site 2. Relation R_2 is sent to site 1 and then each site processes its part of the join: $R_1 \text{ x}_A R_2 = (F_{11} \text{ x}_A R_2) \cup (F_{12} \text{ x}_A R_2)$ processed in parallel in sites 1 and 2, where x_A denotes an equi-join on attribute A.

Using hash-partitioning [5, 8, 12], it is possible to obtain a more efficient result. We first define hash-partitioning.

Definition 1. *A relation Ri is hash-partitioned on attribute A into d disjoint fragments* $\{F_{ij}\}$ *if 1)* $R_i = \cup F_{ij}$*; 2)* $F_{ij} \cap F_{ik} = \emptyset$ *for* $j \neq k$*; and 3)* $\forall T \in F_{ij}$*:* $h(T.A) = c_{ij}$*, where h() is a hash function and* c_{ij} *is a constant for a given j.*

Hash partitioning allows the expression $R_1 \text{ x}_A R_2$ to be processed faster as $(F_{11} \text{ x}_A F_{21}) \cup (F_{12} \text{ x}_A F_{22})$, as fragments are results of hashing the relations such that $F_{11} \text{ x}_A F_{22} = \emptyset$ and $F_{12} \text{ x}_A F_{21} = \emptyset$. The join cost at each site is now smaller as only a fragment of R_2 is joined instead of the whole relation. Communication costs are also reduced, as only a fragment instead of the whole relation R2 must be moved. On the other hand, hash-partitioning R_1 and R_2 introduces some overhead. However, if a query joins R_1 to R_2 by another join attribute or one of the relations participates in a join with other relation on a different attribute, it is necessary to repartition the fragments and redistribute them before this other join can take place. For instance, R1 can be repartitioned by repartitioning fragments F_{11} and F_{21} in parallel and exchanging data to build F'_{11} and F'_{21} in nodes 1 and 2 respectively. Although repartitioning can occur in parallel at all nodes, it entails processing and data communication costs.

Unbalanced placement and processing load is a concern with hash-partitioning which can be minimized by using a suitable hash function, managing and allocating a number of fragments larger than the number of nodes for better load balancing. With this concern taken into consideration, hash-partitioning can be advantageous. Our objective is to minimize extra communication overheads due to partitioning or repartitioning needs as well as join processing costs. Join processing cost is minimized by joining only fragments from all except small relations. Communication overhead is minimized in the above example when F_{11}, F_{21} are initially located at node 1 and F_{12}, F_{22} are initially located at node 2, as the join can proceed without any (re)partitioning and communication costs (except to merge partial results). It is therefore important to determine the best hash-partitioned placement in the presence of a query workload.

3 NPDW Placement Basics

A Node Partitioned Data Warehouse (NPDW) is a data warehouse divided into a set of nodes and a query processing middle layer that uses those nodes for fast query processing. In order to process joins efficiently, we propose hash-partitioning every large relation and replicating small ones. If we consider a strict star schema with only small dimensions and a single large fact, Figure 2 shows the resulting data placement. Join processing in the system of Figure 2 is similar to the PRS strategy, but with relations already replicated at the processing nodes. As a result, all nodes can process independently and return their partial answer for a posterior merge step to obtain the final answer.

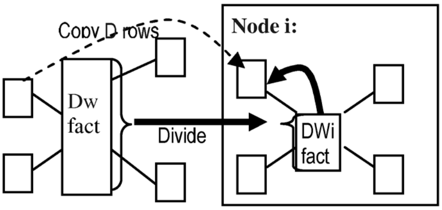


Fig. 2. Basic Star-Partitioning

Actual schemas are typically more complex than a pure star, resulting in not so independent node processing of partial queries and higher repartitioning and communication costs. Schemas can have several stars with cross-querying, possibly larger dimensions and possibly a less pure star design. Joins with large dimensions and with more than one fact are frequent. The most important concern in such an environment is how to partition facts and large dimensions horizontally to solve the partitioning issue described in section 2 as efficiently as possible. More generically, the concern is how to handle data placement and join processing for larger relations and therefore minimize repartitioning and communication costs?

TPC-H [17] provides an example schema to better illustrate our rationale. Figure 3 summarizes the schema, which represents ordering and selling activity (LI-lineitem, O-orders, PS-partsupp, P-part, S-supplier, C-customer).

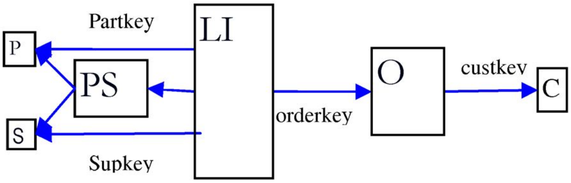


Fig. 3. TPC-H schema

For simplicity, we will portray relations PS, LI and O as facts and the remaining ones (P, S, C) as dimensions. For simplicity also we will assume that dimensions P, S and C are sufficiently small to deserve being replicated into each node. On the other hand, relations LI, PS and O should be horizontally partitioned into the nodes using a strategy that minimizes data exchange requirements while processing joins involving them. Joins involving only one of the horizontally-partitioned relations and any number of replicated relations require no data exchange, regardless of the partitioning scheme used. Therefore, we concentrate on minimizing data exchange needs for joins involving more than one partitioned relation (LI, O, PS) besides dimensions.

4 Relevant Costs in the NPDW

The main processing costs are the partitioning, repartitioning, data communication and local processing costs. Partition (PC) and Repartitioning Costs (PC) are related to scanning the relation or a node-fragment only once. They are monotonically increasing on the relation size. Since there can be two or more relations to be partitioned and they can be processed in parallel in two or more nodes, the partition delay PC for a given query is the largest partition cost among the nodes participating simultaneously. Repartitioning is similar to partitioning, but involves a fragment in each node instead of the whole relation. It is used to re-organize the partitioned relation, hashing on a different equi-join attribute. The fragments resulting from this repartitioning need to be redistributed to other nodes to process a hash-join. Data Communication Costs (DC) are also monotonically increasing with the size of the data transferred and equal between any number of nodes. We assume a switched network (and broadcast capability). Local Processing Costs (LC) are also a function of the relation size and depend on access paths and the size of relations participating in the join. Merging Costs (MC) are related to applying a final query to collected partial results at merging node.

As in [13] we define weighting parameters: β, which is a partitioning cost weight and α local processing weight, so that β/α denotes the ratio of partitioning cost to local processing cost (e.g. ~2 in [14]). Considering facts with size F_i, N nodes and such linear cost model, we can obtain a simple expression for the cost or repartitioning versus the cost of local processing without requiring repartitioning when the placement favors local processing. For simplicity, the following expressions consider only two facts. The fact fragment size is F_i/N. The join-processing cost for queries requiring the join between equi-partitioned facts dimensions di is:

$$\text{Cost}_{\text{equipart}} = \alpha \times \left(\frac{F_1}{N} + \frac{F_2}{N} + d_1 + \ldots + d_l \right) \quad (1)$$

The cost when facts are not equi-partitioned on a switched network is:

$$\text{C}_{\text{Rep}} = \left(\frac{IR}{N} - \frac{IR}{N^2} \right) \times \beta + \alpha \times \left(\frac{F_1}{N} + \frac{F_2}{N} + d_1 + \ldots + d_l \right) \quad (2)$$

This expression includes the repartitioning overhead of an intermediate result IR (either the fact or the result of joining with some dimensions locally before repartitioning) and the local processing cost of 1/N of the facts. The increase in cost if (2) is required instead of (1) is therefore $\left(\frac{IR}{N} - \frac{IR}{N^2} \right) \times \beta$, which can be avoided many times if the facts are carefully placed.

5 Placement and Processing in NPDW

Under NPDW all nodes participate in the computation of every query, so that typical queries must be "broken" into subset queries to distribute into all nodes and merging components to merge result sets coming from nodes. We define Global Processing as a processing "cycle" that involves sending data to nodes, processing in each node, collecting results from those nodes and merging them. Local Processing is the independent processing in a node. Query transformation produce processing and merging query components.

The global processing DCCM - Distribute-Compute-Collect-Merge – shown in Figure 4 involves parsing the query, transforming it into a "local query" to be processed in each node, running the local query independently in each node, collecting the corresponding partial results in the submitting node and merging them.

Set operators, typically UNION ALL, are useful for merging partial results from all nodes. Selection and aggregation query row sets will typically be processed independently by nodes and the partial results are merged as the union of partial result sets, followed by aggregation in the case of aggregation queries.

With this strategy, most query operations (selections, projections, aggregations) can be processed independently by each node, followed by a merge phase that receives independent results and merges them into a final result. Some operations, including most subqueries, require additional DCCM steps. In this paper we are concerned with the placement and processing of joins in the context of NPDW.

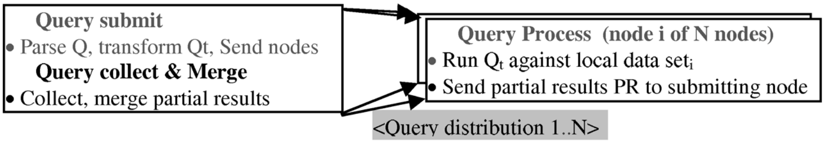

Query submit
• Parse Q, transform Qt, Send nodes
Query collect & Merge
• Collect, merge partial results
Query Process (node i of N nodes)
• Run Q_t against local data set_i
• Send partial results PR to submitting node
<Query distribution 1..N>

Fig. 4. The DCCM Procedure

5.1 Partitioning and Placement in NPDW

In order to determine the most appropriate partitioning strategy for a schema, knowledge of the query workload is required. In our example and experiments with TPC-H we considered the query set with equal weight for each query. In a practical situation knowledge of the query set would be accompanied by expected or historic frequency of execution of each query. More generically, it should be possible to know the query set and expected workload for a given data warehouse.

From the query workload we can determine a query workload join graph. A join graph $G_J = (V_J, E_J)$ is a graph where vertices V_J correspond to attributes R.A participating in equi-joins and the edges E_J depict the set of equi-joins between those attributes. We add weights to the edges of the join graph, which correspond to the number of occurrences of the corresponding equi-join in the query workload. A component is a set of connected edges in the join graph. Figure 5 shows the weighted join graph for TPC-H. In this join graph, replicated relations (P, S, C) are not represented because they pose no repartitioning issues for processing joins. Only those relations that are partitioned must be represented. Relation sizes are also relevant to the algorithm.

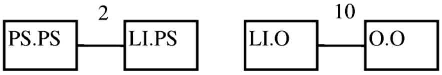


Fig. 5. Weighted Join Graph for TPC-H

This join graph can be produced either manually or automatically, as long as replicated relations are well identified. The next step is to decide which partitioning attributes should be chosen for each relation. The choice is based on the relation sizes and frequency values of the join graph. The most frequent equi-join component including the largest relation has top priority. In Figure 3 it is clear that LI and O (Orders) should be partitioned by orderkey (O), so that joins between these two relations do not require any repartitioning. But if LI is partitioned by orderkey (O), the other, less frequent equi-join in Figure 3 (LI with PS) will require repartitioning of LI. More generically, the algorithm is:

1. *For the largest relation, find largest frequency component in the join graph;*
2. *Partition intervening relations by the corresponding equi-join attribute;*
3. *Delete partitioned relations from all edges in all components of the join graph, as their partitioning attribute is already decided;*
4. *If done, stop. Otherwise, go to 1.*

This strategy can be effectively applied automatically to any schema and query set. It implies simply some knowledge of the query workload. In this paper we use TPC-H to exemplify the use of such strategy.

5.2 Join Processing in NPDW

Given the placement of relations, join processing is based on cost minimization and uses parallel hash-join when required, as discussed in [13]. The focus is on minimizing the repartitioning and communication costs. The complexity and relevance of the join processing strategy in such a system has led us to treat the subject in depth elsewhere. However, we describe the basic strategy in this section using examples.

Intuitively, the most selective joins should be processed first. For illustration purposes we do not take into account different selectivities in the following examples.

Assuming that LI and O are pre-partitioned by orderkey, a join involving relations LI and O (and any other number of replicated dimensions) in the TPC-H schema of Figure 3 is transformed into the union of the independent node joins $\cup_{\text{all nodes}}$ (F_{LI} x_o F_O), where F_X represents a fragment of X.

A more complex join involving all partitioned relations and dimensions P, S and C of Figure 3 can be processed as (P x_P (S x_S F_{PS})) x_{PS} (F_{LI} x_o (F_O x_c C)), based in processing joins with the smallest relations first. This processing path requires a single repartitioning. As P, S and C are replicated in all nodes, each binary join involving them can be processed independently in each node, which accounts for (F_O x_c C) and (P x_P (S x_S F_{PS})). Then, (F_{LI} x_o (F_O x_c C)) can also be processed independently without repartitioning because the placement algorithm co-located O and LI fragments by hash-partitioning on orderkey (o). Finally, the last join requires repartitioning of the intermediate result (F_{LI} x_o (F_O x_c C)) by the (ps) key to join with (P x_P (S x_S F_{PS})).

6 Experiments

In this section we compare the performance of alternative placement and processing strategies in NPDW using typical TPC-H queries over the schema of Figure 3. For all of them, we considered a setup in which the smaller relations P, S, C were replicated. The alternatives evaluated are:

HWP (Best Strategy): The Hash-based workload partitioning strategy. It involves placement based on join-wise workload-based partitioning of large relations and join processing based on PHJ. The only extra overhead comes from repartitioning when needed. The placement algorithm tries to minimize such need;

H: Parallel Hash-Join with hash-partitioned placement based on primary keys. It involves placement based on primary-key partitioning and join processing based on PHJ. The extra overhead comes from repartitioning when the primary-key is not the equi-join attribute between partitioned relations. The objective is to compare the performance of placement with workload knowledge (HWP) to "blind" placement (H);

PRS: The PRS strategy [15] considering that only the largest relation (LI) is pre-partitioned by placement. This strategy involves replicating intermediate results from all but one of the large relations into all nodes (broadcast). The major overheads are replication and processing costs as each node has to process un-fragmented intermediate results for all except the pre-partitioned relation;

Round-Robin PRS (RR-PRS): This strategy involves placement by round-robin partitioning of facts. Join processing is similar to PRS;

Round-Robin Hash (RR-H): This strategy involves placement by round-robin partitioning of facts. Join processing with more than one partitioned fact is based on PHJ but involves repartitioning overhead for all partitioned facts.

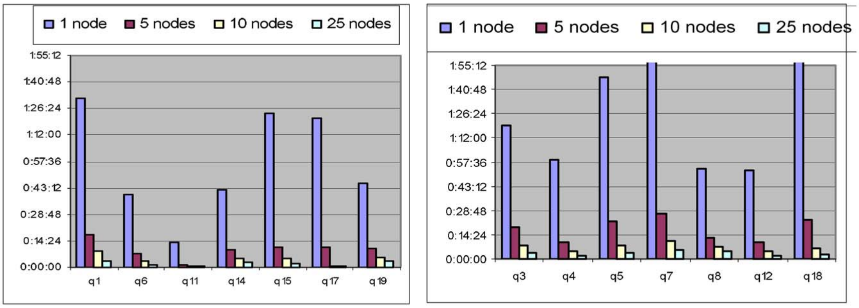
The differences between the strategies were apparent in the experimental results when processing queries with multiple partitioned relations. PRS and RR-PRS strategies typically incurred high replication and join processing costs in those cases; The RR-H strategy incurred repartitioning overhead for all partitioned relations; The H strategy incurred repartitioning overhead for some of the intermediate results (joins including the LI relation); HWP had the best results because it uses PHJ as H and RR-H but chooses an initial workload-dependent placement that minimizes repartitioning needs. Parallel hash-join algorithm is more efficient as it incurs much smaller overheads.

The experimental evaluation was conducted by measuring the node and communication costs for the strategies assuming similar nodes on a 100Mbps switched network and the TPC-H decision support benchmark™ [17]. Four data sets were generated for these experiments corresponding to a 50GB TPC-H generation into 1, 5, 10, 25 nodes. Our experiments also involved transforming TPC-H queries to run against the NPDW.

6.1 Performance Results

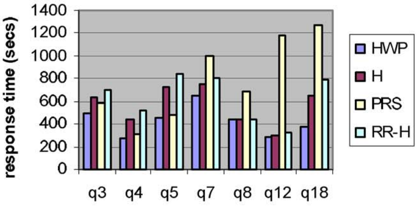
First of all we analyze queries accessing a single fact relation, as for these queries all but one relation (the fact) is replicated to all nodes and all the strategies have similar results. Such queries account for half the TPC-H query set. Figure 6a shows the response time results for some of these queries with a nearly linear speedup with the number of nodes. Another large fraction of the TPC-H queries involve joining relations LI and O besides the replicated dimensions. Figure 6b shows the response times of HWP for some of those queries. Once again the speedup is near to linear, as only 1/N fragment of the facts has to be processed in each of N nodes.

Figure 7a shows response time results for these queries comparing HWP, H, RR and PRS for 10 nodes. As can be seen from the Figure, the relative performance of the strategies varies with the query. This is for two main reasons: the extra repartitioning and processing overhead of the strategies centers on different relations (H on LI, PRS on O, RR on LI and O) and we applied local joins and restrictions (selection operators) on the relations and dimensions before exchanging intermediate results between nodes in order to minimize extra overheads. In spite of this, the conclusion is that HWP was consistently better: on average it was 25% better than H and 46% better than PRS (and RR-PRS) for these queries. The only cases in which the difference is smaller is when the slower strategies were able to obtain a very small intermediate result locally before incurring the large exchange and processing overhead. HWP is also better than the “blind” hash strategy (H) or RR-H, which require repartitioning before processing the join.



(a) Single Fact (b) Access to LI/O

Fig. 6. Response Time Single Fact Queries



10 NODES	HWP	H	PRS
Q9	426	500	2173
Q20	235.5	378	2023

(a) Queries LI/O (b) Queries Q9 and Q20

Fig. 7. Response Time Comparison, Queries LI/O

Finally, queries joining more than two facts using different join attributes require repartitioning in HWP as well as H and replication of all but one fact in PRS/RR-PRS. TPC-H contains two such queries: Query Q20 joining Lineitem to Partsupp by ps and query Q9 joining all three relations Orders, Lineitem and Partsupp. Figure 7b shows the response times for these queries (secs). Our strategy was on average 26% better than H and 84% better than PRS/RR-PRS for these queries.

The PRS/RR-PRS overhead increases quickly with the number of large relations requiring replication not only due to larger communication overhead but also due to much larger node processing overheads.

7 Conclusions

We have proposed and studied experimentally a partitioning, placement and join processing solution to process join queries efficiently in a node-partitioned data warehouse. The strategy is workload-based. Comparative evaluation against TPC-H has shown the advantages of the strategy and alternatives.

References

1. Apers, P. M. G.. Data allocation in distributed database systems. ACM Transactions on Database Systems, 13(3):263--304, September 1988.
2. P.A. Bernstein et al., "Query Processing in a System for Distributed Databases (SDD-l)," ACM Trans. DB Sys., vol. 6, no. 4, pp. 602-625, Dec. 1981.
3. Chen, Hao, Chengwen Liu: An Efficient Algorithm for Processing Distributed Queries Using Partition Dependency. Int'l Conf. on Par. and Distr. Sys., ICPADS 2000: 339-346.
4. Copeland G. P., William Alexander, Ellen E. Boughter, Tom W. Keller: Data Placement In Bubba. SIGMOD Conference 1988: 99-108.
5. David J. DeWitt, Robert Gerber, Multiprocessor Hash-Based Join Algorithms, Proceedings of the 11th Conference on Very Large Databases, Morgan Kaufman pubs. Stockholm.
6. Hua, K. A. and Lee, C., "An Adaptive Data Placement Scheme for Parallel Database Computer Systems" Proc. VLDB Conf., Brisbane, Australia, 1990.
7. M. Kitsuregawa, H. Tanaka, and T. Motooka. Application of hash to database machine and its architecture. New Generation Computing, 1(1):66-74, 1983.
8. Liu, Chengwen and Hao Chen, "A Hash Partition Strategy for Distributed Query Processing", International Conference on Extending Database Technology (EDBT) 1996.
9. Liu, Chengwen, Hao Chen, Warren Krueger, "A Distributed Query Processing Strategy Using Placement Dependency", Proc. 12th Int'l Conf. on Data Eng, pp. 477-484, Feb. 1996.
10. Livny, Miron, Setrag Khoshafian, Haran Boral: "Multi-Disk Management Algorithms". In Procs. Of ACM SIGMETRICS 1987, pp69-77.
11. Rao, Jun. , Chun Zhang, Nimrod Megiddo, Guy M. Lohman: Automating physical database design in a parallel database. SIGMOD Conference 2002: 558-569.
12. Sacca, D. and Wiederhold, G.: Database Partitioning in a Cluster of Processors. ACM TODS, Vol. 10, No. 1, pp. 29-56, Mar 1985.
13. Shasha, Dennis and Wang, Tsong-Li: Optimizing Equijoin Queries In Distributed Databases Where Relations Are Hash Partitioned. ACM Transactions on Database System, Vol. 16, No. 2, pp. 279-308, June 1991.
14. Teradata Corporation. Database Computer System Concepts and Facilities. Document C02-0001-01, Teradata Corporation, Los Angeles, Oct. 1984.
15. Yu, C., Guh, K., Brill, D., Chen, A.: Partition strategy for distributed query processing in fast local networks. IEEE Trans. on Software Eng., Vol. 15, No. 6, pp. 780-793, June 1989.
16. Zhou S., M.H. Williams, "Data placement in parallel database systems," Parallel Database Techniques, IEEE Computer Society Press, 1997.
17. Transaction Processing Council Benchmarks, www.tpc.org.
18. Zilio, Daniel C., Anant Jhingran, Sriram Padmanabhan, Partitioning Key Selection for a Shared-Nothing Parallel Database System *IBM Research Report RC 19820 (87739) 11/10/94* ,T. J. Watson Research Center, Yorktown Heights, NY, October 1994

Novelty Framework for Knowledge Discovery in Databases

Ahmed Sultan Al-Hegami, Vasudha Bhatnagar, and Naveen Kumar

Department of Computer Science
University of Delhi
Delhi-07, India
{ahmed,vb,nk}@csdu.org

Abstract. Knowledge Discovery in Databases (KDD) is an iterative process that aims at extracting *interesting*, previously unknown and hidden patterns from huge databases. Use of objective measures of interestingness in popular data mining algorithms often leads to another data mining problem, although of reduced complexity. The reduction in the volume of the discovered rules is desirable in order to improve the efficiency of the overall KDD process. Subjective measures of interestingness are required to achieve this. In this paper we study *novelty* of the discovered rules as a subjective measure of interestingness. We propose a framework to quantify *novelty* of the discovered rules in terms of their deviations from the known rules. The computations are carried out using the importance that the user gives to different deviations. The computed degree of novelty is then compared with the user given threshold to report *novel* rules to the user. We implement the proposed framework and experiment with some public datasets. The experimental results are quite promising.

1 Introduction

Knowledge discovery in databases (KDD) is a process of extracting previously unknown, hidden, novel and interesting knowledge from massive volumes of data stored in databases [1,16,17]. It is an iterative process carried out in three stages. The KDD process begins with the understanding of a problem and ends with the analysis and evaluation of the results. It includes preprocessing of the data (*Data Preparation* stage), extracting information from the data (*Mining* stage), and analyzing the discovered knowledge (*Analysis* stage) [1,17].

Actual extraction of patterns is preceded by preliminary analysis of data, followed by selection of relevant horizontal or vertical subset and appropriate data transformations. This is the preprocessing stage of KDD and it is considered to be the most time-consuming stage [2]. Often, the preparation of the data is influenced by the extraction algorithms used during the mining (second) stage. Data mining algorithms are applied during the second stage of the KDD process, which is considered to be the core stage. It involves selection and application of appropriate mining algorithm to search for patterns in the data. Sometimes

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 48–57, 2004.

combination of mining algorithms may be required to extract interesting patterns from the pre-processed data [3, 18]. The outcome of this stage is the discovery of models/patterns hidden in databases, which are interpreted and analyzed during the third stage. The final stage of KDD process is analysis and evaluation of the knowledge discovered in the second stage. Having obtained patterns/models is not the end of the KDD process. Evaluation and analysis is equally important (if not more), particularly in view of the proliferation of KDD techniques being used to solve real-life applications.

It is common knowledge that the volume of patterns discovered from data mining algorithms becomes huge due to the large size of target database [4, 5, 7, 9, 10]. Identifying interesting patterns from the vast set of discovered patterns still remains fundamentally a mining problem, though of reduced complexity. Time required to generate rules, space required to store, maintain and understand the rules by end users are some of the practical issues that need attention.

The problem of reducing the volume of the discovered knowledge has been attacked at all the three stages of KDD process. Psaila proposed analysis of data to identify meta-patterns during pre-processing stage [19]. During the data mining stage, researchers commonly use either constraints or appropriate measures of interestingness to reduce the number of discovered rules. The third stage of the KDD process, which aims at analysis and interpreting the discovered knowledge is carried out by the end user. Post analysis of the discovered patterns as proposed in [4, 6, 9] aids the user to focus on a small subset of discovered patterns. On account of wide variation of users' need and their subjectivity, the end users design and develop need based filters in an adhoc manner.

1.1 Motivation for the Present Work

The approach in [4, 6, 9] assumes that the user is familiar with the domain and has a fair idea ("general impression") of the trends/patterns prevalent in the domain. In case the assumption fails (end user is new to the domain), the user is deluged with the massive volume of discovered patterns, possibly leading to errors in decision-making.

Our work pertains to the third stage of the KDD process. We propose a self-upgrading filter, that captures and quantifies novelty of the discovered knowledge with respect to (user specified) domain knowledge and previously discovered knowledge. The domain-innocent user is aided by the filter and is always returned "novel" rules, as per the specified threshold. The threshold can be dynamically varied to suit the needs of users of different experience levels. The proposed filter quantifies novelty of the discovered knowledge on the basis of deviation of the newly discovered rules with respect to the known knowledge. We consider unexpectedness as implicit novelty.

The paper is organized as follows: In section 2, we review the literature related to the discovery of "novel" knowledge. Section 3 presents a generic architecture of a post-processing filter and sets the tone for the next section. Section 4 presents our approach to quantify the novelty. Section 5 presents the effectiveness of the proposed framework through experiments on real-life datasets. Finally, section 6 concludes the paper.

2 Related Works

There are many proposals that study novelty in disciplines such as robotics, machine learning and statistical outliers detection [11–14]. Generally, these methods build a model of a training set that is selected to contain no examples of the important (i.e. novel) class. Then, the model built; detect the deviation from this model by some way. For instance, Kohonen and Oja proposed a novelty filter, which is based on computing the bit-wise difference between the current input and the closest match in the training set [11]. In [12] a sample application of applying association rule learning is presented. By monitoring the variance of the confidence of particular rules inferred from the association rule learning on training data, it provides information on the difference of such parameters before and after the testing data entering the system. Hence, with some pre-defined threshold, abnormalities can be fairly detected.

In [14], novelty is estimated based on the lexical knowledge in WordNet. The proposed approach defines a measure of semantic distance between two words in WordNet and determined by the length of the shortest path between the two words (w_i, w_j). The novelty then is defined as the average of this distance across all pairs of the words (w_i, w_j), where w_i is a word in the antecedent and w_j is a word in the consequent.

The techniques that have been proposed in statistical literature are focused on modeling the support of the dataset and then detecting inputs that don't belong to that support. The choice of whether to use statistical methods or machine learning methods is based on the data, the application, and the domain knowledge [13].

3 Novelty Framework

The notion of *novelty* of the knowledge indicates the extent to which the discovered rules contribute to new knowledge for a user. It is purely subjective and encompasses the unexpectedness of discovered model with respect to the known knowledge (domain knowledge (DK) and previously discovered knowledge (PDK)). The newness of knowledge may vary not only from user to user but also from time to time for the same user. This makes it necessary to quantify the degree of newness at time t for user u. Figure 1 shows a generic architecture for the proposed framework.

At time t_i, database D_i is subjected to the mining algorithm, resulting into discovery of knowledge K_i. The Novelty filter process K_i, in light of knowledge that the user already has and the previously discovered knowledge i.e. *known knowledge*[1]. The filter can be designed based on either syntactic or semantic

[1] This necessitates storage of DK and PDK in rule base before any matching techniques can be used.

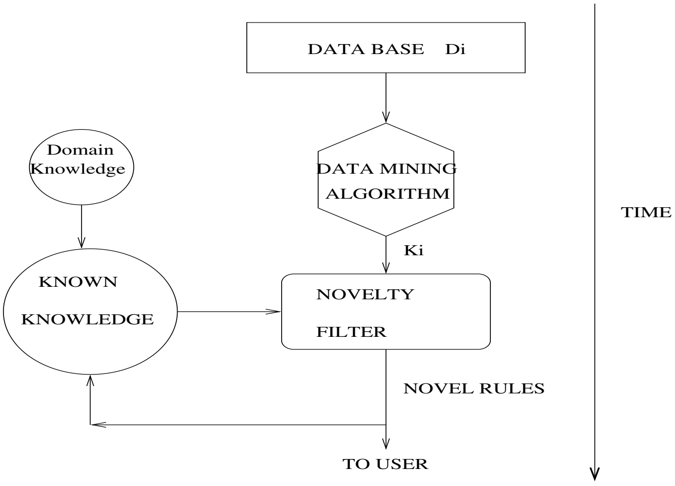


Fig. 1. Novelty Framework Architecture

matching techniques. In this work, we focus on syntactic techniques and leave semantic techniques for future work.

To evaluate the novelty of a rule, we have to measure the deviation at conjunct level first. When the deviation of conjuncts of a rule has been detected, it can be generalized to rule level. The following subsections describe the approach for computing these two types of deviation.

3.1 Deviation at Conjunct Level

Deviation degree of a conjunct is determined based on the result of a matching comparison of the conjuncts of currently discovered knowledge (CDK) against those conjuncts in DK/PDK. All conjuncts in DK/PDK/CDK should have the same knowledge representation and the rules must be comparable. A rule is comparable if it has at least one conjunct contains an attribute that exists in DK/PDK and the consequents are same. When comparison, the attributes, operators, and attribute values of CDK are compared against attributes, operators, and attribute values of DK and PDK respectively.

We suggest five possible degrees of deviation for the conjuncts as shown in definition 1. We believe that, these situations of deviation degree are sufficient enough to define the deviation at conjunct level. These degrees vary from deviation $degree = 0$ which indicates no deviation exist between any two conjuncts, and deviation $degree = \gamma_4$ which indicates higher deviation which means that the attributes tests are not matching at all, and therefore, implies new conjunct.

3.2 Deviation at Rule Level

The deviation of a rule can be estimated by generalizing the deviation at conjuncts level. The rule must be comparable before any matching can be conducted.

A rule is comparable with another rule if they have the same consequent; otherwise, the discovered rule is having highest degree of deviation. A rule deviation is calculated as a linear combination of deviation of the set of conjuncts of the rules in DK/PDK. After rule deviation is computed, we have to decide either the rule is novel or simply a deviation of an existing rule. Whether a rule is novel or not depends on the user feeling about the domain, which is determined by a certain threshold value.

4 Quantification of Novelty

A rule $\Re$ has the form: $\dot{A} \rightarrow \dot{C}$ where $\dot{A}$ denotes an antecedent and $\dot{C}$ denotes a consequent. Both $\dot{A}$ And $\dot{C}$ are in CNF $(c_1 \cap c_2 \ldots c_k)$. The conjuncts c_j are of the form $\dot{A}_1\dot{O}_1\dot{V}_1$, where $\dot{A}$ is an attribute, Dom$(\dot{A})$ is the domain of $\dot{A}$, and $\dot{V} \in$ Dom$(\dot{A})$, $\dot{O} \in \{=, <, >, \leq, \geq\}$. We consider both $\dot{A}$ and $\dot{C}$ as sets of conjuncts in our framework. In this section we give definitions, which are used subsequently in the proposed framework. We consider two conjuncts comparable if and only if they involve the same attributes.

Definition 1. *Let c_1 and c_2 be two conjuncts $\dot{A}_1\dot{O}_1\dot{V}_1$ and $\dot{A}_2\dot{O}_2\dot{V}_2$ respectively. The deviation of c_1 with respect to c_2 is defined as follows:*

$$\triangle(c_1, c_2) = \left\{ \begin{array}{l} 0 \;\; \textit{if}\; \dot{A}_1 = \dot{A}_2, \dot{O}_1 = \dot{O}_2, \;\; \textit{and}\; \dot{V}_1 = \dot{V}_2 \\ \gamma_1 \;\textit{if}\; \dot{A}_1 = \dot{A}_2, \dot{O}_1 = \dot{O}_2, \;\; \textit{and}\; \dot{V}_1 \neq \dot{V}_2 \\ \gamma_2 \;\textit{if}\; \dot{A}_1 = \dot{A}_2, \dot{O}_1 \neq \dot{O}_2, \;\; \textit{and}\; \dot{V}_1 = \dot{V}_2 \\ \gamma_3 \;\textit{if}\; \dot{A}_1 = \dot{A}_2, \dot{O}_1 \neq \dot{O}_2, \;\; \textit{and}\; \dot{V}_1 \neq \dot{V}_2 \\ \gamma_4 \qquad\qquad\qquad \textit{otherwise} \end{array} \right\} \tag{1}$$

$\gamma_1, \gamma_2, \gamma_3$, and γ_4 are user specified numeric quantities, such that $0 \leq \gamma_1 \leq \gamma_2 \leq \gamma_3 \leq \gamma_4 \leq 1$

Definition 2. *Let S_1 and S_2 be two sets of conjuncts. We define a matching function $\psi(S_1, S_2)$ as follows:*

$$\psi(S_1, S_2) = \left\{ \begin{array}{lc} 0 \;\textit{iff}\; |S_1| = |S_2|, \forall c_i \in S_1, \exists c_j \in S_2, \textit{such that} \; \triangle\,(c_i, c_j) = 0 \\ 1 \qquad\quad \forall c_i \in S_1, \exists c_j \in S_2 \;\textit{,such that}\; \triangle\,(c_i, c_j) < \gamma_4 \\ \beta \qquad\qquad\qquad\qquad \textit{otherwise}, 0 \leq \beta \leq 1 \end{array} \right\} \tag{2}$$

where $\beta = \frac{1}{|S_1|} \sum_{c_i \in S_1} \sum_{c_j \in S_2} \text{Min} \triangle\,(c_i, c_j)$

As per definition 2, $\psi(S_1, S_2) = 0$ indicates that S_1 and S_2 are identical, $\psi(S_1, S_2) = 1$ indicates the extreme deviation and computed value of β quantifies an intermediate degree of deviation.

Definition 3. *Let $\Re_1 : \dot{A}_1 \rightarrow \dot{C}_1$ and $\Re_2 : \dot{A}_2 \rightarrow \dot{C}_2$ be two rules. We say that $\Re_1$ is comparable with respect to $\Re_2$, if :*

$$\psi(\dot{C}_1, \dot{C}_2\} = 0 \tag{3}$$

Definition 4. *Let $\Re_1 : \dot{A}_1 \rightarrow \dot{C}$ and $\Re_2 : \dot{A}_2 \rightarrow \dot{C}$ be two comparable rules. We say that $\Re_1$ is generalization of $\Re_2$, denoted by G($\Re_1, \Re_2$) if :*

$$G(\Re_1, \Re_2) = \left\{ \begin{array}{ll} 1 & \textit{iff}\ \dot{A}_1 \subset \dot{A}_2 \\ 0 & \textit{otherwise} \end{array} \right\} \quad (4)$$

the degree of generalization of $\Re_1$ with respect to $\Re_2$ is defined as:

$$\dot{g} = 1 - \frac{|\dot{A}_1 \cap \dot{A}_2|}{|\dot{A}_2|} \quad (5)$$

Definition 5. *Let $\Re_1 : \dot{A}_1 \rightarrow \dot{C}$ and $\Re_2 : \dot{A}_2 \rightarrow \dot{C}$ be two comparable rules. We say that $\Re_1$ is specialization of $\Re_2$, denoted by S($\Re_1, \Re_2$) if :*

$$S(\Re_1, \Re_2) = \left\{ \begin{array}{ll} 1 & \textit{iff}\ \dot{A}_2 \subset \dot{A}_1 \\ 0 & \textit{otherwise} \end{array} \right\} \quad (6)$$

the degree of specialization of $\Re_1$ with respect to $\Re_2$ is defined as:

$$\dot{s} = 1 - \frac{|\dot{A}_1 \cap \dot{A}_2|}{|\dot{A}_1|} \quad (7)$$

The next definition is the basis of computation of the degree of novelty of rule. We, suggest that a rule's deviation degrees can be classified into five categories, as shown in the following definition.

Definition 6. *Let $\Re_1 : \dot{A}_1 \rightarrow \dot{C}_1$ and $\Re_2 : \dot{A}_2 \rightarrow \dot{C}_2$ be two rules. The deviation of $\Re_1$ with respect to $\Re_2$ denoted by $\triangle(\Re_1, \Re_2)$ is computed as follows:*

$$\triangle(\Re_1, \Re_2) = \left\{ \begin{array}{ll} 0 \ \textit{iff}\ \psi(\dot{A}_1, \dot{A}_2) = 0 \ \textit{and}\ \psi(\dot{C}_1, \dot{C}_2) = 0 & \\ \dot{g} & \textit{iff}\ G(\Re_1, \Re_2) = 1 \\ \dot{s} & \textit{iff}\ S(\Re_1, \Re_2) = 1 \\ 1 & \textit{iff}\ \psi(\dot{C}_1, \dot{C}_2) = 1 \\ \beta & \quad \textit{otherwise} \end{array} \right\} \quad (8)$$

where $0 \leq \dot{g}, \dot{s}, \beta \leq 1$, and β, $\dot{g}$ and $\dot{s}$ are computed as per definitions 2, 4 and 5 respectively.

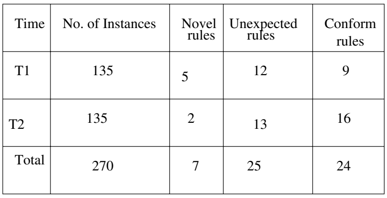
As per definition 6, $\triangle(\Re_1, \Re_2) = 0$ indicates that $\Re_1$ and $\Re_2$ are identical, $\triangle(\Re_1, \Re_2) = 1$ indicates the extreme deviation between $\Re_1$ and $\Re_2$. $\triangle(\Re_1, \Re_2) = \beta$ indicates intermediate degree of deviation of $\Re_1$ with respect to $\Re_2$. $\dot{g}$ and $\dot{s}$ indicate the degree of generalization and specialization respectively as per the case. The user is encouraged to specify the thresholds to sift novel rules based on the computation of $\triangle(\Re_1, \Re_2)$.

5 Implementation and Experimentation

The novelty framework is implemented and tested using real-life datasets. The system is built using c programming language. Since, there is no other approaches available, which handle the novelty, we could not perform any comparison against our framework. We will explain our framework through the following experiments.

5.1 Experiment 1 – The Heart Disease Dataset

The first experiment was run on the Heart Disease Dataset that is created by George John and appears on the UCI ML Data Repository at `http://kdd.ics.uci.edu`. It contains 13 attributes and 270 instances and two classes that determine either a person is or is not having heart disease. We obtain a set of rules to represent a user's domain knowledge. This set of rules has the same form as the rules discovered by CBA rules [15]. We generate rules using CBA classifier with 0.1% and 1% to indicate minimum confidence and minimum support respectively. Further, we partition the dataset into two groups each of which has 135 instances which represent instances at time T_1 and time T_2. CBA discover 26 and 31 rules at time T_1 and T_2 respectively as shown in Fig. 2.



Time	No. of Instances	Novel rules	Unexpected rules	Conform rules
T1	135	5	12	9
T2	135	2	13	16
Total	270	7	25	24

Fig. 2. The discovered knowledge at time T_1 and T_2

For better understanding our framework, let's consider the following set of rules discovered by CBA at time T_1:
$\Re_1 : sex = male \rightarrow yes$
$\Re_1 : Angina \geq 0.5 \cap MaxHeartRate < 150.5 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_2 : Thal \geq 4.5 \cap Angina \geq 0.5 \cap MaxHeartRate < 150.5 \rightarrow 2$
$\Re_3 : Thal < 4.5 \cap STSlope \geq 1.5 \cap MaxHeartRate \geq 150.5 \cap Age < 54.5 \rightarrow 1$
$\Re_4 : Vessels \geq 0.5 \cap MaxHeartRate \geq 150.5 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_5 : OldPeak < 0.85 \cap Angina < 0.5 \cap MaxHeartRate < 150.5 \rightarrow 1$
$\Re_6 : Vessels < 0.5 \cap ChestPain < 3.5 \cap Age < 54.5 \rightarrow 1$
$\Re_7 : OldPeak \geq 0.85 \cap MaxHeartRate < 150.5 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_8 : Vessels < 0.5 \cap STSlope < 1.5 \cap ChestPain < 3.5 \rightarrow 1$
$\Re_9 : Thal \geq 4.5 \cap OldPeak \geq 0.85 \cap MaxHeartRate < 150.5 \rightarrow 2$
The user specified domain knowledge (DK) as follows:
1. $ChestPain < 3.5 \cap Sex < 0.5 \rightarrow 1$
2. $Sex < 0.5 \cap Age < 54.5 \rightarrow 1$
3. $MaxHeartRate < 147 \cap Age < 54.5 \rightarrow 2$
4. $ChestPain \geq 3.5 \cap Sex < 0.5 \rightarrow 2$
Notice that, the number between [ ] indicates the degree of deviation. If the user provides the following threshold values for conforming, generalizing/specializing, unexpected, and novel rules:
$0 \leq Deviation \leq 0.3$ — Conforming rule
$0.3 < Deviation \leq 0.3003/0.3004$ —Generalizing/Specializing rule
$0.3003/0.3004 < Deviation \leq 0.6$ —Unexpected rule

$Deviation > 0.6$ —Novel rule.
Our framework discovered the following set of rules at time T_1:

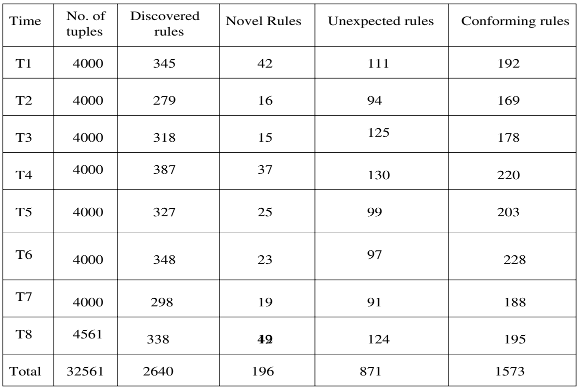
- Novel Rules.
  $\Re_1[1.000]$, $\Re_3[1.000]$, $\Re_9[0.667]$
- Unexpected Rules.
  $\Re_2[0.333]$: This rule is unexpected with respect to $\Re_1$ in PDK with deviation degree 0.33 since its deviation degree with $\Re_3$ in DK is 0.7.
  $\Re_4[0.433]$: This rule is unexpected since its deviation degree with $\Re_1$ in DK is 1 while it is 0.43333 with respect to $\Re_1$ in PDK.
  $\Re_5[0.575]$: It is unexpected since its deviation degree with $\Re_3$ in DK is 1 while it is 0.575 with respect to $\Re_3$ in PDK.
  $\Re_6[0.500]$: This rule is unexpected with respect to $\Re_3$ in PDK with deviation degree 0.50 since its deviation degree with respect to $\Re_1$ or $\Re_2$ in DK is 1.
  $\Re_7[0.333]$: The rule is unexpected with respect to $\Re_1$ in PDK with deviation degree 0.333 since its deviation degree with respect to $\Re_3$ or $\Re_4$ in DK is 1.
  $\Re_8[0.575]$: The rule is unexpected since its deviation degree with $\Re_1$ or $\Re_2$ in DK is 1 while it is 0.575 with respect to $\Re_3$ in PDK.
- Conforming Rules
  Due to the threshold values maintained, no conforming rules found at time T_1.

Furthermore, the following rules are discovered using CBA at time T_2:
$\Re_{10} : Thal \geq 6.5 \cap OldPeak \geq 1.15 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_{11} : Vessels \geq 0.5 \cap STSlope \geq 1.5 \cap OldPeak \geq 1.15 \rightarrow 2$
$\Re_{12} : Thal < 6.5 \cap Vessels < 0.5 \cap OldPeak < 1.15 \cap MaxHeartRate \geq 147.5 \rightarrow 1$
$\Re_{13} : STSlope < 1.5 \cap MaxHeartRate \geq 147.5 \cap ChestPain < 3.5 \rightarrow 1$
$\Re_{14} : Thal < 6.5 \cap Vessels < 0.5 \cap MaxHeartRate \geq 147.5 \cap ChestPain < 3.5 \rightarrow 1$
$\Re_{15} : Vessels \geq 0.5 \cap STSlope \geq 1.5 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_{16} : Thal \geq 6.5 \cap Vessels \geq 0.5 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_{17} : Vessels \geq 0.5 \cap Angina \geq 0.5 \cap ChestPain \geq 3.5 \rightarrow 2$
$\Re_{18} : Thal < 6.5 \cap Vessels < 0.5 \cap OldPeak < 1.15 \cap Angina < 0.5 \rightarrow 1$
$\Re_{19} : Thal < 6.5 \cap OldPeak < 1.15 \cap Angina < 0.5 \cap MaxHeartRate \geq 147.5 \rightarrow 1$
Applying the novelty framework to these discovered rules at time T_2 taking into consideration the same DK and the same threshold values as well as the rules which are already discovered at time T_1, the following type of rules are generated:

- Novel Rules.
  $\Re_{11}[0.700]$, $\Re_{16}[0.667]$, $\Re_{18}[0.775]$
- Unexpected Rules.
  $\Re_{10}[0.400]$, $\Re_{12}[0.550]$, $\Re_{13}[0.350]$,$\Re_{14}[0.550]$, $\Re_{15}[0.333]$, $\Re_{17}[0.333]$
- Conforming Rules.
  $\Re_{19}[0.250]$

5.2 Experiment 2 – The Adult Dataset

The second experiment was run on the census-income database that appears on the UCI ML. Data Repository at http://kdd.ics.uci.edu. This dataset contains 6 continuous, 8 nominal attributes, 48842 instances, mix of continuous and discrete (train=32561, test=16281), and two classes determine either a person's income is $\leq 50K$ or $> 50K$. With the same confidence and support of first experiment, we run CBA against 8 partitions of the dataset representing instances arrive at time $T_1, T_2, T_3, T_4, T_5, T_6, T_7$, and T_8 respectively. Fig. 3 shows the different types of knowledge that are generated. We assume that the user does not have any background knowledge about domain and hence the discovered knowledge is compared against PDK only.



Time	No. of tuples	Discovered rules	Novel Rules	Unexpected rules	Conforming rules
T1	4000	345	42	111	192
T2	4000	279	16	94	169
T3	4000	318	15	125	178
T4	4000	387	37	130	220
T5	4000	327	25	99	203
T6	4000	348	23	97	228
T7	4000	298	19	91	188
T8	4561	338	49	124	195
Total	32561	2640	196	871	1573

Fig. 3. The discovered knowledge at time T_1 to T_8

The results were simply enough to show the effectiveness of our framework and produce different types of knowledge include conforming, generalized/specialized, unexpected and novel rules.

6 Conclusion and Future Work

In this paper, we propose a framework for self-upgrading post-analysis filter used in *Analysis* stage of KDD process. The filter is based on quantification of novelty of the currently discovered rules (CDK) with respect to domain knowledge (DK), and previously discovered knowledge (PDK). We capture the user subjectivity by asking for thresholds for labeling different types of rules. The framework is implemented and evaluated using two datasets and has shown encouraging results.

Our future work includes enhancing the novelty framework using semantic matching techniques. Currently the work is going on to embed the framework in the mining algorithm, thus using it in the *Mining* stage of the KDD process.

References

1. Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery: An Overview. In Advances in Knowledge Discovery and Data Mining, Edited by U. M. Fayyad , G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Menlo Park, CA:AAAI/MIT Press. (1996).
2. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann, San Francisco, CA, USA. (1999).
3. Duda, R. O., Hart, P. E., Stork, D. G.: Pattern Classification. 2nd Edition, John Wiley & Sons (Asia) PV. Ltd. (2002).
4. Liu, B., Hsu, W., Chen, S.: Using General Impressions to Analyze Discovered Classification Rules. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD 97). (1997).
5. Piateskey-Shapiro, G., Matheus, C. J.: The Interestingness of Deviations. In Proceedings of AAAI Workshop on Knowledge Discovery in Databases. (1994).
6. Liu, B., Hsu, W., Mun, L., Lee, H.: Finding Interesting Patterns Using User Expectations. Technical Report:TRA7/96. Department of Information Systems and Computer Science, National University of Singapore (1996).
7. Silberschatz, A., Tuzhilin, A.: On Subjective Measures of Interestingness in Knowledge Discovery. In Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining. (1995).
8. Silberschatz, A., Tuzhilin, A.: What Makes Patterns Interesting in Knowledge Discovery Systems. In IEEE Transactions on Knowledge and Data Engineering. V.5, No.6. (1996).
9. Liu, B., Hsu, W.: Post Analysis of Learned Rules. In Proceedings of the 13th National Conference on AI(AAAI'96). (1996).
10. Padmanabhan, B., Tuzhilin, A.: Unexpectedness as a Measure of Interestingness in Knowledge Discovery. Working paper # IS-97-6, Dept. of Information Systems, Stern School of Business, NYU. (1997).
11. Kohonen, T.: Self-Organization and Associative Memory. 3rd edition, Springer, Berlin. (1993).
12. Yairi, T., Kato, Y., Hori, K.: Fault Detection by Mining Association Rules from House-keeping Data. In Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space (SAIRAS 2001).
13. Marsland, S.: On-Line Novelty Detection Through Self-Organization, with Application to Robotics. Ph.D. Thesis, Department of Computer Science, University of Manchester. (2001).
14. Basu, S., Mooney, R. J., Pasupuleti, K. V., Ghosh, J.: Using Lexical Knowledge to Evaluate the Novelty of Rules Mined from Text. In Proceedings of the NAACL workshop and other Lexical Resources: Applications, Extensions and Customizations. (2001).
15. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD98). (1998).
16. Pujari, A. K.: Data Mining Techniques. 1st Edition, Universities Press(India) Limited. (2001).
17. Dunham, M. H.: Data Mining: Introductory and Advanced Topics. 1st Edition, Pearson Education (Singaphore) Pte. Ltd. (2003).
18. Williams, G. J.: Evolutionary Hot Spots Data Mining: An Architecture for Exploring for Interesting Discoveries. In Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD99). (1999).
19. Psaila, G.: Discovery of Association Rule Meta-Patterns. In Proceedings of 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK99). (1999).

Revisiting Generic Bases of Association Rules

S. Ben Yahia[1] and E. Mephu Nguifo[2]

[1] Départment des Sciences de l'Informatique
Faculté des Sciences de Tunis
Campus Universitaire, 1060 Tunis, Tunisie
sadok.benyahia@fst.rnu.tn

[2] Centre de Recherche en Informatique de Lens
Rue de l'Université SP 16, 62307 Lens Cedex
mephu@cril.univ-artois.fr

Abstract. As a side effect of unprecedented amount of digitization of data, classical retrieval tools found themselves unable to go further beyond the tip of the Iceberg. Data Mining in conjunction with the Formal Concept Analysis, is a clear promise to furnish adequate tools to do so and specially to be able to derive concise generic and easy understandable bases of "hidden" knowledge, that can be reliable in a decision making process. In this paper, we propose to revisit the notion of association rule redundancy and to present sound inference axioms for deriving all association rules from generic bases of association rules.

1 Introduction

The last two decades witnessed the emergence of Data mining, with a clear promise to efficiently discover valuable, non obvious information from large databases by furnishing the adequate techniques and/or tools. Much research in data mining from large databases has focused on the discovery of association rules [1–3]. However, exploiting and visualizing association rules became far from being a trivial task, mostly because of the huge number of potentially interesting rules that can be drawn from a dataset. Various techniques are used to limit the number of reported rules, starting by basic pruning techniques based on thresholds for both the frequency of the represented pattern (called the *support*) and the strength of the dependency between premise and conclusion (called the *confidence*). More advanced techniques that produce only a limited number of the entire set of rules rely on closures and Galois connections [4–6]. These formal concept analysis (FCA) [7] based techniques have in common a feature, which is to present a better trade-off between the size of the mining result and the conveyed information than the "frequent patterns" algorithms. Finally, works on FCA have yielded a row of results on compact representations of closed set families, also called *bases*, whose impact on association rule reduction is currently under intensive investigation within the community [4, 5, 8]. Therefore, the problem of mining association rules might be reformulated under the FCA point of view as follows [9]:

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 58–67, 2004.

1. Discover two distinct "closure systems", i.e., sets of sets which are closed under the intersection operator, namely the two "order ideals" [7]: the set of closed itemsets and the set of minimal generators. Also, the *upper covers* (Cov^u) of each closed itemset should be available.
2. From all the information discovered in the first step, i.e., two closure systems and the upper covers sets, derive generic bases of association rules (from which all the remaining rules can be derived).

Once these generic bases were obtained, all the remaining (redundant) rules can be derived "easily". However, none of the reported works was interested in providing inference mechanisms to derive all the remaining rules. Indeed, a lot of shadow zones still remain in the derivation process. For example, is it sufficient to augment a generic exact rule's premise by an item of the same rule's conclusion? Are those generic bases "self containing"?, i.e., do they contain all the necessary information to derive all the remaining association rules?

In this paper, we try to find an answer to the above mentioned questions and to provide the adequate tools, through inference axioms, to derive all association rules from the generic bases. It is noteworthy that such derivation process could be useful in a visualization prototype [10]. Indeed, aiming to avoid a user knowledge overwhelming, only generic bases of association rules are visualized first. Then, once a user focuses on a particular rule, then additional knowledge could be furnished on demand (as recommended in Shneiderman's guidelines [11]).

The remainder of the paper is organized as follows. Section 2 introduces the mathematical background of FCA and its connection with the derivation of (non-redundant) association rules bases. Section 3 introduces a new definition of association rule redundancy and discusses in depth the process of association rule derivation from both exact and approximative generic bases of association rules. We provide for both generic bases a set of inference axioms that we prove their soundness. Section 4 concludes this paper and points out future research directions.

2 Mathematical Background

In the following, we recall some key results from the Galois lattice-based paradigm in FCA and its applications to association rules mining.

2.1 Basic Notions

In the rest of the paper, we shall use the theoretical framework presented in [7]. In this paragraph, we recall some basic constructions from this framework.

Formal Context:
A formal context is a triplet $\mathcal{K} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$, where $\mathcal{O}$ represents a finite set of objects (or transactions), $\mathcal{A}$ is a finite set of attributes and $\mathcal{R}$ is a binary (incidence) relation (i.e., $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$). Each couple $(o, a) \in \mathcal{R}$ expresses that the

transaction $o \in \mathcal{O}$ contains the attribute $a \in \mathcal{A}$. Within a context (c.f., Figure 1 on the left), objects are denoted by numbers and attributes by letters.

We define two functions, summarizing links between subsets of objects and subsets of attributes induced by R, that map sets of objects to sets of attributes and *vice versa*. Thus, for a set $O \subseteq \mathcal{O}$, we define $\phi(O) = \{a \mid \forall o, o \in O \Rightarrow (o, a) \in \mathcal{R}\}$; and for $A \subseteq \mathcal{A}$, $\psi(A) = \{o \mid \forall a, a \in A \Rightarrow (o, a) \in \mathcal{R}\}$. Both functions ϕ and ψ form a Galois connection between the sets $\mathcal{P}(\mathcal{A})$ and $\mathcal{P}(\mathcal{O})$ [12]. Consequently, both compound operators of ϕ and ψ are closure operators, in particular $\omega = \phi \circ \psi$ is a closure operator.

In what follows, we introduce the frequent closed itemset[1], since we may only look for itemsets that occur in a sufficient number of transactions.

Frequent Closed Itemset: An itemset $A \subseteq \mathcal{A}$ is said to be *closed* if $A = \omega(A)$, and is said to be *frequent* with respect to *minsup* threshold if supp(A)= $\frac{|\psi(A)|}{|\mathcal{O}|}$ $\geq$ *minsup.*

Formal Concept: A formal concept is a pair $c = (O, A)$, where O is called *extent*, and A is a closed itemset, called *intent*. Furthermore, both O and A are related through the Galois connection, i.e., $\phi(O) = A$ and $\psi(A) = O$.

Minimal Generator: An itemset $g \subseteq \mathcal{A}$ is *called minimal generator* of a closed itemset A, if and only if $\omega(g) = A$ and $\nexists g' \subseteq g$ such that $\omega(g') = A$ [4]. The closure operator ω induces an equivalence relation on items power set, i.e., the power set of items is partionned into disjoint subsets (also called *classes*). In each distinct class, all elements are equal support value. The minimal generator is the smallest element in this subset, while the closed itemset is the largest one. Figure 1(Right) sketches sample classes of the induced equivalence relation from the context $\mathcal{K}$.

Galois Lattice: Given a formal context $\mathcal{K}$, the set of formal concepts $\mathcal{C}_{\mathcal{K}}$ is a complete lattice $\mathcal{L}_c = (\mathcal{C}, \leq)$, called the *Galois (concept) lattice*, when $\mathcal{C}_{\mathcal{K}}$ is considered with inclusion between itemsets [7, 12]. A partial order on formal concepts is defined as follows $\forall$ $c_1, c_2 \in \mathcal{C}_{\mathcal{K}}$, $c_1 \leq c_2$ iif *intent*$(c_2) \subseteq$ *intent*(c_1), or equivalently *extent*$(c_1) \subseteq$ *extent*(c_2).

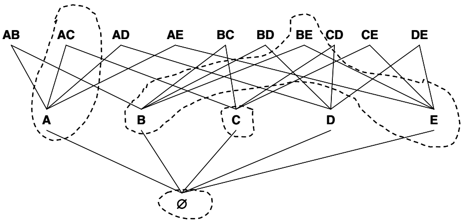
The partial order is used to generate the lattice graph, called *Hasse diagram*, in the following manner: there is an arc (c_1, c_2), if $c_1 \preceq c_2$ where $\preceq$ is the transitive reduction of $\leq$, i.e., $\forall c_3 \in \mathcal{C}_{\mathcal{K}}$, $c_1 \leq c_3 \leq c_2$ implies either $c_1 = c_3$ or $c_2 = c_3$.

Iceberg Galois Lattice: When only frequent closed itemsets are considered with set inclusion, the resulting structure $(\hat{\mathcal{L}}, \subseteq)$ only preserves the LUBs, i.e., the joint operator. This is called a join semi-lattice or upper semi-lattice. In the remaining of the paper, such structure is referred to as "*Iceberg Galois Lattice*".

Example 1. Let us consider the extraction context given by Figure 1 (Left). The associated Iceberg Galois lattice, for minsup=2, is depicted by Figure 1

[1] Itemset stands for a set of items.

	A	B	C	D	E
1	×		×	×	
2		×	×		×
3	×	×	×		×
4		×			×
5	×	×	×		×



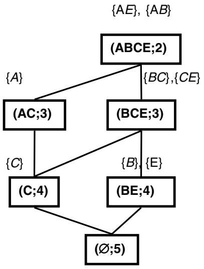


Fig. 1. Left: Extraction context $\mathcal{K}$ **Right:** Associated sample of equivalence relation classes **Bottom:** Associated Iceberg Galois lattice for minsup =2.

(bottom)[2]. Each node in the Iceberg is represented as couple (closed itemset; support) and is decorated with its associated minimal generators list.

In the following, we present the general framework for the derivation of association rules, then we establish its important connexion with the FCA framework.

2.2 Derivation of (Non-redundant) Association Rules

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of m distinct items. A transaction T, with an identifier further called *TID*, contains a set of items in $\mathcal{I}$. A subset X of $\mathcal{I}$ where $k = |X|$ is referred to as a $k-$itemset (or simply an itemset), and k is called the length of X. A transaction database, say $\mathcal{D}$, is a set of transactions, which can be easily transformed in an extraction context $\mathcal{K}$. The number of transactions of $\mathcal{D}$ containing the itemset X is called the support of X, i.e., supp(X)=| $\{T \in \mathcal{D} \mid X \subseteq T\}$ |. An itemset X is said to be frequent when support(X) reaches at least the user-specified minimum threshold minsup.

Association rule derivation is achieved from a set F of frequent itemsets in an extraction context $||$, for a minimal support *minsup*. An association rule R is a relation between itemsets of the form R: $X \Rightarrow (Y - X)$, in which X and Y are frequent itemsets, and $X \subset Y$. Itemsets X and $(Y - X)$ are called, respectively, *premise* and *conclusion* of the rule R. The valid association rules are those whose the strength metric conf(R)= $\frac{suppt(XY)}{supp(X)}$ is greater than or equal to the minimal threshold of confidence minconf. If conf(R)=1 then R is called *exact association rule (ER)*, otherwise it is called *approximative association rule (AR)*.

The problem of the relevance and usefulness of extracted association rules is of primary importance. Indeed, in most real life databases, thousands and even

[2] We use a separator-free form for sets, e.g., AB stands for $\{A, B\}$.

millions of high-confidence rules are generated among which many are redundant. This problem encouraged the development of tools for rule classification according to their properties, for rule selection according to user-defined criteria, and for rule visualization. In this paper, we are specially interested in the lossless information reduction, which is based on the extraction of a generic subset of all association rules, called *base*, from which the remaining (redundant) association rules may be derived. Bastide *et al.* considered the following rule-redundancy definition [13]:

Definition 1. *An association rule R: $X \Rightarrow Y$ is a minimal non-redundant association rule iff $\nexists$ an association rule R_1: $X_1 \Rightarrow Y_1$ fulfilling the following constraints:*

1. *supp(R) = supp(R_1) and conf(R) = conf(R_1);*
2. *$X_1 \subset X$ and $Y \subset Y_1$* [3]*.*

Exact association rules, of the form R: $X \Rightarrow (Y)$, are implications between two itemsets X and XY whose closures are identical, i.e., $\omega(X) = \omega(XY)$. Indeed, from the aforementioned equality, we deduce that X and XY belong to the same class of the equivalence relation induced by the closure operator ω on $\mathcal{P}(\mathcal{A})$ and then supp(X)= supp(XY) (i.e., conf(R) = 1).

Bastide *et al.* characterized what they called "*the generic basis for exact association rules*" (adapting the global implications base of *Duquenne and Guigues* [14]). The generic base for exact association rules, that has been proved to be lossless information, is defined as follows:

Definition 2. *Let FC be the set of frequent closed itemsets extracted from the context and, for each frequent closed itemset c, let us denote $\mathcal{G}_c$ the set of minimal generators of c. The generic basis for exact association rules is as follows:*
$GB = \{R : g \Rightarrow (c - g) \mid c \in FC$ and $g \in \mathcal{G}_c$ and $g \neq c\}$.

The authors also characterized the *informative basis* for approximate association rules, based on extending the family of bases of partial implications of *Luxenburger* [15] e.g., cover, structural and arborescent bases. The informative basis is defined as follows :

Definition 3. *The informative basis for approximate association rules is given by: LuxB = $\{R \mid R : X \Rightarrow Y$, $Y \in FC$ and $\omega(X) \leq Y$ and c=conf(R) $\geq$ minconf and supp(Y) $\geq$ minsup$\}$.*

With respect to Definitions 2 and 3, we consider that given an Iceberg Galois lattice, representing precedence-relation-based ordered closed itemsets, generic bases of association rules can be derived in a straightforward manner. We assume that in such structure, each closed itemset is "decorated" with its associated list of minimal generators. Hence, *AR* represent "inter-node" implications, assorted with a statistical information, i.e., the confidence, from a sub-closed-itemset to a super-closed-itemset while starting from a given node in an ordered structure.

[3] The strict inclusion is relaxed in [4].

Rule #	Implication
R_1	E⇒B
R_2	B⇒E
R_3	A⇒C
R_4	BC⇒E
R_5	CE⇒B
R_6	AB⇒CE
R_7	AE⇒BC

Rule #	Implication	Rule #	Implication
R_8	C$\overset{.75}{\Rightarrow}$A	R_{13}	B$\overset{.75}{\Rightarrow}$CE
R_9	C$\overset{.75}{\Rightarrow}$BE	R_{14}	E$\overset{.5}{\Rightarrow}$ABC
R_{10}	C$\overset{.5}{\Rightarrow}$ABE	R_{15}	B$\overset{.5}{\Rightarrow}$ACE
R_{11}	A$\overset{.66}{\Rightarrow}$BCE	R_{16}	BC$\overset{.66}{\Rightarrow}$AE
R_{12}	E$\overset{.75}{\Rightarrow}$BC	R_{17}	CE$\overset{.66}{\Rightarrow}$AB

Fig. 2. Left: non-redundant exact association rules **Right:** non-redundant approximative association rules.

Inversely, *ER* are "intra-node" implications extracted from each node in the ordered structure. For example, Generic bases of exact and approximative association rules that can be drawn from the context $\mathcal{K}$, are respectively depicted in Figure 2 (Left and Right).

3 Deriving All Association Rules

Little attention was paid to reasoning from generic bases comparatively to the battery of papers to define them. Essentially, they were interested in defining syntactic mechanisms for deriving rules form generic bases. In [16], the author presented a definition of a generic base and the associated derivation axioms. However, two drawbacks may be addressed. At first, only frequent itemsets are used to define the generic base, and one can imagine the length and the number of such frequent itemsets that could be derived from correlated extraction contexts. Second, the introduced generic base is not informative, i.e., it may exist a derivable rule for which it is impossible to determine exactly both its support and confidence. In [17], the author introduced another syntactic derivation operator, called the *cover*, defined as follows: Cover(X ⇒Y) = $\{X \cup Z \Rightarrow V \mid Z, V \subseteq Y \wedge Z \cap V = \emptyset \wedge V \neq \emptyset\}$. The number of the derived rules from a rule R: X ⇒ Y is equal to |Cover(R)|=3^m-2^m, where |Y|=m. For a derived rule R', we have conf(R')= max { conf(R_i)| R'∈ Cover(R_i)}. In the case of dense extraction contexts, where the length of conclusion rule drawn form such contexts are particularly high, one can claim that deriving association rule "informatively" (i.e., with their associated confidences) is nearly unfeasible. In what follows, we try to provide mechanisms to derive all association rules from generic bases. As defined, generic bases of association rules provide the "maximal boundaries" in which stand all the derivable rules. Indeed, a derivable rule cannot present a smaller premise than that of its associated generic rule, i.e., from which it can be derived. On the other hand, a derivable rule cannot present a larger conclusion than that of its associated generic rule. That's why we do not agree with the conclusion's strict inclusion relaxing in the redundancy definition presented in [4]. Therefore, we try to revisit the redundancy definition presented in Definition 1 and we will show through a toy example that it is not sufficient. For example, let us consider the exact rule R: AB⇒C that can be drawn from the extraction context depicted in Figure 1 (Left). With respect to Definition 1, R is not considered as redundant (or derivable). However, R can be easily derived, by decomposing the rule's conclusion, from the generic exact association rule R_6: AB⇒CE. We can converge to the same claim when we consider the approximative rule R': A$\overset{.66}{\Rightarrow}$B which can be easily derived from the generic approximative rule R_{11}: A$\overset{.66}{\Rightarrow}$BCE without that such "derivability" is formally catched out by Definition 1. That's why to palliate such insufficiency, we propose the following rule's redundancy.

Definition 4. *Let $\mathcal{AR}_{\mathcal{K}}$ be the set of association rules that can be drawn from a context $\mathcal{K}$. A rule R: $X\overset{c}{\Rightarrow}Y$* [4] *$\in \mathcal{AR}_{\mathcal{K}}$ is considered as redundant (or derivable) with respect to (from) R_1: $X_1\overset{c}{\Rightarrow}Y_1$ if R fulfils the following conditions:*

1. *Supp(R)=Supp(R_1) and conf(R)=conf(R_1)=c*
2. *($X_1 \subset X$ and $Y \subset Y_1$) or ($X_1 = X$ and $Y \subset Y_1$)*

With respect to Definition 4, the rule AB⇒C is considered as derivable from the rule AB⇒CE. Now, if we try to roughly guess how to derive association rules from generic bases, we will find that it can be done through two inference mechanisms. Indeed, since a generic rule's premise is minimal then one can guess that it is possible to augment this premise to derive another association rule. Also, since a generic rule's conclusion is maximal then one can derive another rule by splitting this conclusion. In the following, we will discuss in depth the association rules derivation process and we show that it is based mainly on a couple of augmentation/decomposition operations.

3.1 Axiomatizing Generic Bases of Exact Association Rules

In this subsection, we will try to axiomatize the derivation of exact association rules from its associated generic base. In the following, we introduce a set of inference axioms, providing mechanisms to derive exact association rules from the generic base of exact association rules, and prove their soundness.

Proposition 1. *Let $\mathcal{ERB}_{\mathcal{K}}$ and $\mathcal{EAR}_{\mathcal{K}}$ be the set of generic exact rules and the set of exact association rules, respectively, that can be drawn from a context $\mathcal{K}$. Then, the following inference axioms are sound:*

A0.Reflexivity: *If $X\Rightarrow Y \in \mathcal{ERB}_{\mathcal{K}}$ then $X\Rightarrow Y \in \mathcal{EAR}_{\mathcal{K}}$*
A1.Decomposition: *If $X\Rightarrow Y \in \mathcal{ERB}_{\mathcal{K}}$ then $X\Rightarrow Z \in \mathcal{EAR}_{\mathcal{K}}$, $\forall$ $Z \subset Y$*
A2.Augmentation *If $X\Rightarrow Y \in \mathcal{ERB}_{\mathcal{K}}$ then $X \cup Z\Rightarrow Y$-$\{Z\} \in \mathcal{EAR}_{\mathcal{K}}$, $\forall$ $Z \subset Y$*

Proof. **A0.Reflexivity:** follows from the proper definition of the generic base of exact association rules.

[4] When conf(R: X$\overset{c}{\Rightarrow}$Y) =1, then c is omitted, i.e., R is written as R: X⇒Y.

A1.Decomposition: Since X$\Rightarrow$Y $\in \mathcal{ERB}_{\mathcal{K}}$, we know that all itemset standing in the interval $[X, XY]$ belong to the same equivalence relation class, i.e., for all itemset I: X$\leq$I$\leq$XY, we have supp(X)=supp(I)=supp(XY). Therefore, since Z $\subset$ Y, then, by construction, the itemset XZ belongs to the interval $[X, XY]$, i.e., $[X, \ldots, XZ, \ldots, XY]$. Hence, supp(XZ)=supp(X) and conf (X$\Rightarrow$Z)=1, i.e., X$\Rightarrow$Z $\in \mathcal{EAR}_{\mathcal{K}}$.

A2.Augmentation Since X$\Rightarrow$Y $\in \mathcal{ERB}_{\mathcal{K}}$, then conf(X$\Rightarrow$Y)=1, i.e., supp(X)= supp(XY). We have to show that the interval [XZ,XZY] is included in that of [X,XY]. Since, X$\leq$XZ$\leq$XY, then X is a lower bound of XZ and from the fact that Z $\subset$ Y, we can conclude that XZY$\equiv$XY. Therefore,[XZ,XZY] $\subset$ [X,XY]. Hence, conf(X $\cup$ Z$\Rightarrow$Y-{Z})=1 and X $\cup$ Z$\Rightarrow$Y-{Z}$\in \mathcal{EAR}_{\mathcal{K}}$.

Example 2. Let us consider the generic base of exact rules depicted in Figure 2 (Left). Then by applying the axiom set, we extract the set of exact association rules, which are depicted in Figure 3 (Top). Note that all derived rules using the Decomposition axiom are marked by an (*), and those derived thanks to the Augmentation axiom are marked by a ($^{+}$).

3.2 Axiomatizing Generic Bases of Approximative Association Rules

In this subsection, we will try to axiomatize the derivation of approximative association rules from its associated generic base. In the following, we provide a set of inference axioms, providing mechanisms to derive approximative association rules from their associated approximative generic base, and prove their soundness.

Proposition 2. *Let $\mathcal{ARB}_{\mathcal{K}}$ and $\mathcal{AAR}_{\mathcal{K}}$ the set of generic approximative rules and the set of approximative association rules, respectively, that can be drawn from a context $\mathcal{K}$. Then, the following inference axioms are sound:*

A0.Reflexivity: *If $X \overset{c}{\Rightarrow} Y \in \mathcal{ARB}_{\mathcal{K}}$ then $X\overset{c}{\Rightarrow}Y \in \mathcal{AAR}_{\mathcal{K}}$*

A1.Decomposition: *If $X\overset{c}{\Rightarrow}Y \in \mathcal{ARB}_{\mathcal{K}}$ then $X\overset{c}{\Rightarrow}Z \in \mathcal{EAR}_{\mathcal{K}}$ / $|XZ|$=$|XY|$ and $Z \subset Y$.*

A2.Augmentation *If $X\overset{c}{\Rightarrow}Y \in \mathcal{ARB}_{\mathcal{K}}$ then $X \cup Z\overset{c}{\Rightarrow}Y$-{Z} $\in \mathcal{AAR}_{\mathcal{K}}$ / $|XZ|$ =$|X|$ and $Z \subset Y$.*

Proof. **A0.Reflexivity:** follows from the proper definition of the generic base of approximative association rules.

A1.Decomposition: Since, X$\overset{c}{\Rightarrow}$Y $\in \mathcal{ARB}_{\mathcal{K}}$ then conf(X$\overset{c}{\Rightarrow}$Y)=c $\Leftrightarrow \frac{|XY|}{|X|}$=c $\Leftrightarrow$ |XY|=|X|$\times$ c. We have also |XZ|=|XY|, then |XZ|= |X|$\times$ c. Therefore, the rule X$\overset{c}{\Rightarrow}$Z holds since Z $\subset$ Y. Hence, X$\overset{c}{\Rightarrow}$Z $\in \mathcal{EAR}_{\mathcal{K}}$.

A2.Augmentation Since, X$\overset{c}{\Rightarrow}$Y $\in \mathcal{ARB}_{\mathcal{K}}$ then conf(X$\overset{c}{\Rightarrow}$Y)=c $\Leftrightarrow \frac{|XY|}{|X|}$=c. We have to prove that $\frac{|XZY|}{|XZ|}$=c. Indeed, from Z $\subset$ Y we can say that XZY$\equiv$XY. From the equality |XZ|=|X|, we have $\frac{|XZY|}{|XZ|}=\frac{|XY|}{|X|}$=c. Therefore, X $\cup$ Z$\overset{c}{\Rightarrow}$Y-{Z} $\in \mathcal{AAR}_{\mathcal{K}}$.

Example 3. Let us consider the generic base of approximative rules depicted in Figure 2 (Left). Then by associated axiom set, we extract the set of approximative association rules, which are depicted in Figure 3 (Bottom). Note that, all derived rules using the Decomposition axiom are marked by an (*), and those derived thanks to the Augmentation axiom are marked by a ($^+$).

$E\Rightarrow B$	$B\Rightarrow E$	$A\Rightarrow C$	$BC\Rightarrow E$	$CE\Rightarrow B$	$AB\Rightarrow CE$	$AE\Rightarrow BC$
$AB\Rightarrow C^*$	$AB\Rightarrow E^*$	$AE\Rightarrow B\ ^*$	$AE\Rightarrow C^*$	$ABC\Rightarrow E^+$	$ABE\Rightarrow C^+$	$ACE\Rightarrow B^+$

$C\overset{.75}{\Rightarrow}A$	$C\overset{.75}{\Rightarrow}BE$	$C\overset{.5}{\Rightarrow}ABE$	$A\overset{.66}{\Rightarrow}BCE$	$E\overset{.75}{\Rightarrow}BC$	$B\overset{.75}{\Rightarrow}CE$
$E\overset{.5}{\Rightarrow}ABC$	$B\overset{.5}{\Rightarrow}ACE$	$BC\overset{.66}{\Rightarrow}AE$	$CE\overset{.66}{\Rightarrow}AB$	$AC\overset{.66}{\Rightarrow}BE^+$	$BE\overset{.75}{\Rightarrow}C^+$
$BE\overset{.5}{\Rightarrow}AC^+$	$BCE\overset{.66}{\Rightarrow}A^+$	$C\overset{.75}{\Rightarrow}B^*$	$C\overset{.75}{\Rightarrow}E^*$	$C\overset{.5}{\Rightarrow}AB^*$	$C\overset{.5}{\Rightarrow}AE^*$
$A\overset{.66}{\Rightarrow}B^*$	$A\overset{.66}{\Rightarrow}BC^*$	$A\overset{.66}{\Rightarrow}BE^*$	$A\overset{.66}{\Rightarrow}CE^*$	$E\overset{.75}{\Rightarrow}C^*$	$B\overset{.75}{\Rightarrow}C^*$
$E\overset{.5}{\Rightarrow}A^*$	$E\overset{.5}{\Rightarrow}AB^*$	$E\overset{.5}{\Rightarrow}AC^*$	$B\overset{.5}{\Rightarrow}A^*$	$B\overset{.5}{\Rightarrow}AC^*$	$B\overset{.5}{\Rightarrow}AE^*$
$BC\overset{.66}{\Rightarrow}A^*$	$CE\overset{.66}{\Rightarrow}A^*$	$AC\overset{.66}{\Rightarrow}B^*$	$AC\overset{.66}{\Rightarrow}E^*$	$BE\overset{.5}{\Rightarrow}A^*$	$BE\overset{.5}{\Rightarrow}C^*$

Fig. 3. (**Top**) Set of all exact association rules drawn from the context $\mathcal{K}$. (**Bottom**) Set of all approximative association rules drawn from the context $\mathcal{K}$.

It is important to mention that for deriving all approximative association rules, the order of applying the inference axioms is important. Indeed, we have to apply first the Augmentation axiom and next apply the Decomposition axiom on the resulting set of association rules, i.e., set of approximative generic rules and those derived by the Augmentation axiom. For example, if we begin by applying the Decomposition axiom first, we risk to miss deriving the last four rules depicted in Figure 3, i.e., $AC\overset{.66}{\Rightarrow}B$, $AC\overset{.66}{\Rightarrow}E$, $BE\overset{.5}{\Rightarrow}A$ and $BE\overset{.5}{\Rightarrow}C$.

Another fact concerning the "informative" property of the presented generic bases. Since all the necessary information is stored in the generic exact association rule base, it is easy to determine the support of an itemset in a derived rule. Indeed, by concatenation of the premise and the conclusion of a generic exact rule, we obtain the associated closed itemset and its associated support[5].

4 Conclusion

In this paper, we reported a syntactic and semantic discussion about generic bases of association rules. We provided also a set of sound inference axioms for deriving all association rules from generic bases of association rules. It is noteworthy that, contrarily to what was claimed by many authors, generic base of approximative association rules is not self containing. The reported algorithms are currently under implementation in order to include them in a graphical association rules visualization prototype [10].

[5] It is known that the support of an itemset is equal to the support of the smallest closed itemset containing it.

References

1. Agrawal, R., Skirant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Databases. (1994) 478–499
2. Brin, S., Motwani, R., Ullman, J.: Dynamic itemset counting and implication rules. In: Proceedings ACM SIGMOD, International conference on Management of Data, Tucson, Arizona, USA. (1997) 255–264
3. Manilla, H., Toinoven, H., Verkamo, I.: Efficient algorithms for discovering association rules. In: AAAI Worshop on Knowledge Discovery in Databases, Seattle, Washington (USA). (1994) 181–192
4. Bastide, Y., Pasquier, N., Taouil, R., Lakhal, L., Stumme, G.: Mining minimal non-redundant association rules using frequent closed itemsets. In: Proceedings of the international Conference DOOD'2000,LNCS, Springer-verlag. (2000) 972–986
5. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Intelligent structuring and reducing of association rules with formal concept analysis. In: Proc. KI'2001 conference, LNAI 2174, Springer-verlag. (2001) 335–350
6. Zaki, M.J.: Generating non-redundant association rules. In: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,Boston, MA. (2000) 34–43
7. Ganter, B., Wille, R.: Formal Concept Analysis. Springer-Verlag (1999)
8. BenYahia, S., Cherif, C.L., Mineau, G., Jaoua, A.: Découverte des règles associatives non redondantes: application aux corpus textuels. Revue d'Intelligence Artificielle (special issue of Intl. Conference of Journées francophones d'Extraction et Gestion des Connaissances(EGC'2003), Lyon,France **17** (2003) 131–143
9. BenYahia, S., Nguifo, E.M.: A survey of Galois connection semantics-based approaches for mining association rules. Technical report, IUT-Lens, Centre de Recherche en Informatique de Lens (CRIL), Lens, France (2004)
10. BenYahia, S., Nguifo, E.M.: Contextual generic association rules visualization using hierarchical fuzzy meta-rules. In: Proceedings of the IEEE Intl. conference on Fuzzy Systems FUZZ-IEEE'04 (to appear), Budapest, Hungary. (2004)
11. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualization. In Press, I.C.S., ed.: Proceedings IEEE Symposium on Visual Languages, Boulder, Colorado, 336–343 (1996)
12. Barbut, M., Monjardet, B.: Ordre et classification. Algèbre et Combinatoire. Hachette, Tome II (1970)
13. Pasquier, N.: Data Mining: algorithmes d'extraction et de réduction des règles d'association dans les bases de données. Doctorat d'université, Université de Clermont-Ferrand II, France (2000)
14. Guigues, J., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines (1986) 5–18
15. Luxenburger, M.: Implication partielles dans un contexte. Mathématiques et Sciences Humaines **29** (1991) 35–55
16. Luong, V.P.: Raisonnement sur les règles d'association. In: Proceedings 17ème Journées Bases de Données Avancées BDA'2001, Agadir (Maroc),Cépaduès Edition. (2001) 299–310
17. Kryszkiewicz, M.: Concise representations of association rules. In Hand, D.J., Adams, N.M., Bolton, R.J., eds.: Proceedings of Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK. Volume 2447 of Lecture Notes in Computer Science., Springer (2002) 92–109

Mining Maximal Frequently Changing Subtree Patterns from XML Documents

Ling Chen, Sourav S. Bhowmick, and Liang-Tien Chia

School of Computer Engineering
Nanyang Technological University, Singapore, 639798, Singapore

Abstract. Due to the dynamic nature of online information, XML documents typically evolve over time. The change of the data values or structures of an XML document may exhibit some particular patterns. In this paper, we focus on the sequence of changes to the structures of an XML document to find out which subtrees in the XML structure frequently change together, which we call Frequently Changing Subtree Patterns (*FCSP*). In order to keep the discovered patterns more concise, we further define the problem of mining *maximal FCSPs*. An algorithm derived from the FP-growth is developed to mine the set of *maximal FCSP*s. Experiment results show that our algorithm is substantially faster than the naive algorithm and it scales well with respect to the size of the XML structure.

1 Introduction

With the fast proliferation of available XML documents, data mining community has been motivated to discover knowledge from XML repositories. For example, recently, there has been increasing research effort in mining association rules from XML documents[2]; classifying XML data [8] and clustering XML[9]. One of the common features shared by existing work on XML mining is that they usually mine from a collection of XML documents at a certain time point.

Due to the dynamic nature of online information, XML documents typically evolves over time. Consequently, issues related to detecting changes to XML documents received considerable attention recently[13][5]. Detected changes to XML can be used in search engines, XML query systems etc, however, to the best of our knowledge, they have never been used for data mining. In this paper, we propose to learn knowledge from changes to XML. Particularly, we study mining frequent patterns from a sequence of changes to an XML document.

Current work on mining frequent patterns from XML documents exploited two types of data in XML: XML content and XML structure. The former discovers which data values occur together frequently[2]; the latter finds out which substructures usually appear together in an XML document[11]. Correspondingly, changes to XML documents can be divided as changes to XML content (also called *content deltas*) and changes to XML structure (also called *structural deltas*). Then, frequent patterns mined from XML content deltas discover which data values frequently change together, whereas frequent patterns mined

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 68–76, 2004.

from XML structural deltas find out which substructures are usually adjusted together. In this paper, we focus on the latter to mine frequent patterns from the changes to XML structure.

Discovered frequently changing subtree patterns can be useful in a wide range of applications. We enumerate some as follows.

- **Structure-based Document Clustering:** Clustering XML documents based on the structures embedded in documents is proposed in [12]. However, it may not be accurate enough to cluster according to structures existing in a particular snapshot only. In some cases, the evolutionary patterns of the structures can distinguish documents with higher precision. Then frequently changing subtree patterns mined from historical versions of different XML documents can be used to discriminate XML documents whose structures are similar in a snapshot but evolve differently.
- **Semantic XML Search Engine:** If two subtrees in an XML structure frequently change together, it is likely that the objects represented by the two subtrees have underlying semantic correlation. Then frequently changing subtree patterns can be used by semantic XML search engine [6], which returns semantically related document fragments that satisfy users' queries.

Therefore, novel knowledge obtained by mining changes to XML documents are useful. We then, in [4], proposed to mine Frequently Changing Subtree Patterns (*FCSP*) from a sequence of changes to an XML document. Given a sequence of historical versions of an XML document, we discovered which subtrees of the XML structure frequently change together. In order to keep the set of discovered *FCSP*s concise so that knowledge can be understood easily, we define the notion of *maximal FCSPs* in this paper, which contains only concise information of *FCSPs*. Our definition on *maximal FCSPs* is fundamentally different from the traditional definition on maximal frequent patterns because of the different framework, which necessitates developing new mining solutions.

The main contributions of this paper are summarized as follows.

- A new problem of mining maximal frequent pattern from structural changes to historical XML documents is formally defined. The inherent relationship between discovered *FCSP*s is exploited to define the concise set of *maximal FCSP*s.
- An algorithm, derived from the FP-growth algorithm, is developed to mine the set of *maximal FCSP*s.
- Experiments are conducted to evaluate the efficiency and scalability of the designed algorithm.

The remainder of the paper is organized as follows. We define the problem of *maximal FCSP* mining formally in Sections 2. Section 3 presents the algorithm we developed for *maximal FCSP* mining. In Section 4, we evaluate the performance of the algorithm based on some experimental results. We conclude our work and discuss future work in Section 5.

2 Problem Statement

In this section, we first describe some preliminary concepts and basic change operations resulting in structural changes. Then, we define several metrics to measure the change degree and the change frequency of substructures. Finally, the problem of *maximal FCSP* mining is defined based on the metrics.

An XML document can be represented as a tree according to Document Object Model (DOM) specification. In this paper, we model the structure of an XML document as an unordered tree $T = (N, E)$, where N is the set of nodes and E is the set of edges. Then substructures in the XML document can be modelled as *subtrees*. A tree $t = (n, e)$ is a subtree of T, denoted as $t \prec T$, if and only if $n \subset N$ and for all $(x, y) \in e$, x is a parent of y in T. Actually, we treat an XML tree T as a *forest* which is a collection of subtrees $t \prec T$. Furthermore, we call subtree $\hat{t}$ an ancestor of subtree t if the root of $\hat{t}$ is an ancestor of the root of t. Conversely, subtree t is a descendant of subtree $\hat{t}$. In the context of *FCSP* mining, we consider the following two basic change operations: *Insert(X(name, value), Y)*, which creates a new node X, with node name "name" and node value "value", as a child node of node Y in an XML tree structure and *Delete(X)*, which removes node X from an XML tree structure.

Now we introduce some metrics which are used to measure the degree of change and the frequency of change for subtrees. Frequently changing subtree patterns can then be identified based on these metrics.

Degree of Change. Let $<t^i, t^{i+1}>$ be two historical versions of a subtree t in an XML tree structure T. Let $|d(t, i, i+1)|$ be the number of basic change operations which is required to change the structure of t from the ith version to the $(i+1)$th version. Let $|t^i \cup t^{i+1}|$ be the number of unique nodes of tree t in ith version and $(i+1)$th version. Then the *degree of change* for subtree t from version i to version $(i+1)$ is:

$$DoC(t,\ i,\ i{+}1) = \frac{|d(t, i, i+1)|}{|t^i \cup t^{i+1}|}$$

□

If the two versions are the same, *DoC* of the subtree will be zero; if the two versions are completely different, *DoC* of the subtree will be one. Obviously, the greater the value of *DoC*, the more dramatically the subtree has changed.

Frequency of Change. Let $<T^1, T^2, \ldots T^n>$ be a sequence of historical versions of an XML tree structure T. Let $<\Delta_1, \Delta_2, \ldots \Delta_{n-1}>$ be the sequence of *deltas* generated by comparing each pair of successive versions, where Δ_i $(1 \leq i \leq n\text{-}1)$ consists of subtrees changed in two versions. Let S be a set of subtrees, $S = \{t_1, t_2, \ldots t_m\}$, $DoC(t_j,\ i,\ i{+}1)$ be the *degree of change* for subtree t_j from ith version to $(i+1)$th version. The *FoC* of the set S is:

$$FoC(S) = \frac{\sum_{i=1}^{n-1} V_i}{n-1}$$

$$where\ V_i = \prod_{j=1}^{m} V_{j_i}\ and\ V_{j_i} = \begin{cases} 1, \text{if } DoC(t_j, i, i+1) \neq 0 \\ 0, \text{if } DoC(t_j, i, i+1) = 0 \end{cases} 1 \leq j \leq m \qquad \square$$

When subtrees in a set change together in every *delta*, *FoC* of the set will be one; when subtrees in a set never change together, *FoC* of the set will be zero.

Weight. Let $<T^1,T^2,...T^n>$ be a sequence of historical versions of an XML tree structure. Let $<\Delta_1,\Delta_2,...\Delta_{n-1}>$ be the sequence of deltas. Let S be a set of subtrees, S=$\{t_1,t_2,...t_m\}$, $FoC(S)$ be the *frequency of change* for the set S. Suppose a user-defined *minimum DoC* for each subtree is α, we define the *Weight* of the set of subtrees as follows:

$$Weight(S) = \frac{\sum_{i=1}^{n-1} D_i}{(n-1) * FoC(S)}$$

$$where\ D_i = \prod_{j=1}^{m} D_{j_i}\ and\ D_{j_i} = \begin{cases} 1, \text{if } DoC(t_j, i, i+1) \geq \alpha \\ 0, \text{otherwise} \end{cases} 1 \leq j \leq m \qquad \square$$

If all subtrees in a set change significantly every time they change together, the *Weight* of the set will be one; if subtrees in a set never change significantly when they change together, the *Weight* of the set will be zero.

Frequently Changing Subtree Pattern (FCSP). An *FCSP* is defined to be a set of subtrees which not only frequently change together but also usually change significantly when they change together.

Definition 1. *A set of subtrees* S=$\{t_1, t_2, ..., t_m\}$ *is a Frequently Changing Subtree Pattern if it satisfies the following two conditions:*

- *FoC of the set is no less than some user-defined minimum FoC* β*, FoC(S)* $\geq \beta$.
- *Weight of the set is no less than some user-defined minimum Weight* γ*, Weight(S)* $\geq \gamma$. □

Maximal Frequently Changing Subtree Pattern. In classical frequent pattern mining, all subsets of a frequent pattern are frequent as well. Maximal patterns are defined to be those which do not have any frequent superset [10]. Then the complete set of frequent patterns can be represented by the concise set of maximal frequent patterns. However, in the context of *FCSP* mining, *maximal FCSPs* cannot be defined in the same way because of the "Non Downward Closure" property. That is, a subset of an *FCSP* is not necessary an *FCSP* as well. Then, even if we find a set of *FCSPs*, in which each pattern does not have any frequent supersets, we cannot use it to represent the complete set of *FCSP*s. Hence, we examine first which *FCSP*s can be inferred from others.

Definition 2. *Given two subtree sets* S *and* S_1*, where* $S = S_1 \cup \{t_1, t_2, ..., t_n\}$. *If* $\forall$ *i* ($1 \leq i \leq n$), $\exists\ t_j \in S_1$ *s.t.* $t_j \prec t_i$*, we say* S_1 *is subsumed by* S*, or* S *subsumes* S_1*, denoted as* $S_1 \prec S$. □

Property 1. Given two subtree sets S and S_1 s.t. $\mathrm{S}_1 \prec \mathrm{S}$. If subtree set S is an *FCSP*, then S_1 is an *FCSP* as well.

The proof is given [3]. Based on Property 1, we can infer that all subsumed subsets of an *FCSP* are *FCSP*s as well. Then we define *maximal FCSP* as follows.

Definition 3. *A set of subtrees is a maximal FCSP if it is an FCSP and it does not subsumed by any other FCSPs.* □

Obviously, the set of *maximal FCSP* is a tightened set of *FCSP*, $\{$*maximal FCSP*$\} \subseteq \{$*FCSP*$\}$. Moreover, the complete set of $\{$*FCSP*$\}$ can be inferred from $\{$*maximal FCSP*$\}$.

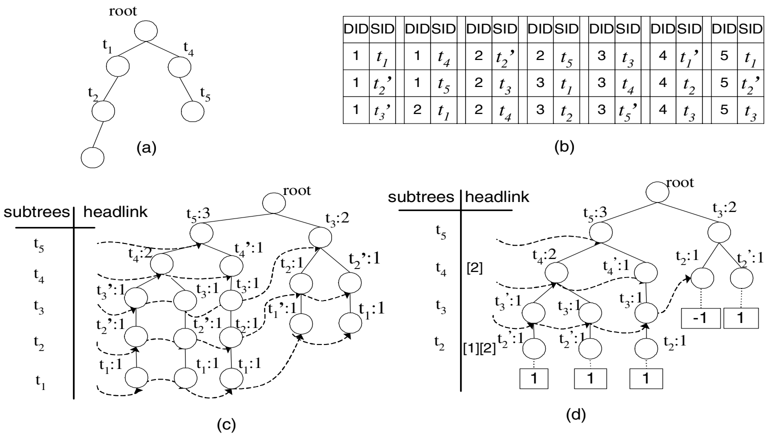
Problem Definition. The problem of *maximal FCSP* discovery can be formally stated as follows: Let $<T_1, T_2, \ldots\ T_n>$ be a sequence of historical versions of an XML tree structure. Let $<\Delta_1,\ \Delta_2,\ \ldots, \Delta_{n-1}>$ be the sequence of *deltas*. A **Structural Delta DataBase** *SDDB* can be formed from the set of *deltas*, where each tuple <*DID*, *SID*, *DoC*> comprises of a delta identifer, a subtree identifier and a *degree of change* for the subtree. Let $S=\{t_1, t_2, \ldots t_m\}$ be the set of changed subtrees such that each $\Delta \subseteq S$. Given an *SDDB*, an *FoC* threshold β and a *Weight* threshold γ, a subtree set $X \subseteq S$ is a **Frequently Changing Subtree Pattern** if $FoC(X) \geq \beta$ and $Weight(X) \geq \gamma$. A subtree set $Y \subseteq S$ is a **maximal Frequently Changing Subtree Pattern** if it is an *FCSP* and it does not subsumed by any other *FCSP*. The **problem of *maximal FCSP* discovery** is to find the set of all maximal frequently changing subtree patterns.

3 Algorithms

In this section, we present the algorithm to find the set of *maximal FCSPs* efficiently. Before the mining process, the *SDDB* should be generated first from the input of a sequence of historical XML versions. Since existing change detection systems [13][5] can detect all change operations resulting in structural changes (*insert* and *delete*), the *SDDB* can be constructed in a straightforward way. Hence, the following discussion is focused on the mining procedure, with the input of an *SDDB*.

3.1 Frequent Changing Subtree Pattern Discovery

We developed an algorithm in [4] to discover the set of *FCSPs*: *Weighted-FPgrowth*, which is based on the well known FP-growth[7] for classical frequent pattern mining. Basically, we construct a similar data structure as *FPtree*, where each node in the FPtree-like structure represents a changed subtree and different labels are used to indicate whether a subtree changes significantly or not in each *delta*. Interested readers can refer to [4] for the details.



DID	SID	DID	SID	DID	SID	DID	SID	DID	SID	DID	SID	DID	SID
1	t_1	1	t_4	2	t_2'	2	t_5	3	t_3	4	t_1'	5	t_1
1	t_2'	1	t_5	2	t_3	3	t_1	3	t_4	4	t_2	5	t_2'
1	t_3'	2	t_1	2	t_4	3	t_2	3	t_5'	4	t_3	5	t_3

Fig. 1. Ancestor Relationship and Modified Weighted-FPtree

3.2 Maximal Frequently Changing Subtree Pattern Discovery

The search for the set of *maximal FCSPs* can be achieved by a few optimization strategies as explained below, based on *Weighted-FPgrowth.* In order to mine the set of *maximal FCSPs* efficiently, it is advisable to examine a pattern only if none of its superset which subsumes it is frequent. Thus, a pattern S should be generated before any pattern S_1 such that $S_1 \prec S$. According to the Definition 2 on subsumption relationship, we have the following optimizing technique.

Optimization 1 *FCSPs containing subtrees rooted at nodes of higher-levels in an XML tree should be generated earlier than those containing subtrees rooted at nodes of lower-levels.*

Optimization 1 guides how to order the list of subtrees in the head table of *Weighted-FPtree.* We decide to arrange individual changed subtrees in the reverse order of depth-first first traversal in consideration of the following property.

Property 2. Given two subtree sets, S_1 and S, such that $S_1 \prec S$, the projected *delta* sets of S is same as the projected *delta* sets of S_1.

The proof of the property is given in [3]. According to Property 2, we are presented with the opportunity to mine both S and S_1 from a same *conditional Weighted-FPtree.* For example, given the ancestor relationship shown in Figure 1 (a), we arrange individual subtrees in reverse order of depth-first traversal, such as $\{t_5, t_4, t_3, t_2, t_1\}$, then the *conditional Weighted-FPtree* constructed for subtree t_1 can be used to mine not only all patterns related to t_1 but also all patterns related to t_2 (or t_3) but t_1

As explained in [4], labels of some nodes in *conditional Weighted-FPtree* need to be shifted to record correct information. In this case, a *conditional Weighted-FPtree* may not be sharable. Hence, we propose an alternative technique of shifting node labels. We append a tag to each path in the *conditional Weighted-FPtree*, which indicates the states of subtrees whose patterns are currently being mined. For example, given a transformed structural delta database shown in Figure 1 (b), where t_i and t_i*'* mean subtree t_i change significantly or insignificantly in this delta, a *Weighted-FPtree* can be constructed as in Figure 1 (c). Figure 1 (d) shows the *conditional Weighted-FPtree* for subtree t_1. Each path is appended with a tag: "1" indicates that t_1 changed significantly in this path while "-1" indicates it changed insignificantly in this path.

Moreover, with the above scheme on ordering subtrees and the technique making *conditional Weighted-FPtree* shareable, we can quickly decide whether or not to examine a pattern, depending on the state of the pattern which subsumes it. For example, from the *conditional Weighted-FPtree* shown in Figure 1 (d), we mine pattern $\{t_1, t_2\}$ first. Only if $\{t_1, t_2\}$ is not an *FCSP*, we go on mining $\{t_2\}$ from the same data structure. Otherwise, $\{t_2\}$ must be an *FCSP* but not maximal. As shown in Figure 1 (d), there are two counts maintained by t_2 to record the *Weight* information for $\{t_1, t_2\}$ and $\{t_2\}$ respectively. Actually, all patterns related to subtree t_2 can be decided in the same way. Thus, we do not need to mine patterns related to t_2 from the original *Weighted-FPtree* and have the following optimization.

Optimization 2 *From the head table of (conditional) Weighted-FPtree, only the last subtree and subtrees which are not descendant of the subtree whose patterns are just examined in the previous turn need to be mined.*

When the *Weighted-FPtree* contains a single path, we have the following optimization strategy.

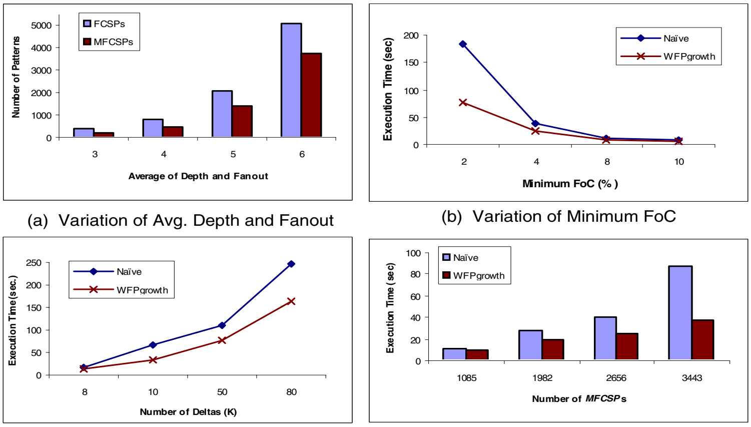
Optimization 3 *If the (conditional) Weighted-FPtree contains single path, maximal FCSPs can be generated directly from subtrees in the head table which are 1) with their node identifier as t_i and 2) either last subtree or not descendant of the subtree mined in the previous turn.*

4 Experiment Results

In this section, we study the performance of the proposed algorithms. The algorithms are implemented in Java. Experiments are performed on a Pentium IV 2.8GHz PC with 512 MB memory. The operating system is Windows 2000 professional. We implemented a synthetic structural delta generator by extending the one used in [1]. The default number of deltas is 10000 and the number of changed subtrees is 1000.

We carried out four experiments to show the conciseness of *maximal FCSP*s, the efficiency and scalability of designed algorithm compared with a naive algorithm which discover the complete set of FCSPs first and then filter non-maximal ones.

- Conciseness of *maximal FCSP*s: Firstly, we contrast the size of the set of *maximal FCSP*s with the size of the complete set of *FCSP*s by varying the



(a) Variation of Avg. Depth and Fanout

(b) Variation of Minimum FoC

(c) Variation of Number of Deltas

(d) Variation of Number of *MFCSP*s

Fig. 2. Experiment Results

average depth and fanout of the ancestor relationships. As shown in Figure 2 (a), the gap between the two sizes is greater when average depth and fanout of ancestor relationships are greater, since more *FCSP*s might be subsumed by their supersets.

– Efficiency Study: We compare the execution time of the developed algorithm against the naive algorithm by varying the threshold of *min_FoC* from 2% to 10%. As shown in Figure 2 (b), when the threshold is lower, our algorithm is more efficient than the naive one because the naive algorithm need to verify more patterns, on the contrary, designed algorithm has the chance to skip checking more patterns.
– Scalability Study I: We test the scale-up features of the two algorithms against the number of deltas, which is varied from 8K to 80K. Figure 2 (c) shows that, when the number of deltas is larger, the gap between the two algorithms is greater, since the more subtrees potentially be *FCSP*s.
– Scalability Study II: We also observed the scalability of the two algorithms with respect to the number of discovered *maximal FCSP*s. As presented in Figure 2 (d), for mining the same number of *maximal FCSP*s, the designed algorithm is faster than the naive one. Furthermore, when the size of the set of *maximal FCSP*s increases, the designed algorithm scales even better.

5 Conclusions and Future Work

This paper proposed a novel problem of mining maximal frequent patterns based on changes to XML structures: *maximal FCSP* mining. An algorithm, optimized *Weighted-FPgrowth*, is designed and implemented. Preliminary experiment results demonstrated that the conciseness of the set of *maximal FCSP*s. Moreover, the designed algorithm is significantly more efficient than the naive algorithm.

As ongoing work, we would like to collect real life dataset to verify the semantic meaning of discovered *maximal FCSPs*. In addition, we would also like to investigate issues about mining frequent patterns from content deltas and hybrid (content and structural) deltas of historical XML documents.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB*, pages 487–499, 1994.
2. D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. L. Lanzi. A tool for extracting xml association rules from xml documents. In *Proc. IEEE ICTAI*, pages 57–65.
3. L. Chen and S. S. Bhowmick. Web structural delta association rule mining: Issues, chanllenges and solutions. In *TR, NTU, Singapore*, http://www.ntu.edu.sg/home5/pg02322722/TR.html.
4. L. Chen, S.S. Bhowmick, and L.T. Chia. Mining association rules from structural deltas of historical xml documents. In *Proceedings of PAKDD 2004, Sydney, Australia*, 2004.
5. G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in xml documents. In *Proc. ICDE*, 2002.
6. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. Xsearch: A semantic search engine for xml. In *Proc. VLDB*, 2003.
7. J.W. Han, J. Pei, and Y.W. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD*, pages 1–12, 2000.
8. M. J.Zaki and C. C. Aggarwal. Xrules: An effective structural classifier for xml data. In *SIGKDD, Washington, DC, USA*, 2003.
9. M. L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: Clustering xml schemas for effective integration. In *ACM 11th CIKM, McLean, VA*, 2002.
10. Jr. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the ACM SIGMOD, Seattle, WA*, 1998.
11. A. Termier, M.C. Rousset, and M. Sebag. Treefinder: A first step towards xml data mining. In *Proc. IEEE ICDE*, pages 450–457, 2002.
12. K. Wang and H. Liu. Discovering structural association of semistructured data. In *IEEE TKDE, vol.12*, pages 353–371, 2000.
13. Y. Wang, D.J. DeWitt, and J.Y. Cai. X-diff: An effective change detection algorithm for xml documents. In *Proc. ICDE*, 2003.

Discovering Pattern-Based Dynamic Structures from Versions of Unordered XML Documents

Qiankun Zhao[1], Sourav S. Bhowmick[1], and Sanjay Madria[2]

[1] School of Computer Engineering, Nanyang Technological University
Singapore, 639798
assourav@ntu.edu.sg

[2] Department of Computer Science, University of Missouri-Rolla
Rolla, MO, USA, 65409
madrias@umr.edu

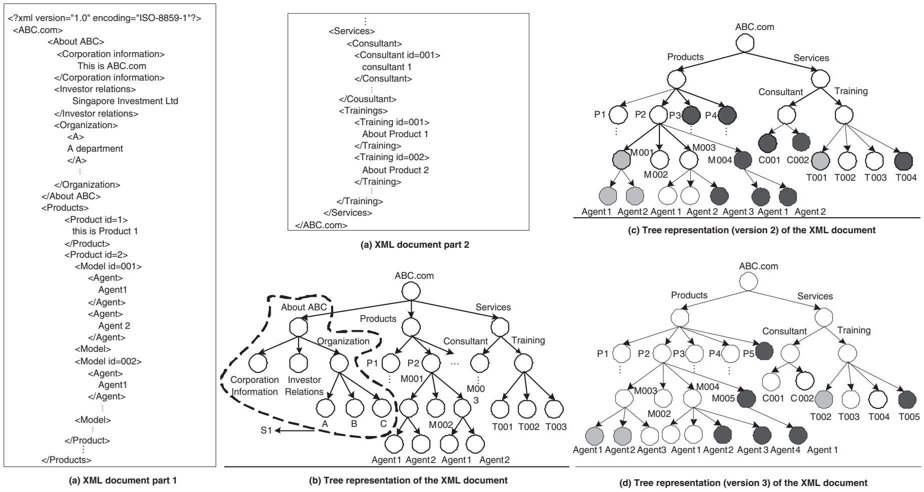
Abstract. Existing works on XML data mining deal with snapshot XML data only, while XML data is dynamic in real applications. In this paper, we discover knowledge from XML data by taking account its dynamic nature. We present a novel approach to extract *pattern-based dynamic structures* from versions of unordered XML documents. With the proposed *dynamic metrics*, the pattern-based dynamic structures are expected to summarize and predict interesting change trends of certain structures based on their past behaviors. Two types of pattern-based dynamic structures, *increasing dynamic structure* and *decreasing dynamic structure* are considered. With our proposed data model, *SMH-Tree*, an algorithm for mining such pattern-based dynamic structures with only two scans of the XML sequence is presented. Experimental results show that the proposed algorithm can extract the pattern-based dynamic structures efficiently with good scalability.

1 Introduction

Recently, with the increasing amount of XML data available, knowledge engineering from XML data becomes an issue of great importance [1, 5, 9, 10]. Existing work on mining XML data includes frequent substructure mining [4, 5, 8, 9], classification [10] and association rule mining [2]. Among these, the frequent substructure mining is the most well researched topic. The basic idea is to extract substructures (subtrees or subgraphs), which occur frequently among a set of XML documents or within an individual XML document.

However, most of the existing frequent substructure mining techniques focus only on snapshot XML data, while, in real life, XML data is dynamic. For example, Figure 1 (a) is an XML document that records the *products, clients*, and *agents* in an e-commerce web site. Figure 1 (b) is the corresponding tree representation of this XML document, where each node denotes for an element/attribute; each edge represents the parent and child relationship. For brevity, we use P_i, C_i, M_i, and T_i to represent the *Product, Consultant, Model,* and *Training* elements whose attribute *id* is *i*. Figures 1 (c) and (d) are the tree

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 77–86, 2004.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <ABC.com>
       <About ABC>
         <Corporation information>
              This is ABC.com
        </Corporation information>
        <Investor relations>
             Singapore Investment Ltd
        </Investor relations>
        <Organization>
           <A>
           A department
           </A>
             ⋮
        </Organization>
      </About ABC>
      <Products>
           <Product id=1>
             this is Product 1
           </Product>
           <Product id=2>
             <Model id=001>
                <Agent>
                  Agent1
                </Agent>
                <Agent>
                  Agent 2
                </Agent>
             <Model>
             <Model id=002>
                <Agent>
                  Agent1
                </Agent>
                   ⋮
             <Model>
                 ⋮
           </Product>
               ⋮
      </Products>
```

(a) XML document part 1

```
                ⋮
        <Services>
            <Consultant>
              <Consultant id=001>
                consultant 1
              </Consultant>
                   ⋮
          </Cousultant>
          <Trainings>
              <Training id=001>
                About Product 1
              </Training>
              <Training id=002>
                About Product 2
              </Training>
                   ⋮
         </Training>
      </Services>
</ABC.com>
```

(a) XML document part 2

(b) Tree representation of the XML document

(c) Tree representation (version 2) of the XML document

(d) Tree representation (version 3) of the XML document

Fig. 1. An Example

representations of the second and third version of the document. We use the black circles to represent nodes that are newly inserted; gray circles are used to represent nodes that are deleted; bolded circles represent nodes that have been updated. The dynamic nature of XML leads to two challenging problems. First, is the maintenance of frequent substructures for existing mining algorithms. Second, is the discovery of novel knowledge hidden behind the historical changes to XML data, some of which are difficult or impossible to discover from snapshot data. In this paper, we focus on the second issue. Consider the previous example, following types of knowledge can be discovered. Notice that, the list is by no means exhaustive.

- *Frequently changing or frozen structures/contents*: Some parts of the XML may change more frequently and significantly (such as the structure rooted at node *products*); while other structures or contents may not changed at all (such as structure s_1 in Figure 1 (b)).
- *Association rules*: Some structures/contents are associated in terms of their changes. For instance, whenever the substructure rooted at node P_2 changes, the substructure rooted at node *Training* also changes.
- *Change patterns*: More and more nodes are inserted under substructure *Consultant*, while nodes are inserted and deleted frequently under substructure *Training* in the above example.

Such novel knowledge can be useful in different applications such as XML change detection, dynamic XML indexing, and semantic meaning extraction [11]. The issue of discovering the first two types of knowledge (frequently changing structures and association rules) has been addressed in [12] and [3]. In this paper, we focus on discovering the *pattern-based dynamic structures* from versions of unordered XML documents. We focus on unordered XML because that the unordered model is more suitable for most database applications [7]. The *change patterns* can be critical for monitoring and predicting trends in e-commerce. Consider the above example in Figure 1, it can be observed that more and more *agent* nodes are inserted/deleted under product model M_{004}/M_{002}, while other structures do not have such change patterns. This change patterns may indicate that product model M_{004} is becoming more popular or the company is paying more efforts to this product model. We call such structures as *pattern-based dynamic structures*. Better promotion, marketing, and R&D strategies can be made based on such structures.

Our contributions can be summarized as follows: (1) We proposed the issue of mining pattern-based dynamic structures from the changes to XML documents in Section 3; (2) In Section 4, an algorithm is presented for discovering pattern-based dynamic structures; (3) Experiments have been conducted to evaluate the performance of the algorithm in Section 5. In the next section we discuss related work.

2 Related Work

In this section, we review some of the structured data mining techniques.

Since XML data is semi-structured and widely used, data mining of semi-structured data has attracted many research efforts recently [4, 5, 8, 9]. Most existing work focus on discovering the frequent substructures from a collection of semi-structured data such as XML documents. AGM [4] is an Apriori-based algorithm for mining frequent substructures. However, results of AGM are restricted to only the *induced* substructures. FSG [5] is also an Apriori-based algorithm for mining all *connected* frequent subgraphs. Experiments results in [5] show that FSG is considerably faster than AGM. However, both AGM and FSG do not scale to very large database. gSpan [8] is an algorithm for extracting frequent subgraphs without candidate generation. It employs the depth-first search strategy over the graph database. Like AGM, gSpan needs elaborate computations to deal with structures with non-canonical forms. More recently, TreeMiner [9] is proposed to discover the frequent *embedded* substructure, which is a generalization of induced substructure. However, the TreeMiner does not scale to very large XML documents either.

Different from the above techniques, which focus on designing ad-hoc algorithms to extract structures that *occur frequently* in the snapshot data collections, pattern-based dynamic structure mining is to extract structures that *change frequently* according to *certain patterns* in their change histories.

3 Definitions and Problem Statement

In this section, we first present the definitions of *dynamic metrics*. Next, we give the problem statement of *pattern-based dynamic structure mining* from versions of XML documents. Preliminary concepts are not included due to the limitation of space.

We model the structures of XML documents as unordered, labeled, rooted trees, which is denoted as $S = (N, E, r)$, where N is the set of labeled nodes, E is the set of edges, $r \in N$ is the root. We do not distinguish between elements and attributes, both of them are mapped to the set of labeled nodes. Each edge, $e = (x, y)$ is an ordered pair of nodes, where x is the parent of y. The *size* of the structure S, denoted by $|S|$, is the number of nodes in N.

From the example in Figure 1, we observed that different substructures of the XML document might change in different ways. To evaluate their behaviors, we propose a set of *dynamic metrics*. The first metric is called SMF (*Structure Mutation Factor*). It measures how significantly a structure has changed from one version to another. The second metric is called VMF (*Version Mutation Factor*). It measures how frequently a substructure has changed.

Definition 1 (SMF). *Let $\langle S_i, S_{i+1} \rangle$ be tree representations of two XML documents. Suppose $s \preceq S_i$. The SMF of s from document X_i to document X_{i+1}, denoted by $SMF_i(s)$, is defined as: $SMF_i(s) = \frac{|\triangle_{s_i}|}{|s_i \uplus s_{i+1}|}$.*

Here $SMF_i(s)$ is the structure mutation factor of s from version i to $i+1$. $SMF_i(s)$ is the percentage of nodes that have changed from X_i to X_{i+1} in s against the number of nodes in its consolidation structure. For example, consider the two structures shown in Figures 1 (b) and 1 (c), the SMF value for the substructure rooted at node P_2 from version 1 to version 2 is 0.58 (7/12). It can be observed that $SMF_i(s) \in [0, 1]$.

Definition 2 (VMF). *Let $\langle S_1, S_2, \cdots, S_n \rangle$ be tree representations of n XML documents. Suppose $s \preceq S_j$. The VMF of s is defined as: VMF(s) $= \frac{\sum_{i=1}^{n} v_i}{n-1}$, where v_i is 1 if $|\triangle_{s_i}| \neq 0$; otherwise v_i is 0.*

Consider the 2 different versions of the XML document in Figure 1. We calculate the VMF value for the substructure rooted at node P_2. The n value here is 3. For the first delta, $|\triangle_{P_{2_1}}|$ and $|\triangle_{P_{2_2}}|$ are not 0, so v_1 and v_2 are both 1. Then, $\sum_{i=1}^{2} v_i = 2$. Consequently, the VMF of this substructure is 1 (2/2). Also, it can be observed that $VMF(s) \in [0, 1]$.

Given a sequence of historical versions of an XML document, the problem of pattern-based dynamic structure mining is to extract structures that are interesting with respect to their historical change behaviors. In this paper, we focus on two types of pattern-based dynamic structures *increasing dynamic structure* and *decreasing dynamic structure*. Based on the dynamic metrics, they are defined as follows.

Definition 3. *[IDS] Let $\langle S_1, S_2, \cdots, S_n \rangle$ be tree representations for n versions of XML documents. Let the thresholds for VMF, strength, and tolerance be α, β and γ respectively. A structure $s \preceq S_j$ is an **increasing dynamic structure (IDS)** in this sequence iff: $VMF(s) \geq \alpha$, $\forall$ $SMF_i(s) \neq 0$, $SMF_i(s)$-$SMF_j(s)$*

$< SMF_i(s)*\gamma$, SMF_j is the last nonzero SMF value where $i > j \geq 1$, and $strength(s) \geq \beta$, where strength(s) is defined as: $strength(s) = \frac{\sum_{i=2}^{n} v_i}{(n-1)*VMF(s)}$, where $v_i = 1$, if $SMF_i(s) \neq 0$ and $SMF_i(s) \geq SMF_j(s)$, otherwise $v_i = 0$.

The IDS is defined based on the predefined thresholds for *VMF*, *strength*, and *tolerance*. Here the *strength* is used to reflect how the overall changes of the SMF values from one version to another comply with the increasing pattern. It is the number of changes that follow the increasing change pattern against the total number of times a structure has changed. The value of strength is between 0 and 1. It is obvious that a larger value of strength implies that this structure complies better with the increasing pattern. Since it is possible that some of the changes may not comply with the increasing pattern, *tolerance* is defined to restrict other change patterns. Here *tolerance* is used to confine that maximal significance of decreasing patterns in the history. That is none of the changes in the history of an IDS structure can decreasing beyond the *tolerance*. The value of *tolerance* is also between 0 and 1. A larger value of tolerance implies a more relax constraint of the decreasing pattern. According to the above definition, to be an IDS, the structure should not only change frequently ($V(s) \geq \alpha$) but also certain fraction of the changes should comply with the increasing pattern defined by strength ($strength(s) \geq \beta$). And there should be no changes that decreases beyond the tolerance ($\forall$ SMF_i(s) $\neq$ 0, SMF_i(s)-SMF_j(s) < SMF_i(s)*γ) as well. Similarly, the decreasing dynamic structure is defined as follows.

Definition 4. *[**DDS**] Let $\langle S_1, S_2, \cdots, S_n \rangle$ be tree representations for n versions of XML documents. Let the thresholds for VMF, strength, and tolerance be α, β and γ respectively. A structure $s \preceq S_j$ is a **decreasing dynamic structure (DDS)** in this sequence iff: $VMF(s) \geq \alpha$, $\forall$ $SMF_i(s) \neq 0$, $SMF_i(s)$-$SMF_j(s)$ $< SMF_i(s)*\gamma$, SMF_j is the last nonzero SMF value where $i > j \geq 1$, and $strength(s) \geq \beta$, where strength(s) is defined as: $strength(s) = \frac{\sum_{i=2}^{n} v_i}{(n-1)*VMF(s)}$, where v_i=1, if SMF_i $(s) \neq 0$ and $SMF_i(s) \leq SMF_j(s)$, otherwise v_i=0.*

Given a sequence of historical XML document versions and the user-defined thresholds for VMF, strength, and tolerance, the problem of discovering all the valid IDS and DDS structures is called the *pattern-based dynamic structure* mining problem.

4 Pattern-Based Dynamic Structure Mining

In this section, we present the pattern-based dynamic structure mining algorithm. First, we elaborate on the data model we proposed for storing and representing historical changes for versions of XML documents. Next, we propose the algorithm for extracting the types of pattern-based dynamic structures.

4.1 Data Model

The structure of an XML document can be represented and stored as a tree such as the DOM tree proposed by W3C. However, in our pattern-based dynamic structure mining problem, given a sequence of history XML documents, it is not efficient to store them in a sequence of DOM trees. To make the pattern-based dynamic structure mining process efficient, a flexible data model is desired. We propose a *SMH-Tree* (Structure History Tree) model to store the history of structural changes. The *SMH-Tree* is an extension of the DOM model with some historical properties so that it can compress the history of changes to XML into a single *SMH-Tree*. Formally, we define an *SMH-Tree* as follows:

Definition 5 (SMH-Tree). *An SMH-Tree is a 4-tuple* $H = (N, A, v, r)$*, where* N *is a set of object identifiers;* A *is a set of labelled, directed arcs* (p, l, c) *where* $p, c \in N$ *and* l *is a string;* v *is a function that maps each node* $n \in N$ *to a set of values* (C_n, C_v)*,* C_n *is an integer and* C_v *is a binary string;* r *is a distinguished node in* N *called the root of the tree.*

We now elaborate on the parameters C_n and C_v. The two parameters are introduced to record the historical changes for each substructure. C_n is an integer that records the number of versions that a substructure has changed significantly enough (the structure dynamic is no less the corresponding threshold). C_v is a binary string that represents the historical changes of a substructure. The length of the string is equal to the number of deltas in the XML sequence. The ith digit of the string denotes the change status of the structure from X_i to X_{i+1}, where the value of 1 means this structure changed, the value of 0 means it did not change. In this model, the types of changes are not specified in C_v since we focus on the frequency and significance of the changes. In the SMH-Tree, the C_v value for each structure is finally updated by using the formula: $C_v(s) = C_v(s_1) \vee C_v(s_2) \vee \cdots \vee C_v(s_j)$, where $s_1, s_2, \cdots, s_j$ are the substructures of s.

4.2 Pattern-Based Dynamic Structure Mining Algorithm

Based on the above data model, we design an algorithm to extract the increasing dynamic structure and the decreasing dynamic structure respectively. In the mining process, there are three major phases: *the mapping phase, the extraction phase,* and *the visualization phase.* In this section, we focus on the first two phases of the algorithms since the last phase is straightforward, in which the IDS and DDS structures are represented in the tree representation of the XML document.

SMH-Tree Construction Phase: Algorithm 1 describes the process of SMH-Tree construction. Given a sequence of XML documents, the SMH-Tree is initialized as the structure of the first version. After that, the algorithm iterates over all the other versions by extracting the structural changes and mapping them into the SMH-Tree. The SX-Diff function is a modification of the X-Diff [7] algorithm that generates only the structural change from two different versions of a document. The structural changes are mapped into the SMH-Tree according to mapping rules described in Algorithm 2. The SX-Diff function and the mapping

```
Input:
  $\langle X_1, X_2, \cdots, X_n \rangle$: XML documents sequence
  S(D): The corresponding DTD
Output:
  $H$: SMH-Tree
$H \leftarrow (S(X_1) \cap S(D_2))$
for ($k = 2; k \leq n; k++$) do
   $\triangle$ = SX-Diff($X_k, X_{k-1}$)
   $H$ = Mapping($H, \triangle$) //Algorithm 2
end for
Finalize($H$)
Return($H$)
```

(a) Algorithm 1

```
Input:
  $H$: SMH-Tree
  $\triangle$: Structural delta
Output:
  $H$: The updated SMH-Tree
for all $n_i \in \triangle$ do
   for $n_i \neq null$ do
      update $C_v(n_i)$
      $n_i = n_i$.parent($H$)
   end for
end for
Return($H$)
```

(b) Algorithm 2

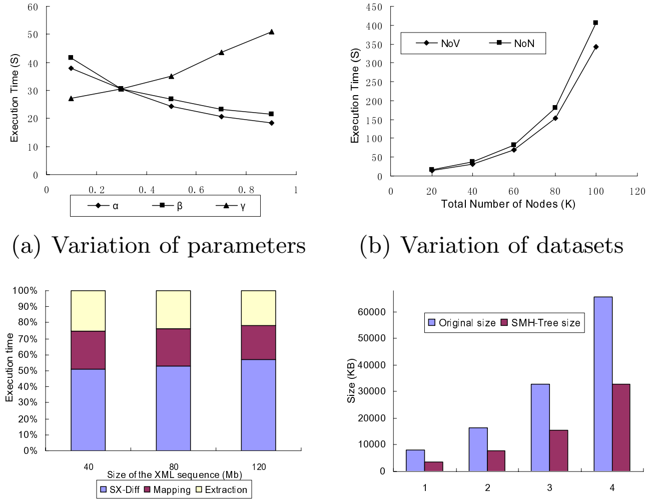
```
Input:
  S: The SMH-Tree
  t: The number of historical XML versions
  $\alpha, \beta, \gamma$: Thresholds for VMF, strength, and tolerance
Output:
  $L$: The list of nodes where all IDS/DDS rooted
for all $n_k$=Top-downTraversal(S)$\neq$ null do
   if $VMF(n_k) \geq \alpha$ then
      for all t $\geq$ i $\geq$ 1 do
         If Any changes beyond the tolerance $\gamma$ break
         else If strength($n_k$) $\geq \beta$ then $L = L \cup n_k$ end if
         end if
      end for
      else prune all descendants of $n_k$
   end if
end for
Return($L$)
```

Algorithm 3

process iterate until no more XML document is left in the sequence. Finally, the SMH-Tree is returned as the output of this phase.

Algorithm 2 describes the mapping function. Given the SMH-Tree and the structural changes, this function is to map the structural changes into the SMH-Tree and return the updated SMH-Tree. The idea is to update the corresponding values of the nodes in the SMH-Tree.

Extraction Phase: In this phase, given the SMH-Tree and predefined thresholds, all the valid pattern-based dynamic structures will be discovered. The extraction phase is solely a traversal of the SMH-Tree. During the traversal, values of the parameters in the definitions are calculated, compared with the predefined values, and the valid pattern-based dynamic structures are returned. Rather than traversal the entire SMH-Tree, the top-down traversal strategy is employed to avoid some unnecessary traversal efforts. Based on Definitions 3 and 4, a pattern-based dynamic structure must have a larger enough VMF value. If the VMF value is less than the predefined threshold, then no further calculation is needed since that structure cannot be a valid pattern-based dynamic structure.



(a) Variation of parameters (b) Variation of datasets

(c) Execution Time (d) Size of the SMH-Tree

Fig. 2. Experiment Results

Algorithm 3 shows the pattern-based dynamic structure extraction algorithm. Since there are three constrains for the valid pattern-based dynamic structures, we check against the three constraints step by step. In each step, a list of candidate pattern-based dynamic structures will be pruned. We first check the VMF value of the structures against the threshold as shown in Line 2. Based on this value, we can prune out some structures for further checking as shown in Line 8. Next, the tolerance criteria is used to further avoid unnecessary calculation for some structures shown in Line 4. Lastly, the value of strength is calculated and compared against the predefined threshold and the pattern-based dynamic structures are incorporated into the final list as shown in Lines 5.

5 Experimental Results

In this section, we evaluate the performance of the pattern-based dynamic structure mining algorithm to demonstrate its efficiency and scalability. The mining algorithm is implemented in Java. All experiments are done on a Pentium IV 1.7 GHz PC with 256 MB RAM running windows 2000. We use synthetic datasets generated from the DBLP and SIGMOD[1].

Figure 2 (a) shows the efficiency of the algorithm using SIGMOD dataset. The average size each version is 468Kb with the average number of nodes is 1120. 20 versions are generated. By varying the predefined thresholds of α, β, and γ, this figure shows how such parameters affect the efficiency of the algorithm. It

[1] http://www.cs.washington.edu/research/xmldatasets

can be observed that as the values of α and β increases, the execution time decreases, while the execution time increases when the value of γ increases. This is because that the number of nodes in the SMH-Tree to be checked in the extraction phase decreases as α and β increases, but as the value of γ increases the constraint becomes more relax and the number of nodes to be checked grows.

Figure 2 (b) shows the scalability of the algorithm using DBLP dataset. The values of α, β, and γ are fixed to 0.20, 0.60, and 0.10 respectively. We increase the total number of nodes in the XML sequence by two methods. One is to increase the average number of nodes (NoN) in each version, another is to increase the total number of version (NoV) in the sequence. We also observed that the increasing of NoN makes the extraction process more expensive compared with the increasing of NoV.

Figure 2 (c) shows the execution time of different phases in the mining process. Three different SIGMOD datasets with the size from 40Mb to 120Mb are used in this experiment. It can be observed that the cost for SX-Diff phase is around 50% of the total cost. Also, as the size of the XML sequence increases, the percentage of SX-Diff cost increases. This is because that XML parsing recurs as a major bottleneck to the overall success of XML related project [6]. Figure 2 (d) presents the compression ratio of our proposed SMH-Tree model. Four different DBLP datasets are used. It shows that the size of the SMH-Tree is only around 40% of the original size of the XML sequence.

6 Conclusion

In this paper, we proposed an approach to extract pattern-based dynamic structures from changes to XML documents. While knowledge extracted from snapshot data is important, it shown that knowledge hidden behind the historical changes is also beneficial for e-commerce applications. Experiments have been conducted to show the efficiency and scalability of our proposed pattern-based dynamic structure mining algorithm.

References

1. A.Termier, M.C.Rousset, and M.Sebag. Treefinder: a first step towards XML data mining. In *Proc. IEEE ICDM*, Pages 450–458, 2002.
2. D. Braga, A. Campi, S. Ceri, and P. L. Lanzi. Discovering interesting information in XML data with association rules. In *Proc. ACM SAC*, Pages 450–454, 2003.
3. L. Chen, S. S. Bhowmick and C. Chia. Mining Association Rules from Structural Deltas of Historical XML Documents. In *Proc. PAKDD*, Pages 452–457, 2004.
4. A. Inokuchi, T. Washio, and H. Motoda. An apriori based algorithm for mining frequent substructures from graph data. In *Proc. PKDD*, Pages 13–23, 2000.
5. M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. IEEE ICDM*, Pages 313–320, 2001.
6. M. Nicola and J. John. XML parsing: a threat to database performance. In *Proc. ACM CIKM*, Pages 175–178, 2003.

7. Y. Wang, D. J. DeWitt, and J.-Y. Cai. X-diff: An effective change detection algorithm for XML documents. In *Proc. IEEE ICDE*, Pages 519–527, 2003.
8. X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. IEEE ICDM*, Pages 721–724, 2002.
9. M. Zaki. Efficiently mining frequent trees in a forest. In *Proc. ACM SIGKDD*, Pages 71–80, 2002.
10. M. J. Zaki and C. C. Aggarwal. XRules: An effective structural classifier for XML data. In *Proc. ACM SIGKDD*, Pages 316–325, 2003.
11. Q. Zhao, S. S. Bhowmick, and S. Madria. Research issues in web structural delta mining. In *Proc. IEEE ICDM Workshop*, 2003.
12. Q. Zhao, S. S. Bhowmick, M. Mohania, and Y. Kambayashi. FCS mining: Discovering frequently changing structures from historical XML structural deltas. *Submitted for publication*, 2004.

Space-Efficient Range-Sum Queries in OLAP

Fredrik Bengtsson and Jingsen Chen

Department of Computer Science and Electrical Engineering
Luleå University of Technology S-971 87 LULEÅ, Sweden

Abstract. In this paper, we present a fast algorithm to answer range-sum queries in OLAP data cubes. Our algorithm supports constant-time queries while maintaining sub-linear time update and using minimum space. Furthermore, we study the trade-off between query time and update time. The complexity for query is $O(2^{\ell d})$ and for updates $O((2^\ell \sqrt[2^\ell]{n})^d)$ on a data cube of n^d elements, where ℓ is a trade-off parameter. Our algorithm improve over previous best known results.

1 Introduction

In a large multidimensional database, aggregate queries can potentially consume enormous amount of time to process, if not properly supported by the database. The responsiveness of both updates to the database and queries from the database can be very poor. This is often very unsatisfactory, for example in an interactive environment, where the user needs a fast response from the database. To support the interactive analysis of large data sets, the On-Line Analytic Processing (OLAP) model [1, 2] has been used and received much attention.

In OLAP, the data consists of records with $d + 1$ attributes. One attribute is the measure attribute and the others are called dimensional attributes. Thus, the database can be regarded as a d-dimensional cube. The data cube [3] is a popular data model for OLAP that allows the implementation of several important queries. Several query types have previously been studied, including range max/min [4, 5] and range-sum [6–10, 1].

In this paper, we study the orthogonal range-sum query on the data cube. Ho et al. [11] introduced the prefix sum technique for fast range-sum queries. The update time, however, is $O(n^d)$ for data cubes of n^d elements. Nevertheless, the prefix sum techniques provides a base for further improvements [9, 6–8, 10]. Poon [12] has a very efficient structure with a query time of $O((2L)^d)$ and update time of $O((2Ln^{(1/L)})^d)$, where L is a parameter. The structure uses $O((2n)^d)$ storage. Geffner et al. [8] have a structure with with a query complexity of $O(2^d)$ and an update complexity of $O((n^{d/2})^d)$. The space usage in this structure is also super linear. Chun et al. [6] presents a structure which uses a standard prefix sum and accumulates updates in an R-tree. Their structure has, according to their simulations, good performance, but the space usage is super linear. All the above-mentioned structures uses extra space in order to achieve high performance for both lookups and queries.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 87–96, 2004.

However, a structure saving space is the *Space Efficient Relative Prefix Sum* by Riedewald et al. [9]. It does not use any extra space, but has the same performance as the *Relative Prefix Sum.* Another structure that does not use extra space is the *Space Efficient Dynamic Data Cube* by Riedewald et al. [9]. It has logarithmic search complexity and logarithmic update complexity.

An interesting special case is when the data cube is considered to be sparse, and thus, the space savings could be huge if the sparsity is taken into account. This have been studied by Chun et al. [7]. They propose a technique called *pc-pool* for sparse cubes. They demonstrate the performance of the pc-pool by simulations.

Our results does not use any extra space, has constant query complexity while maintaining a sub-linear time update. Furthermore, our results provides superior query-update trade-off over previous known structures.

The rest of the paper is organized as follows. Section 2 introduces the prefix sum cube and methods which are closely related to our structure. We present our results together with their performance analysis in Section 3. The next section will investigate the Space Efficient Dynamic Data Cube structure further. First, we will analyse the complexity of the Space Efficient Dynamic Data Cube which are omitted in the original paper.[9]. Then, a study of the query-update complexity trade-off for the Space Efficient Dynamic Data Cube is also pursued. Section 5 presents some further improvements to our structure and Section 6 concludes the paper.

2 Prefix Sum Methods

This section will introduce methods that are closely related to our structure. All those structures are based on the prefix sum [11]. Our work is a generalization of both the Space-Efficient Relative Prefix Sum [9] and the Space-Efficient Dynamic Data Cube [9]. The Space-Efficient Relative Prefix Sum can be derived as a special case of our method.

Consider the two extremes: the original array and the prefix sum array. A query in the orthogonal array takes $O(n)$ time, in the worst case. An update, on the other hand, requires only $O(1)$ time. For an array of prefix sums, it's the opposite. Different techniques combines the best of those two approaches.

In the Relative Prefix Sum (RPS) [8], the data set is represented by two arrays; the *Overlay* array (l) and the *Relative prefix* array (r). Each array is divided in $\sqrt{n_i}$ blocks of $\sqrt{n_i}$ elements each, along dimension i, for $i = 0, 1, \ldots, d-1$.

In Fig. 1, a simple one dimensional example of the RPS is shown, where $n = 16$ and the block boundaries have been emphasized with thick lines.

In r, the prefix sum relative to each block is stored.

In l, the first element of each block stores the sum of all elements (from the data cube) of all previous blocks. In d dimensions, the situation is slightly more complicated, but the idea is the same. Thus, there is no need to rebuild the whole array on update. It is sufficient to rebuild the block of r and all the elements in l, both of which have a size of $O(\sqrt{n})$.

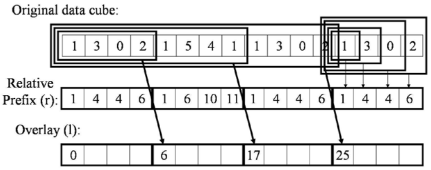


Fig. 1. Simple 1-dim RPS

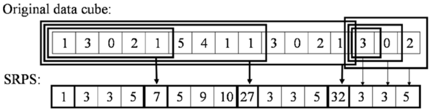


Fig. 2. Simple 1-dim SRPS

The Space-Efficient Relative Prefix Sum (SRPS) by Riedewald et al. [9] is a structure very similar to the RPS, but the space usage is smaller. Consider the one-dimensional RPS of Fig. 1 again. If the elements in the overlay array contains the inclusive sum (instead of the exclusive), we need only store a shorter local prefix sum in the RPS array. It is sufficient that the local prefix sum in the RPS array starts at the 2:nd position in the block (if the overlay sums are inclusive). The first position in the local prefix sum in the RPS array can then be used to store the overlay array. This way, the structure does not use more space than the original data cube (Fig. 2). Search and update is performed similar to the RPS.

In the Space-Efficient Dynamic Data Cube [9], the technique of the SRPS is applied recursively on each block and on the boundary cells of each block. However, instead of splitting the array in $\sqrt{n_i}$ blocks, each dimension is split in 2 (or any other constant). In general, we get 2^d, boxes for a d-dimensional cube. Now, call the region corresponding to the prefix sum of a block, the *inner region* of each block. The inner region, in d dimensions, is all elements in a box, except for the first element in *each dimension.*

The inner region is a prefix sum, but consider the data cube that would have generated the prefix sum (the pairwise difference of the prefix sum). It is possible to recursively apply the SDDC algorithm to that data. Instead of storing the prefix sum in the inner region, we store the data from the recursive application of the SDDC. The recursion continues $O(\log n)$ levels.

Now, consider the border cells of a box (cells that are not the inner region – the first cell of each block, in the one-dimensional case). Observe that, in the one-dimensional case, the border of each block has dimension 0 (it is a scalar). In general, for d dimensions, the border cells contains $\binom{d}{k}$ regions of k dimensions. Each element (border cell) is the sum of all elements in the data cube with smaller index. Each such sum span $d - k$ dimensions. For each region of the border of each box, the SDDC algorithm (of the dimensionality of the region) is applied recursively. The elements are stored in the same place as we would have done without recursion, thus, throwing away the old elements (from before the recursion). Queries and updates are handled similar to the SRPS, but recursively.

This way, we get a structure (and a corresponding algorithm) requiring $O(\log^d n)$ array accesses for both update and query. See Section 4 for a performance analysis, which is omitted in the original paper [9].

3 Our Result

Our structure for range sum queries combine the techniques from both the SRPS and the SDDC. We use a recursive storage technique to obtain fast range sum queries (constant time) and to achieve good trade-off between queries and updates. In essence, instead of storing the elements from smaller subproblems that appear in the structure directly, our algorithm is applied recursively and the result from the recursion is stored. This results in a query performance of $O\left(2^{id}\right)$ while maintaining an update complexity of $O\left((2^i \sqrt[2^i]{n})^d\right)$, where i is a trade-off parameter. This improves over previous known structures, in the asymptotic sense.

3.1 Constant Query Time

The query time of $O\left(2^{id}\right)$ is constant with respect to n for any i and d. Thus, it is possible to achieve very good update performance while still maintaining constant lookup time. We will first present the general idea and then continue with a detailed presentation of the two dimensional case followed by the general case.

The algorithm splits each dimension in $\sqrt{n}$ blocks (each with side-length $\sqrt{n}$). Consider Fig. 4. We make the important observation that it is not only possible to apply the algorithm recursively *within* each block, but that all border regions of k dimensions together with the corresponding border regions from other cells in $d-k$ dimensions form a subproblem of size $O(\sqrt{n})$ and dimensionality d. See Fig. 4 for a 2-dim example of the recursion technique. Here, one row of 1-dim. subproblems (a) form a new 2-dim. subproblem. In the same way, one column of 1-dim. subproblems (b) form a new 2-dim. subproblem. Additionally, the first element of all blocks (0-dim) form a new 2-dim. subproblem (c). Thus, we only have to recurse over subproblems of the same dimensionality. In this way, we can not only describe our fast algorithm simply, but perform the complexity analysis of the algorithm neatly.

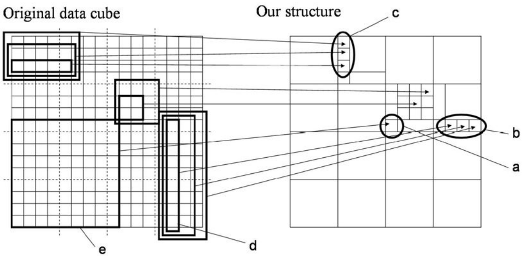


Fig. 3. 2-dimensional example

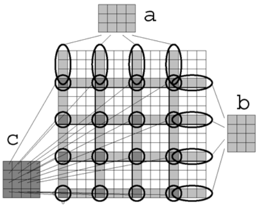


Fig. 4. Recursion in a 2 dimensions

Consider the one-dim. example of Figure 2 again. At just one recursion level, our structure would be the same as the SRPS. However, at two recursion levels, the structure would be formed in each of the inner regions as well (in order from left: $[3, 3, 5]$, $[5, 9, 10]$, $[3, 3, 5]$ and $[3, 3, 5]$). Unfortunately, it takes a much larger example to actually form a two-level structure. Each border region (in order from left: [1], [7], [27] and [32]) together form a subproblem. The structure is recursively applied to them as well. This would yield the vector $[1, 6, 27, 5]$. This vector would then be stored instead of the border elements. The structure is applied to the pairwise difference of the elements ($[1, 6, 20, 5]$), not the elements themselves. Unfortunately, the inner regions in this very small example becomes only one element (the 6 and the 5).

Let d be the dimension of the data cube and $n_i, 0 \leq i \leq d-1$ be the size of the data cube for dimension i. Furthermore, let $e_{(x_0,x_1,\ldots,x_{d-1})}$ denote the elements of the original data cube, where $0 \leq x_i \leq n_i - 1$. Similarly, let $g_{(x_0,x_1,\ldots,x_{d-1})}$ denote the elements of our data structure. Next, we will present the data structure in detail. First, a two-dimensional structure is presented, where after the general case is introduced.

Consider a single block $b_{(x,y)} = g_{(k_0\lceil\sqrt{n_0}\rceil+x, k_1\lceil\sqrt{n_1}\rceil+y)}$, $x \in \left[0, \left\lceil\sqrt{n_o}\right\rceil - 1\right]$, $y \in \left[0, \left\lceil\sqrt{n_1}\right\rceil - 1\right]$ for any k_0 and k_1 (such that the block is within the data cube) of an n_0 by n_1 structure of two dimensions. Then $b_{(0,0)}$ (marked "a" in Fig. 3) will contain the sum of all elements of the original data cube that has smaller index than both x and y (the elements to the lower left, marked "e", in Fig. 3). Next, consider the elements $\left\{b_{(x,0)} : x \in [1, \lceil\sqrt{n}\rceil - 1]\right\}$ (marked "b" in Fig. 4). The element $b_{(1,0)}$ will contain the sum of all elements from the original data cube with smaller y-index than $b_{(1,0)}$, but with the same x-index. This is shown with "d" in Fig. 3. Observe that it is not the elements of the structure that is summed, but the elements of the original data cube. In the same way, $b_{(2,0)}$ contains the sum of the elements with smaller y-index, but with $x = 2$ plus $b_{(1,0)}$. This holds in general. The fact that $b_{(x,0)} = b_{(x-1,0)} + e_{(k_0\lceil\sqrt{n_0}\rceil+x, k_1\lceil\sqrt{n_1}\rceil)}$ makes the elements a prefix sum. The elements $\left\{b_{(0,y)} : x \in \left[1, \left\lceil\sqrt{n_0}\right\rceil - 1\right]\right\}$ (marked "c" in Fig. 3) stores elements in the same way, but with coordinates swapped.

We observe that each block in the two-dimensional case contains one prefix sum of two dimensions (white in Fig. 4), two prefix sums of one dimension (light grey in Fig. 4) and one prefix sum of zero dimension (a scalar, dark grey in Fig. 4). In general, for d dimensions, each block will contain $\binom{d}{k}$ prefix sums of k dimensions. This can be realized from an algebraic point of view. The elements for the prefix sums within a single block can be obtained by fixing selected dimensions to 0 (relative to the block) and letting the other dimensions vary to span the region of the prefix sum. If we choose to fix k of the dimensions, it can be done in $\binom{d}{k}$ ways.

Consider an arbitrary block

$$b_{(i_0,i_1,\ldots,i_{d-1})} = g_{\left(k_0\lceil\sqrt{n_0}\rceil+i_0, k_1\lceil\sqrt{n_1}\rceil+i_1, \ldots, k_1\left\lceil\sqrt{n_{d-1}}\right\rceil+i_{d-1}\right)}$$

of our structure, where $k_0, k_1, \ldots, k_{d-1}$ selects block. Then we have that

$$b_{(i_0,i_1,\ldots,i_{d-1})} = \sum_{j_0=l_0}^{h_0} \sum_{j_1=l_1}^{h_1} \cdots \sum_{j_{d-1}=l_{d-1}}^{h_{d-1}} e_{(j_0,j_1,\ldots,j_{d-1})}$$

where $i_j = 0 \Leftrightarrow l_j = 0, h_j = k_j \left\lceil \sqrt{n_j} \right\rceil$
and $i_j > 0 \Leftrightarrow l_j = k_j \left\lceil \sqrt{n_j} \right\rceil + 1, h_j = k_j \left\lceil \sqrt{n_j} \right\rceil + i_j$ for $j \in [0, d-1]$
iff $\exists i \in \{i_o, i_1, \ldots, i_{d-1}\} : i = 0$.
The last condition restricts the validity of the formula to the borders of the block (where at least one index is zero). For the rest of the block (the local prefix sum, no indices are zero) we have that

$$b_{(i_0,i_1,\ldots,i_{d-1})} = \sum_{j_0=a_0+1}^{a_0+i_0} \sum_{j_1=a_1+1}^{a_1+i_1} \cdots \sum_{j_{d-1}=a_{d-1}+1}^{a_{d-1}+i_{d-1}} e_{(j_0,j_1,\ldots,j_{d-1})}$$

for $i_j \in [1, \left\lceil \sqrt{n_j} \right\rceil - 1], a_j = k_j \left\lceil \sqrt{n_j} \right\rceil, j \in [0, d-1]$.

The above discussed structure is a one-level recursive structure ($i = 1$, see analysis).

Now, we present the recursion for the general case. Consider Fig. 4. Within each block (the inner region), the algorithm can be applied recursively. Consider the prefix sum (white elements) of the inner region of a block. The algorithm could be applied to the original data cube that corresponds to the prefix sum (the pairwise difference). Consider element $(0, 0)$ of *all* blocks (the dark grey elements). They also represent a prefix sum and GRPS can be applied recursively to their pairwise difference. To see this, remember that each of those elements contains the sum of all elements from $e_{(0,0)}$ up to $b_{(0,0)}$ (in two dimensions). Thus, element $(0, 0)$ from *all* blocks together forms a 2-dimensional prefix sum. Similar reasoning can be applied to elements $\{b_{(x,0)} : x \in [1, \left\lceil \sqrt{n_0} \right\rceil]\}$ and $\{b_{(0,y)} : y \in [1, \left\lceil \sqrt{n_1} \right\rceil]\}$ (light grey in Fig. 3). One row (or one column) of such elements form a two dimensional prefix sum and the GRPS can be applied recursively to those sums.

Instead of storing the elements of the structure directly, we store the elements of the recursive application of the algorithm. This is possible since the structure does not require extra space, and therefore, a recursive application does not increase the space requirement. Observe that the size of each dimension of the subproblems is $O\left(\sqrt{n_i}\right)$. In general, the subproblems consists of the elements from the data cube that can be obtained by varying the index *within* blocks for certain dimensions and varying *the block* for the other dimensions. As an example, we show a single box for our structure in 3 dimensions in Fig. 5.

For an update, on a one-level structure, it is necessary to rebuild one block and all of the affected border regions (regions that include the updated element in their sum). However, for a multi-level recursive structure, it is only necessary to update the smaller block in the recursive application and the smaller border regions. A query can be processed fairly simple: The prefix sum (or the range-sum) can be computed by following the recursive structure and adding the appropriate elements.

With the above compact recursive structure, our algorithm achieve the constant time range sum queries.

Theorem 1. *The algorithm, described above, has a range-sum query complexity of $O(2^{id})$, where i is the recursion depth and d is the dimension of the data cube.*

Proof. Let $T_d(n,i)$ denote the query complexity. Clearly,

$$T_d(n,i) = \begin{cases} \sum_{j=0}^{d} \binom{d}{j} T_d\left(\sqrt{n}, i-1\right) & \text{, if } i > 0 \\ 1 & \text{, if } i = 0 \end{cases} .$$

For $i > 0$, we have

$$T_d(n,i) = \sum_{k=0}^{d} \binom{d}{k} T_d\left(\sqrt{n}, i-1\right) = 2^d T\left(\sqrt{n}, i-2\right) = 2^{id} .$$

At the same time, our algorithm requires sub-linear time for update operations while maintaining constant range-sum queries.

Theorem 2. *The algorithm, described above, has an update complexity of* $O(2^{di}\left(\sqrt[2^i]{n}\right)^d)$*, where* i *is the recursion depth and* d *is the dimension of the data cube.*

Proof. Let $U_d(n,i)$ denote the update time. Hence, from the description of the algorithm we have, $U_d(n,i) = 2U_d(\sqrt{n}, i-1)$, if $i > 0$, and $U_d(n,i) = n^d$, if $i = 0$. For $i > 0$, we have $U_d(n,i) = \sum_{k=0}^{d} \binom{d}{k} U_d\left(\sqrt{n}, i-1\right) = 2^d T\left(\sqrt{n}, i-2\right) = 2^{id}\left(\sqrt[2^i]{n}\right)^d$.

3.2 Update-Query Time Trade-Off

Notice that the time complexity of range sum query and of update depends not only on the data size n^d, but also on the number of recursion levels, i, of the data structure (parameter). By choosing the parameter i, we obtain different trade-off between the running time of queries and updates. Hence, i can be chosen to tailor the performance of the structure to the needs of the user. If queries are much more common than updates, probably a constant query time is desirable. If $i = O(1)$, then the query complexity is always constant (with respect to n), but the update complexity is $O(\sqrt[2^i]{n^d})$ with respect to n.

If, on the other hand, the maximum recursion depth possible, is chosen, we get a query complexity of $O(\log^d n)$ with respect to n and an update complexity of $O(\log^d n)$ with respect to n, which is the same as the complexity of the SDDC [9]. Therefore,

Corollary 1. *If the trade-off parameter,* $i = O(\log\log n)$*, both the update and range-sum complexity becomes* $O(\log^d n)$ *(This matches the SDDC). If the trade-off parameter* $i = l$*, for some constant* l*, the range-sum complexity is* $O(1)$ *and the update complexity is* $O(\sqrt[2^i]{n^d}) = o(n^d)$*.*

3.3 Comparison to Previous Structures

Our structure has better asymptotic query-update trade-off compared to previous structures. To our best knowledge, the best previously known constant-query-time structure that does not require extra memory is the SRPS that has an update complexity of $O(\sqrt{n^d})$. For constant query complexity, our structure has $O(\sqrt[2^i]{n^d})$ update complexity. The SDDC, with our proposed trade-off has a query complexity of $O(n^d)$, for constant query complexity. The SDDC was designed with query and update complexities of $O(\log^d n)$. This can be, asymptotically, matched by our structure by choosing the trade-off parameter, $i = \log\log n$.

Additionally, our structure will be easier to parallelize, because the data cube is divided by $O(\sqrt{n})$ instead of blocks of constant size.

4 Analysis of SDDC

In this section, we will present an analysis of the SDDC algorithm [9]. Furthermore, we will introduce a trade-off parameter for the SDDC, which is not introduced in the original SDDC. It is important to observe that the trade-off parameter is the recursion depth in both our structure and the SDDC. Our structure, however, requires a lower recursion depth to achieve the same performance.

Theorem 3. *The range-sum complexity of the SDDC is $\Omega(i^d)$, where i is the recursion depth and d is the dimension of the data cube.*

Proof. Let $T_d(n,i)$ denote the range-sum complexity of SDDC. Since the SDDC algorithm reduce the problem size by a factor of k after each recursion step, we have $T_d(n,i) = \sum_{j=0}^{d} \binom{d}{j} T_j(n/k, i-1)$, if $i > 0$ and $d > 0$ and $T_d(n,i) = 1$, if $d = 0$ or if $i = 0$. We prove that $T_d(n,i) \geq i^d$ by induction on d. For $d = 1$, we know that $T_1(n,i) = i + 1 \geq i$.
Assume that $\forall d' < d : T_{d'}(n,i) \geq i^{d'}$. Hence, $T_d(n,i) = \sum_{j=0}^{d} \binom{d}{j} T_j(n/k, i-1) \geq \sum_{j=0}^{d-1} \binom{d}{j} (i-1)^j + T_d(n/k, i-1) = (i-1+1)^d - (i-1)^d + T_d(n/k, i-1) = i^d + (i-i)^d + T_d(n/k^i, i-i) = i^d + 1 \geq i^d$

The update time of the SDDC algorithm is as follows.

Theorem 4. *The update complexity of the SDDC is $\Omega(n^d/k^{di} + k^d i^d)$, where i is the recursion depth and d is the dimension of the data cube.*

Proof. Let $T_d(n,i)$ denote the update complexity of SDDC. Since the SDDC algorithm reduce the problem size by a factor of k after each recursion step, we have $T_d(n,i) = \sum_{j=0}^{d} \binom{d}{j} T_j(n/k, i-1) k^{d-j}$, if $i > 0$ and $d > 0$ and $T_d(n,i) = n^d$, if $i = 0$. We prove that $T_d(n,i) \geq (n/k^i)^d + (ki)^d$ by induction on d. For $d = 1$, we have $T_1(n,i) = k + T_1(n/k, i-1) = 2k + T_1(n/k^2, i-2) = ik + \frac{n}{k^i}$. Assume that $\forall d' < d : T_{d'}(n,i) \geq (n/k^i)^{d'} + (ki)^{d'}$. Hence, $T_d(n,i) = \sum_{j=0}^{d} \binom{d}{j} T_j(n/k, i-1) k^{d-j} = \sum_{j=0}^{d-1} \binom{d}{j} T_j(n/k, i-1) k^{d-j} + T_d(n/k, i-1)$. Consider the first term in this equation. We have

$$\sum_{j=0}^{d-1}\binom{d}{j}T_j(n/k,i-1)k^{d-j} \ge \sum_{j=0}^{d-1}\binom{d}{j}\left(\left(\frac{n}{k^{i-1}}\right)^j + k^j(i-1)^j\right)k^{d-j} =$$

$$= k^d\sum_{j=0}^{d-1}\binom{d}{j}\left(\left(\frac{n}{k^i}\right)^j (i-1)^j\right)$$

$$= k^d\left(1+\frac{n}{k^i}\right)^d - \left(\frac{n}{k^{i-1}}\right)^d + k^d(1+i-1)^d - k^d(i-1)^d =$$

$$\left(k+\frac{n}{k^{i-1}}\right)^d - \left(\frac{n}{k^{i-1}}\right)^d + k^d i^d - k^d(i-1)^d \ge k^d i^d - k^d(i-1)^d \ .$$

By telescoping, we obtain $T_d(n,i) \ge k^d(i^d - (i-1)^d) + T_d\left(\frac{n}{k^i}, i-i\right) = k^d i^d + \left(\frac{n}{k^i}\right)^d$.

5 Further Improvements

The memory access of our structure (and other similar structures) are quite scattered, which could lead to poor performance on cache systems or on secondary storage.

We can overcome most of this problem by reorganizing the memory layout of the algorithm. In our structure, we suggest storing all the border cells from blocks associated with one recursion call in the same place. This is automatically performed for the inner cells (since they already are clustered into the inner cells). Instead of storing the border cells in their respective block, we suggest storing the border cells first (at low index) and then storing all the inner cells in blocks that are one element smaller than before (because of the missing border cells). This should be performed for each recursion level. This way, the data access becomes more concentrated to the same region. See Figure 6 for a two dimensional example.

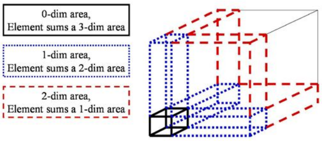

0-dim area,
Element sums a 3-dim area
1-dim area,
Element sums a 2-dim area
2-dim area,
Element sums a 1-dim area

Fig. 5. Single box of 3-dimensional structure

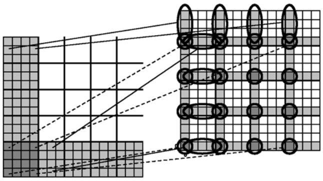


Fig. 6. Efficient space localization

6 Conclusions

We have presented a space-efficient, range-sum data structure and algorithm for use on data cubes. The query performance of the structure is $O\left(2^{id}\right)$ while the update performance is $O\left((2^i \sqrt[2^i]{n})^d\right)$. This provide better query-update trade-off than previously known structures. The trade-off can easily be tuned via the trade-off parameter i.

We also presented a way to improve the space-localization of the structure, so that access to the data cube becomes less scattered.

It would be interesting to investigate if it is possible to achieve a better update-query trade-off. It would also be interesting to investigate the existence of a structure with good space-time trade-off.

References

1. Ho, C.T., Bruck, J., Agrawal, R.: Partial sum queries in olap data cubes using covering codes. IEEE Transactions on Computers **47** (1998) 1326–1340
2. Codd, E.: Providing OLAP (on-line analytical processing) to user-analysts: an it mandate. Technical report, E.F. Codd and Associates (1993)
3. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In: Proceedings of 12th International Conference on Data Engineering. (1996) 152–159
4. Li, H.G., Ling, T.W., Lee, S.Y.: Range-max/min query in OLAP data cube. In: DEXA 2000. Volume 1873 of LNCS. (2000) 467–476
5. Li, H.G., Ling, T.W., Lee, S.Y.: Hierarchical compact cube for range-max queries. In: Proceedings of the 26th VLDB Conference. (2000)
6. Chun, S.J., Chung, C.W., Lee, J.H., Lee, S.L.: Dynamic update cube for range-sum queries. In: Proceedings of the 27th VLDB Conference. (2001)
7. Chun, S.J., Chung, C.W., Lee, J.H., Lee, S.L.: Space-efficient cubes for OLAP range-sum queries. Technical report, Korean Advanced Institute of Science and Technology (2002)
8. Geffner, S.P., Riedewald, M., Agrawal, D., Abbadi, A.E.: Data cubes in dynamic environments. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering (1999)
9. Riedewald, M., Agrawal, D., Abbadi, A.E., Pajarola, R.: Space efficient data cubes for dynamic environments. In: Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DaWak). (2000) 24–33
10. Riedewald, M., Agrawal, D., Abbadi, A.E.: Flexible data cubes for online aggregation. In: ICDT. LCNS (2001) 159–173
11. Ho, C., Agrawal, R., Megiddo, N., Srikant, R.: Range queries in OLAP data cubes. In: Proceedings of the ACM SIGMOD. (1997) 73–88
12. Poon, C.K.: Orthogonal range queries in OLAP. In: Proceedings of the 8th Internatioal Conference on Database Theory (2001). (2001) 361–374

Answering Approximate Range Aggregate Queries on OLAP Data Cubes with Probabilistic Guarantees

Alfredo Cuzzocrea[1], Wei Wang[2,*], and Ugo Matrangolo[3]

[1] DEIS Dept. – University of Calabria 87036 Rende, Cosenza, Italy
cuzzocrea@si.deis.unical.it

[2] School of Computer Science and Engineering – University of New South Wales
2052 Sydney, Australia
weiw@cse.unsw.edu.au

[3] ICAR Inst. – CNR, 87036 Rende, Cosenza, Italy
matrangolo@icar.cnr.it

Abstract. Approximate range aggregate queries are one of the most frequent and useful kinds of queries for Decision Support Systems (DSS). Traditionally, sampling-based techniques have been proposed to tackle this problem. However, its effectiveness will degrade when the underlying data distribution is skewed. Another approach based on the outlier management can limit the effect of data skew but fails to address other requirements of approximate range aggregate queries, such as error guarantees and query processing efficiency. In this paper, we present a technique that provide approximate answers to range aggregate queries on OLAP data cubes efficiently with theoretical error guarantees. Our basic idea is to build different data structures for outliers and the rest of the data. Experimental results verified the effectiveness of our proposed methods.

1 Introduction

Modern advanced information systems, such as B2B and B2C *e*-commerce systems, transactional information systems and control systems, stored huge amount of highly heterogeneous data. Consequently, the use of a Data Warehouse Server (DWS) for collecting, integrating, and making business-critical information available in a "transparent-for-users" way has become mandatory. The enormous size of the multidimensional data presents a main challenge of the data analysis task: it is time-consuming to process queries over large (often tera-bytes or even peta-bytes) multidimensional data sets. Fortunately, the analysis required by decision support-oriented tasks is often qualitative, as in the querying and reporting phases approximate an-

[*] This work was conducted in part when the author was with Hong Kong University of Science and Technology. It is partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region (HKUST6184/02E).

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 97–107, 2004.

swers are often sufficient. In other words, managers and analysts are more interested in the "trend" analysis rather than in the "punctual" analysis.

Range aggregate queries are one of the most typical and important forms of queries for DSS-based applications, as they are fundamental to many data analysis tasks. Range aggregate queries [7] apply an aggregation SQL operator, e.g., SUM, over a set of selected contiguous ranges in the domains of the dimensional attributes. Usually, such queries are resource-intensive as they have high computational overheads in terms of temporal and spatial needs. Sampled data can be used effectively instead of original data without significantly compromising the accuracy of analysis like OLAP. Besides this, sampling-based techniques have gained popularity as they have a very low computational cost. However, when data distributions are skewed, the accuracy of the approximate answers based on sampling can degrade significantly, as some outliers (i.e., values significantly distant from the mean value) might have been sampled. Managing outliers separately from non-outlier data is an effective solution to such limitation. However, previous work only focused on the choice of the outliers and did not address the important issue as how to organize those outliers to support efficient approximate query processing.

In this paper we address the issue of providing fast approximate answers to range queries on OLAP data cubes. We improve the outlier-based approach by enriching it with theoretically-based probabilistic guarantees. Our basic idea is to separate outliers from the rest of the data and manage them using different data structures. Specifically, we propose a novel technique that aims to build the "workload-adaptive" tree-like synopsis, called **T**unable **P**artition-*Tree* (**TP**-*Tree*), that is used to organize samples from different partitions of the original *non-outlier* data. A quad-tree based index is proposed to organize carefully chosen outliers to provide fast response time during approximate query processing. We experimentally evaluated the proposed technique with other well-known techniques based on multi-dimensional histograms, including *GenHist* [5], *EquiDepth* [12], and *MHist* [13]. Our experimental results demonstrate that the proposed technique outperforms those cited.

2 System Overview

Fig. 1 shows an overview of our solution to approximate range aggregate query processing. Given the input data from an OLAP server, we separate outliers from non-outliers and manage them differently: for outliers that are identified by the Outlier Detection module, they are organized into a quad-tree like index within a given space limit; for non-outliers, uniform samples are extracted and preserved in a novel tree-like synopsis: the **TP**-*Tree*. The organization of the **TP**-*Tree* is updated in a batch mode by monitoring query-workload in a time window.

The approximated answer to a range aggregate query is calculated by summing up the answers from probing the index of the outlier and the index of the non-outliers (i.e., the **TP**-*tree*). A tighter theoretical guarantee can be obtained because errors due to "problematic" outliers are no longer existent.

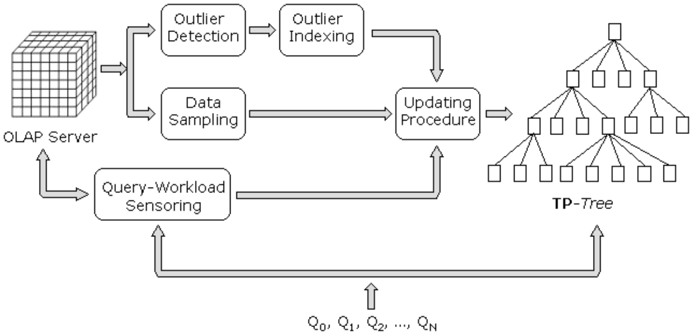


Fig. 1. Technique overview

In the rest of the paper, we assume the MOLAP storage model, which is more efficient for many OLAP operations than the ROLAP model. However, our techniques can be easily generalized to the ROLAP context by preprocessing.

3 Outlier Detection, Indexing and Querying

Outlier detection and management strongly depend on: (*i*) storage space *B* available for housing them and (*ii*) strategy for generating them. Traditionally, outlier detection has been widely investigated in statistics [1,2] and data mining [9,10] and, more recently, has been considered as an efficient method for improving the degree of accuracy of the answers in the context of approximate query processing.

We adopt a technique very similar to that proposed by Chaudhuri et al. in [3] that it exploits the proper outlier definition. Compared with [3], our approach also starts from the detecting outliers, but differs in that we also define a quad-tree-based indexing data structure that allows us to efficiently access the outlier set in the query time. Indexing the outliers is an important issue as otherwise it could heavily degrade performances of the Approximate Query Engine. The major benefit of managing outliers separately is that the overall approximation error is mitigated as the error due to the outliers is zero (assuming enough memory is given), and the quality of the answer depends only on the sampling of the relatively uniform non-outliers. In detail, our solution consists of the following steps:

1. Detection of the (whole) outlier set Ω inside the data cube *- Determine all the outliers of the data cube that deviates Y_m by at least δY_m, (Y_m is the absolute average value of the data cells and δ is a positive factor used for ranging the outlier number). Compared to* [1], *this method is more efficient yet highly effective for most real datasets.*

2. Extraction of a subset Ω_B from Ω in dependence on the available memory size B *- Extract a subset Ω_B from the set Ω depending on the available memory size B: if $|\Omega| \leq B$ then $\Omega_B = \Omega$; otherwise, determine the subset Ω_B by minimizing the standard deviation of its complement set: $A \backslash \Omega_B$. This is the optimal outlier subset according to* [1].

Efficient indexing of the outlier set is our main aim: the process of evaluating the query over this set must not overwhelm the entire querying phase. In addition, it is not unreasonable to assume that when the size of data cubes is enormous, which happens very often in the context of DSS-based applications, the size of the outlier set will be proportionally large. This intuition is a valuable improvement over [3].

We propose a fast and efficient multi-resolution indexing scheme based on a *multi-dimensional quad-tree*, as it is very suitable for indexing spatial data [4]. Given a d-dimensional spatial data set D, a multidimensional quad-tree built on it, called $QT(D)$, is a tree-like index such that each node can have almost 2^d children nodes (see Fig. 3). The $QT(D)$ building task is based on superimposing a multi-resolution grid on the d-dimensional space. In such a way, each level l of $QT(D)$ represents a partition $P(l)$ of D in $|P(l)|$ buckets, such that each bucket b of $P(l)$ is recursively split into a number r of other buckets, with r belonging to the set $\{0,..,2^d\}$. In our implementation, each node w of the quad-tree contains the following information: the bounds of the multi-dimensional region (i.e., the bounds of the corresponding bucket) it represents, the sum of the outliers contained in this region, and the pointers to its children nodes if w is not a leaf node. If w is a leaf node, we substitute these last pointers with the logical addresses of the disk blocks containing the outliers indexed by w.

Given the outlier set Ω_B constructed in the outlier detection phase, we can incrementally build the multi-dimensional a corresponding quad-tree $QT(\Omega_B)$, by posing the following bound over the depth value l_{max} of $QT(\Omega_B)$:

$$l_{\max} \leq \frac{1}{d}\log_2 \frac{s|\Omega_B|}{F} \qquad (1)$$

where s is the size of a single outlier (expressed in bytes) and F is the overall amount of bytes that can be stored in a single disk block. This heuristic implies that the outliers contained in the bucket represented by a leaf node are mostly likely to be contained in a single disk block. Furthermore, in order to minimize the overall computational overheads due to the possible great fan-out of each node of $QT(\Omega_B)$, nodes representing buckets without outliers are *not* store in $QT(\Omega_B)$, so that both the space and time complexity can be reduced to a minimum.

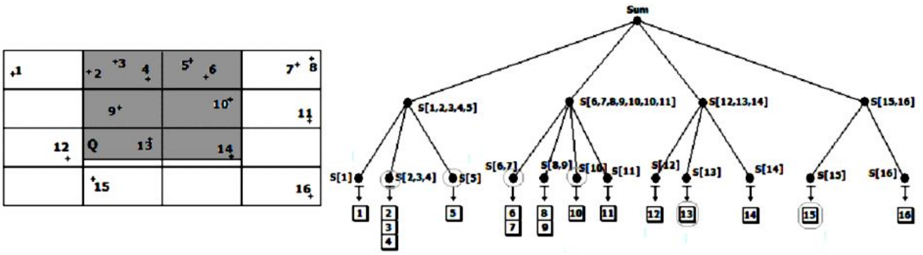


Fig. 2. The index $QT(\Omega_B)$ for a 2-D data cube

We assume that the resulting $QT(\Omega_B)$ can be maintained in main memory. This allows a multi-resolution approach in the evaluation of a given query Q on Ω_B; this approach can save a lot of disk I/Os by simply using the pre-calculated sum of outliers values associated with each index node. The algorithm `evalQueryOnQTNode` evaluates a query Q on a node w of the quad-tree $QT(\Omega_B)$ and given a query Q on a data cube A, the portion of answer due to the outliers of A is obtained by calling the previous algorithm on the root node of $QT(\Omega_B)$.

```
Algorithm evalQueryOnQTNode(Q,w)
  sum = 0;
  if ($w.b \cap Q \neq \emptyset$) then
    if ($w.b \subseteq Q$) then
      sum = sum + w.sum;
    else
      if (w.list().size == 0) then
        sum=sum + readOutliers(w.diskBlock);
      else
        for each node c in w.list() do
          sum = sum + evalQueryOnQTNode(Q,c);
  return sum;
```

wherein: (*i*) `w.b` is the bucket *b* of the node *w*; (*ii*) the procedure `list`, called on a node *w*, returns the list of the child nodes of *w*; (*iii*) the procedure `readOutliers` takes the disk block indexed by a given node *w* `w.diskBlock`, and returns the sum of the outliers contained in it.

4 TP-*Tree*: Managing Samples Adaptively

A fundamental issue that must be faced in the context of approximate multi-dimensional range query processing by pre-computation is determining a partition of multi-dimensional data cubes. In fact, finding a set of "typical" queries as the basis to drive the synopses building task is the core issue of such techniques. This derives from the assumption claiming that typical queries must be defined starting from some "probable" partition, according to the particular data domain.

We propose a different solution: the *tunable workload-aware* partitioning, on the basis of which, given a data cube A, its "current" partition Φ_k depends on the query-workload against it. In such a way, in a certain temporal set starting at a given time t_S and having size M, with M>0, $\{t_S,t_{S+1},...,t_{S+M-1}\}$ we have a set of corresponding partitions $\{\Phi_S,\Phi_{S+1},...,\Phi_{S+M-1}\}$, such that Φ_k is the partition at the time t_k. We denote the temporal window between two consecutive times t_h and t_k, with $h < k$, as $TW(t_h,t_k)$; it is the period of the query-workload sensoring process (see Fig. 1), that is the period during which input queries are collected and stored. Queries are used at the time t_k for updating the **TP**-*Tree*. So, we have: $\Phi_k = TW(t_h,t_k)$.

4.1 Deriving the Theoretical Bound

The **TP**-*Tree* is the codification of the current partition Φ_k and its configuration evolves as Φ_k evolves. In this sense, the **TP**-*Tree* is a "self-adjusting" data structure, meaning that it is periodically updated by an ad-hoc trigger-like procedure reconfiguring its bucket set according to the query-workload against the target data cube. The period of the updating procedure is defined by the width of the temporal window $TW(t_h,t_k)$ and the query-workload taken into account is formed by the set of queries executed against the target data cube during the time interval $[t_h,t_k]$.

In order to provide probabilistic guarantees on the approximate, we exploit the Hoeffding inequality [8]. In more detail, we apply the Hoeffding bound over each bucket b of the current partition Φ_k, and this allows us to obtain a probabilistic bound over the value of the aggregation operator AVG on b. From the value of the aggregation operator AVG, we easily derive the value of SUM one by multiplying the first by the size of the sampled data set, thus answering any given range-sum query Q on b. By iterating this procedure for each bucket b belonging to Φ_k that intersects Q, we provide the approximate answer to Q, called $A(Q)$, along with probabilistic guarantees on it. We recall that the outlier set does not add any error to the approximate query answering procedure (by setting an appropriate α value).

The Hoeffding inequality asserts the following. Let $Z = \{Z_0,Z_1,...,Z_{M-1}\}$ be a set of independent random variables and let C be a scalar such that $0 \leq Z_k \leq C$, such that k belongs to the set $\{0,1,...,M\text{-}1\}$; furthermore, let $\overline{Z} = \frac{1}{M}\sum_{k=0}^{M-1} Z_k$ be the sample mean of the set Z and let μ be the mean value of the set Z (note that μ is the unknown value that corresponds to the aggregation operator AVG). Then, for each $\varepsilon > 0$ we have:

$$P\left(\left|\overline{Z}-\mu\right| \leq \varepsilon\right) \geq 1-2e^{\frac{-2M\varepsilon^2}{C^2}} \qquad (2)$$

So, we can obtain a probabilistic bound for the event $\mu = \overline{Z} \pm\varepsilon$, that is having the mean value of Z (i.e., μ), within $\pm\varepsilon$ the sample mean of Z (i.e., $\overline{Z}$). Note that $\overline{Z}$ can be quickly computed from the set Z.

In order to usefully adopt such a property in our model, the independent random variable set Z must be defined. We built it using random sampling over each bucket b of the current partition Φ_k: this ensures that each item of b has the same probability $P_0 = 1/|b|$ to be selected. We introduce the random variable set Z(k) defined on the current partition Φ_k as follows: for each bucket b belonging to Φ_k we define the random variable $Z^0(k,b)$ that returns a data cell of the bucket b by applying a uniform sampling in $[0,|d_k|\text{-}1]$ over each dimension of A, with k belonging to the set $\{0,1,\ldots,N\text{-}1\}$. In more detail, each uniform sampling on $[0,|d_k|\text{-}1]$ provides a sampled coordinate s_k and the set $\{s_0,s_1,\ldots,s_{N-1}\}$ is obtained by iterating this single task for each dimension of A. So, the sampled data cell is $A[s_0,s_1,\ldots,s_{N-1}]$. Furthermore, letting $S(k,b)$ be the sample number for a given bucket b in Φ_k, the random variable set $Z_S(k,b)$ on b is defined as follows:

$$Z_S(k,b)=\left\{Z_0^0(k,b),\ Z_1^0(k,b),\ \ldots,\ Z_{S-1}^0(k,b)\right\} \tag{3}$$

that is, $Z_S(k,b)$ is composed of all the random variables $Z^0(k,b)$ defined on b. Note that the uniform sampling task ensures that the random variables belonging to the set $Z_S(k,b)$ are independent, as required by the Hoeffding inequality. Finally, the random variable set $Z(k)$ on Φ_k is composed of all the random variables $Z_S(k,b)$ defined on the buckets of Φ_k.

4.2 The TP-*Tree* Data Organization

The major difference of our approach from [6] is that [6] proposed computing queries in an online fashion, without storing any data structure, while ours stores samples in an off-line fashion. Sampling offline does not affect performances at query time and its storage requirement is modest. In order to manage samples in an query-friendly manner, we propose an index structure called **TP**-*Tree*. Each node n of the **TP**-*Tree* contains the following data: (*i*) the multi-dimensional region R that represents the corresponding bucket b of the current partition Φ_k; (*ii*) the sampled data set $S(b)$ built on b and, for each sample s belonging to $S(b)$, the set of coordinates of s in the original representation; (*iii*) the "current" number of samples $|S(b)|$; (*iv*) the pointers to the child nodes (if n is not a leaf node). Note that we need to store inside the nodes of the **TP**-*Tree* the original coordinates of the sampled data; in fact, this information allows us to execute a query Q on a node n of the **TP**-*Tree* since at query time we need to know which samples of the node n intersect Q (this can be founded by comparing Q and $n.R$).

5 The Overall Query Model

The approximate answer to a query Q is calculated by summing up the query results against the **TP**-*Tree* and against the set of outliers (Ω_B). We note that the query error is solely due to the error in estimating the aggregate of the non-outliers from their samples. In more detail, let Q be a multi-dimensional query involving the bucket set $\{b_0,b_1,\ldots,b_{T-1}\}$ (i.e., $b_k \cap Q \neq \emptyset$); the approximate answer to Q, called $\tilde{A}(Q)$, is obtained following these steps:

1. Aggregate outliers *– Apply Q to the outliers accessed via the index $QT(\Omega_B)$; this step provides the (exact) portion of answer due to the outliers, called $A_O(Q)$.*

2. Aggregate non-outliers *– Apply Q to the sampled data involved in it; this step provides the (approximate) portion of answer due to the sampled data, called $A_S(Q)$.*

***3. Combine aggregates** – Obtain the approximate answer to Q by combining the portions of answer given by step 1 and step 2 as follows:* $\widetilde{A}(Q) = A_O(Q) + A_S(Q)$.

Note that the error in estimating Q, called $\varepsilon(Q)$, is only due to the error in estimating Q on sampled data; that is, $\varepsilon(Q) = \varepsilon_S(Q)$. In Section 3, we provided the algorithm `evalQueryOnQTNode` for computing $A_O(Q)$. Now, we can provide the algorithm `evalQueryOnTPNode` for computing $A_S(Q)$. This portion of answer is computed by using the Hoeffding inequality (2), thus obtaining probabilistic bounds over the degree of approximation of the answer. The algorithm `evalQueryOnTPNode` allows us to evaluate a given query Q on a node n of the **TP**-*Tree* by using the following relation that provides the (approximate) answer to Q on a bucket b of Φ_k involved in it:

$$A(Q,k,b) = |S(k,b)|\left(\overline{Z}_S(k,b)\right) \tag{4}$$

```
Algorithm evalQueryOnTPNode($Q$,$n$)
  sum = 0;
  Set inter = null;
  if ($n.R \cap Q \neq \emptyset$) then
    if ($n.R \subseteq Q$) then
      sum = sum + hoeff($n.\{S\}$);
    else
      if ($n$.list().size == 0) then
        inter = Util.getInterSamples($n.R$,$Q$);
        sum=sum + hoeff(inter);
      else
        for each node $c$ in $n$.list() do
          sum = sum + evalQueryOnTPNode($Q$,$c$);
  return sum;
```

wherein: (*i*) `n.R` is the region of the node *n*; (*ii*) the procedure `hoeff` takes a sampled data set *I* and returns the (approximate) answer to the range-sum query on *I* (see relation (9)); (*iii*) the procedure `getInterSamples`, belonging to the utility package `Util`, takes two multi-dimensional region R_1 and R_2 and returns the points of R_1 which have a not null intersection with R_2; (*iv*) the procedure `list`, called on a node *n*, returns the list of the children nodes of *n*.

The whole (approximate) answer to a multi-dimensional range query Q is provided by calling the `evalQueryOnQTNode` on the root node of the quad-tree $QT(\Omega_B)$, and the `evalQueryOnTPNode` on the root node of the **TP**-*Tree* respectively.

6 Experimental Results

In the interest of space, we present only a simplified experimental results against synthetic data cubes, as this allows us to study the performance of our algorithms on datasets with various controllable features (such as dimensionality, sparseness, size and distribution). To generate synthetic data cubes we randomly generate a set of points in the multi-dimensional space defined by the target data cube and use these points as the centres of cuboids and fill the cuboids according to a given data distribution. In the remainder of the paper, we denote each synthetic data cube by A = $<d,\zeta,s>$, where d is the number of dimensions, ζ is the data distribution used for filling the cuboids, and s is the sparseness coefficient of the data cube. Three data distributions are used: *Uniform* (denoted by $U(u_{min},u_{max})$, such that each data cell value is uniformly distributed in the range $[u_{min},u_{max}]$), *Gaussian* (denoted by $G(p,\sigma,m)$, such that p is the number of peaks, σ is the standard deviation of the peaks, and m is the percentage number of data cells contained in each Gaussian bell with respect to the overall number of non zero data cells of the data cube), and *Zipf* (denoted by $Z(z_{min},z_{max})$, where the range $[z_{min},z_{max}]$ is used for obtaining the parameter of the distribution z as uniformly distributed in this range.). We built seven synthetic data cubes, as listed in Table 1. We follow the same method to generate query-workload as that given in the previous work [11], because it can capture the user behaviour as well as its dynamics.

Table 1. Synthetic data cubes

Data Cube	d	ζ	s[%]
A_0	6	$U(25,75)$	0.001
A_1	6	$G(100,25,0.01)$	0.001
A_2	6	$Z(0.2,1.2)$	0.001
A_3	10	$Z(0.5,1.5)$	0.0001
A_4	10	$U(15,85)$	0.0001
A_5	10	$G(150,35,0.006)$	0.0001
A_6	10	$Z(0.3,1.0)$	0.0001

Table 2. Query-workloads

Query-Workload	z_{min}	z_{max}	ν[%]
QWL_0	0.2	0.9	1
QWL_1	0.5	1.4	1
QWL_2	0.8	1.7	1
QWL_3	0.5	1.5	ν

Such a method uses *Zipf* distributions (as it models real-life phenomena most closely) and a *selectivity function*, denoted by $V[\nu]$ (see Table 2). In the first experiment (see Fig. 3 (a), (b), and (c)) we tested the performances of the **TP**-*Tree*; in the second experiment (see Fig. 3 (d), (e), and (f)) we tested the scalability of the **TP**-*Tree*, by varying the selectivity of the query-workload $QWL_3(\nu)$ = $<0.5,1.5,V[\nu]>$ (i.e., the value of ν in the $V[\nu]$ function) against the synthetic data cubes: A_4, A_5, and A_6. It can be observed that the new method outperforms previous approaches under most of the experimental settings.

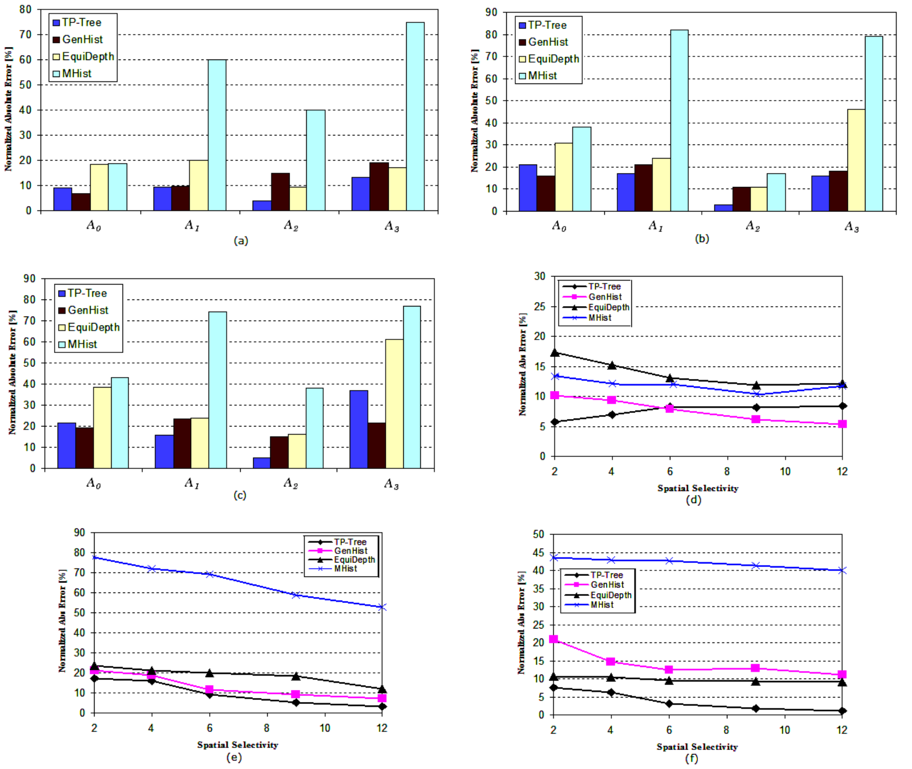


Fig. 3. Performances for the query-workloads QWL_0 (a), QWL_1 (b), and QWL_2 (c) and scalability w.r.t. the spatial selectivity of QWL_3 for the data cubes A_4 (d), A_5 (e), and A_6 (f)

7 Conclusions

In this paper we have presented the **TP**-*Tree*, a tree-like “self-adjusting” synopses data structures for approximate range query answering in OLAP environments. **TP**-*Tree* is based on outlier management and provides theoretical-based probabilistic guarantees over the degree of approximation of the answers. The main contributions of our proposal are: (*i*) low spatial-temporal computational overhead for building and maintaining the **TP**-*Tree*; (*ii*) accuracy control, due to the theoretical framework provided by the Hoeffding inequality. Experimental results clearly confirm the effectiveness of the **TP**-*Tree* against various classes of data cubes, outperforming other well-known similar techniques.

References

1. Barnett, V., and Lewis, T., “Outiliers in Statistical Data”, John Wiley, 3th Edition, 1994.
2. Chatfield, C., “The Analysis of Time Series”, Chapman and Hall, 1984.

3. Chaudhuri, S., Das, G., Datar, M., Motwani, R., and Rastogi, R., “Overcoming Limitations of Sampling for Aggregation Queries”, in *ICDE* 2001.
4. Gaede, V., and Gunther, O., “Multidimensional Access Methods”, in *ACM Computing Surveys*, Vol. 30(1), pp. 170-231, 1998.
5. Gunopulos, D., Kollios, G., Tsotras, V.J., and Domenicani, C., “Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes”, in *SIGMOD* 2000.
6. Hellerstein, J.M., Haas, P.J., and Wang, H.J., “Online Aggregation”, in *SIGMOD* 1997.
7. Ho, C.T., Agrawal, R., Megiddo, N., and Srikant, R., “Range Queries in OLAP Data Cubes”, in *SIGMOD* 1997.
8. Hoeffding, W., “Probability Inequalities for Sums of Bounded Random Variables”, in *Journal of the American Statistical Association*, Vol. 58(301), pp. 13-30, 1963.
9. Jagadish, H., Koudas, N., and Muthukrishnan, S., “Mining Deviantes in Time Series Databases”, in *VLDB* 1999.
10. Knorr, E., Ng, R.T., “Algorithms for Mining Distance-Based Outliers in Large Datasets”, in *VLDB* 1998.
11. Pagel, B.-U., Six, H.-W., Toben, H., and Widmayer, P., “Towards an Analysis of Range Query Performance in Spatial Data Structures”, in *PODS* 1993.
12. Piatetsky-Shapiro, G., and Connell, C., “Accurate Estimation of the Number of Tuples Satisfying a Condition”, in *SIGMOD* 1984.
13. Poosala, V., Ioannidis, Y.E., Haas, P.J., and Shekita, E., “Improved Histograms for Selectivity Estimation of Range Predicates”, in *SIGMOD* 1996.

Computing Complex Iceberg Cubes by Multiway Aggregation and Bounding

LienHua Pauline Chou and Xiuzhen Zhang

School of Computer Science and Information Technology
RMIT University, Melbourne, VIC., Australia, 3000
{lchou,zhang}@cs.rmit.edu.au

Abstract. Iceberg cubing is a valuable technique in data warehouses. The efficiency of iceberg cube computation comes from efficient aggregation and effective pruning for constraints. In advanced applications, iceberg constraints are often non-monotone and complex, for example, “Average cost in the range [δ_1, δ_2] and standard deviation of cost less than β”. The current cubing algorithms either are efficient in aggregation but weak in pruning for such constraints, or can prune for non-monotone constraints but are inefficient in aggregation. The best algorithm of the former, Star-cubing, computes aggregations of cuboids simultaneously but its pruning is specific to only monotone constraints such as “COUNT($*$) $\geq \delta$”. In the latter case, the Divide and Approximate pruning technique can prune for non-monotone constraints but is limited to bottom-up single-group aggregation. We propose a solution that exhibits both efficiency in aggregation and generality and effectiveness in pruning for complex constraints. Our bounding techniques are as general as the Divide and Approximate pruning techniques for complex constraints and yet our multiway aggregation is as efficient as Star-cubing.

1 Introduction

Data Warehousing and OLAP technologies require summarised and subject-oriented views to data for better and faster high level analysis and decision making. To this purpose, data is modelled multi-dimensionally. In a multi-dimensional model, *dimensions* such as Product and Customer-Group describe the subjects of interest; the *measure* such as total sales is the target of analysis in terms of dimensions.

A data cube generalises the SQL Group-By operator [6] to compute Group-Bys of all combinations of dimensions. Each Group-By is a *cuboid* which comprises a set of groups grouped by the same dimensions. For example, (a_1,b_1, c_1,d_1) is one of the groups in the cuboid $ABCD$. Note that upper case letters denote dimensions and cuboids while subscripted lower-case letters denote dimension-values and groups. The cuboid-lattice in Figure 1 for the cube on dimension A, B, C, and D shows the parent and child relationship between cuboids. Since a parent cuboid has more grouping-dimensions than a child cuboid, a group in the parent cuboid is a sub-group of some group in the child cuboid. For example, ($a_1b_1c_1$) is a sub-group of (a_1b_1).

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 108–117, 2004.

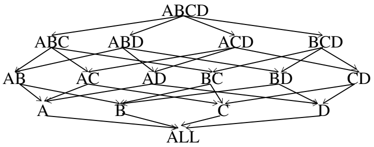


Fig. 1. A cuboid lattice

Given a dataset of k dimensions where each dimension has a cardinality of p, potentially the cube has $(p+1)^k$ groups. Given an aggregate function, cubing is to compute all groups in the data cube for answering queries on aggregations. SUM() COUNT() MIN() MAX(), and AVG() are common SQL aggregate functions. Cubing algorithms [11, 10, 3, 8] exploit the advantage of shared-computation by computing each child-cuboid from a parent cuboid. This aggregation strategy is refered to as *top-down* computation. The Multiway Array Cube [10] is the most efficient among these algorithms.

Iceberg cubes [2] were proposed to compute only interesting groups that meet users information need. Users specify *iceberg constraints* on the aggregate values of groups and only satisfying groups are computed. Iceberg constraints can be either *monotone* or *non-monotone* [2]. A monotone constraint possesses the property that if a group fails the constraint, so do all its sub-groups[1]. The monotone property is commonly used for pruning in iceberg cubing [2]. BUC [2] and H-cubing [7] are current iceberg cubing algorithms which apply the divide-and-conquer approach where groups are computed before their sub-groups. Such an aggregation strategy is called *bottom up* with respect to the cuboid lattice. Pruning for monotone constraints can be easily applied to the bottom up approach.

Star-cubing [4] is the most efficient iceberg cubing algorithm. It makes use of shared computation as in Multiway Array Cubing [10] and incorporates pruning for monotone constraints. It is shown that Star-cubing is more efficient than BUC and H-cubing. However, the optimisation in Star-cubing is specific to only monotone constraints such as "$\mathsf{COUNT}(*) \geq \delta$".

The diversity of users' information needs means that practical constraints are often complex and non-monotone, such as constraints with AVG or SD (StandardDeviation). Iceberg cubing for complex non-monotone constraints is challenging due to the difficulty of pruning. Recently, Wang et al. [9] proposed an algorithm *Divide and Approximate* for pruning for non-monotone constraints. The pruning techniques however are only suitable for BUC like bottom-up single group oriented computation. A more detailed analysis of Divide and Approximate pruning is provided in Section 4.

In this work, we propose the iceberg cubing algorithm *Bound-cubing*, which incorporates bounding techniques for pruning with complex non-monotone constraints. It is not only general and effective in pruning but also well suited for an efficient top-down shared aggregation strategy. Importantly, little overhead is

[1] In the literature, this is also called anti-monotone.

incurred by bounding. Our experiments show that *Bound-cubing* is marginally more efficient than Star-cubing in computing iceberg cubes with monotone constraints; it significantly outperforms the Divide and Approximate algorithm [9] for non-monotone constraints.

2 Bounding Aggregate Functions for Pruning

In this section, we describe our bounding techniques for pruning. We aim at pruning with general constraints of the form "$f(x)$ *op* δ", where $f(x)$ is an aggregate function, *op* is the comparison operator $\geq$ or $\leq$, and δ is a threshold. We will focus on bounding for a single constraint; this can be easily extended to multiple constraints. The main idea of bounding is to bound the aggregation values of a set of groups from the same values used for computing the aggregations of the groups. The derived bounds can be used to prune the set of groups at once.

In cube computation, aggregate functions are categorised into *Distributive*, *Algebraic*, and *Holistic* functions, based on how a group can be aggregated from its sub-groups [6].

- **Distributive**: An aggregate function F is distributive if for a group g, $F(g)$ can be aggregated from the values of $F(g_s)$, where g_s groups are g's sub-groups. Aggregate functions SUM, COUNT, MIN, and MAX are distributive.
- **Algebraic**: An aggregate function F is algebraic if for a group g, $F(g)$ can be computed from a known M number of intermediate aggregations of g's sub-groups by some aggregate functions. For clarity, we call these aggregate functions in F *local aggregate functions.* Aggregate functions AVG, SD, MaxN, MinN, and CenterOfMass are algebraic. Taking AVG as an example, AVG = "SUM(SUM(m))/SUM(COUNT($*$))", where m is the measure. Note that distributive functions are a special case of algebraic functions, where $M = 1$ with a single local aggregate funciton being F.
- **Holistic**: There is no fixed number of intermediate aggregate values that can aggregate a group for a holistic function. RANK is a holistic function. There exists no known computation methods other than computing the aggregation from the raw data hence they are not considered in this work.

Given a cube defined on k dimensions, the *data cube core* [6] refers to the k-dimensional groups. In the context of a cube, the data cube core are the base units of aggregation which can be further aggregated to compute other aggregations in the whole cube [6]. For example, given CUBE(A, B, C) with SUM, the groups (a_1, b_i, c_j) for some i can be aggregated to compute SUM((a_1, b_i)), all of which can in turn be aggregated to compute SUM((a_1)). We extend this definition to define the *core* of a set of groups with super and sub-group relationship, which is the basis for bounding.

Definition 1. *(The core groups and the group-Core) Given a group g and the set of its sub-groups $\mathcal{S}_g$, the* group-Core *of g and $\mathcal{S}_g$, denoted as $\mathcal{C}_g$, are the set of* *sub-groups in $\mathcal{S}_g$ that have the maximum number of grouping-dimensions. Each group in $\mathcal{C}_g$ is a* core group *denoted as g_c.*

All later discussions are implicitly within the context of g and $\mathcal{S}_g$. Also, F^U and F^L denote the upper and lower bound of a distributive or algebraic function F for g and $\mathcal{S}_g$ respectively. The symbols +ve and -ve denote positive and negative.

Like the data cube core, g_c groups are the base units of aggregations, a set of g_c groups can be aggregated to compute g or a group in $\mathcal{S}_g$. Therefore, the largest and smallest aggregations of F that can possibly be produced by any subset of $\mathcal{C}_g$ are the common bounds of F for g and $\mathcal{S}_g$.

While the set of base units can be any set of higher dimensional sub-groups of g_c groups for g and every group in $\mathcal{S}_g$, the g_c groups are the most aggregated base units. Most stringent pruning by bounding is achieved with g_c groups.

It is clear that exhaustively checking all subsets of $\mathcal{C}_g$ is infeasable. Furthermore, there are multiple local aggregate functions in a algebraic function F . We formally define the boundability of an aggregate function as follows.

Definition 2. *An aggregate function F is boundable if (1) the bounds for F are made up from the boundary-values of every local aggregate function f; and (2) enumerating the subsets of $\mathcal{C}_g$ is unnecessary in deriving the boundary-values of every f. The* boundary-values *of f include the max +ve, min +ve, max -ve, and min -ve values of f. The appropriate boundary-value of each f for bounding F is based on the context of the arithmetic operations f is in.*

2.1 Bounding Distributive Functions

All distributive functions are boundable since (1) the boundary-values of the single f prodce the bounds of F and (2) they can be derived with a single scan of $\mathcal{C}_g$. The bounds in terms of the set of g_c groups of all distributive functions for +ve and -ve measures are listed in Table 1.

Table 1. The upper and lower bounds of all distributive functions

F	F^U	F^L
SUM	if $\exists$ SUM(g_c) > 0, SUM(SUM(g_c)), where SUM(g_c) > 0; otherwise, MAX(SUM(g_c))	if $\exists$ SUM(g_c) < 0, SUM(SUM(g_c)), where SUM(g_c) < 0; otherwise, MIN(SUM(g_c))
COUNT	SUM(COUNT(g_c))	MIN(COUNT(g_c))
MAX	MAX(MAX(g_c))	MIN(MAX(g_c))
MIN	MAX(MIN(g_c))	MIN(MIN(g_c))

2.2 Bounding Algebraic Functions

Bounding covers algebraic functions that apply only the basic arithmetic operations (+, $-$, $\times$, and /) on their local aggregate functions.

The boundability of algebraic functions is illustrated by the following two examples. Assume the measure can be +ve or -ve and let s and c be the local aggregate values *sum* and *count* at g_c groups.

Example 1. (A boundable algebraic function) The algebraic function $\mathsf{AVG} = \mathsf{SUM}(s)/\mathsf{SUM}(c)$. The max +ve of $\mathsf{SUM}(s)$ and the min +ve of $\mathsf{SUM}(c)$ derive AVG^U. Also both the max +ve $\mathsf{SUM}(s)$ and the min +ve $\mathsf{SUM}(c)$ are boundable, as is shown in Table 1. AVG is boundable.

Example 2. (An unboundable algebraic function) The algebraic function $(1/\mathsf{AVG}) = \mathsf{SUM}(c)/\mathsf{SUM}(s)$. The max +ve of $\mathsf{SUM}(c)$ and the min +ve of $\mathsf{SUM}(s)$ derive $(1/\mathsf{AVG})^U$. Unfortunately, the min +ve of $\mathsf{SUM}(s)$ comes from some combinations of g_c groups which cannot be determined by a single scan of $\mathcal{C}_g$. $(1/\mathsf{AVG})$ is unboundable.

2.3 Optimisation for the Expression "$\mathsf{SUM}(a)$ /$\mathsf{SUM}(+b)$"

We optimise the common expression "$\mathsf{SUM}(a)$ / $\mathsf{SUM}(+b)$" which exists in many algebraic functions, including AVG, CenterOfMass, and VAR (SD). For example, $\mathsf{VAR} = \mathsf{SUM}(s^2)/\mathsf{SUM}(c) - 2 \times (\mathsf{SUM}(s)/\mathsf{SUM}(c))^2 + \mathsf{SUM}(s)/\mathsf{SUM}(c)$ where s^2, s, and c are the square of sum, sum and count values at g_c groups and $\mathsf{SUM}(c)$ is always positive. The function contains three expressions of the form $\mathsf{SUM}(a)/\mathsf{SUM}(+b)$.

Theorem 1. *(Bounding the expression $E = \mathsf{SUM}(a)/\mathsf{SUM}(+b)$) Given E, let a_i and b_i be two intermediate aggregate values a and b in the i^{th} g_c group and b_i is always positive. Then $E^U = \mathsf{MAX}(a_i/b_i)$ and $E^L = \mathsf{MIN}(a_i/b_i)$.*

Due to space constraints, the formal proof is omitted. The optimised E^U and E^L are more stringent than the general techniques presented in Section 2.2. We use AVG to liiustrate that the optimisation achieves more strigent bounding. AVG^U is $\mathsf{SUM}(s)^U/\mathsf{SUM}(c)^L$ with the techniques in Section 2.2, and it is $\mathsf{MAX}(a/b)$ with the optimisation. Since $\mathsf{SUM}(a)^U \geq a_i$ and $\mathsf{SUM}(b)^L \leq b_i$ for any a_i and b_i, $\mathsf{SUM}(a)^U/\mathsf{SUM}(b)^L \geq \mathsf{MAX}(a/b)$.

In the next section, we present an efficient aggregation strategy which can easily identify $\mathcal{C}_g$ for incorporating bounding during computation.

3 Top-Down Multiway Aggregation

We adopt a strategy where multiple cuboids are aggregated simultaneously. The *Group-Bounding Tree* (*GB-tree*) is built to directly represent multiple cuboids. The simultaneous computation of multiple remaining cuboids in a data cube involves the construction of sub-GB-trees. Bounding with $\mathcal{C}_g$ are incorporated for pruning branches while building a sub-GB-tree.

3.1 The Group-Bounding Tree

A Group-Bounding Tree (*GB-tree*) of an aggregation F for a k-dimensional dataset consists of a root node and k levels of branch nodes, each of which corresponds to a dimension. A node contains a dimension-value, one or more intermediate aggregate values for F, and the links to its child nodes. An example GB-tree on dimensions A, B, C, D, and E for AVG is presented in Figure 2.

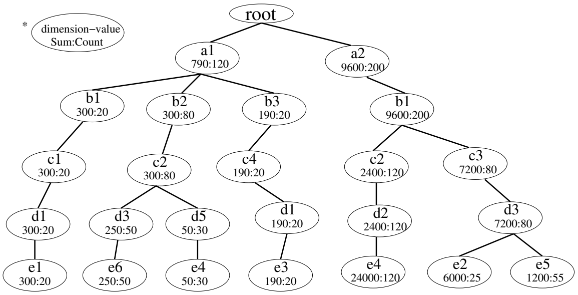


Fig. 2. Group-Bounding Tree

Groups and Sub-groups in the GB-Tree. In a GB-tree, a group g is represented by the nodes of g's last grouping dimension-values on the branches containing all of g's grouping dimension-values. A sub-group of g is represented on a subset of g's branches that contain the additional grouping-dimension-values of the sub-group. In the example, `(a1,b1)` is a sub-group of `(a1)` but not of `(b1)`. Two observations are made:

Observation 1 *On a GB-Tree, the branch from the root to a node is a unique combination of dimension-values. Each node represents a unique group and a descendent node its sub-group.*

Observation 2 *The leaf nodes under a group g are g's sub-groups having the largest number of grouping-dimensions. They are the g_c groups of g and $\mathcal{S}_g$.*

3.2 The Multiway Aggregation Strategy

Building a GB-tree simultaneously computes many groups. In the example, groups in cuboids ALL, A, AB, ABC, $ABCD$, $ABCDE$ are computed. For the remaining cuboids, each non-leaf dimension of the GB-tree is dropped to construct a sub-GB-tree; and each non-leaf dimension that is below the last dropped dimension of a sub-GB-tree is subsequently dropped to further construct sub-GB-trees. When dropping a dimension D on a GB-tree, the branches below each d_i $(d_i \in D)$ are amalgamated to form the new set of descendent levels of the parent of the dropped dimension.

Each sub-GB-tree computes the set of child cuboids at the levels below the last dropped dimension; they share the prefix dimensions before the last dropped dimensions. Let a group grouped by the prefix dimensions be a *prefix group*; groups subsequently aggregated are all its sub-groups.

In our example, the sub-trees of the original tree are $BCDE$ $(-A)$, $ACDE$ $(-B)$, $ABDE$ $(-C)$, and $ABCE$ $(-D)$ where the last dropped dimension is indicated in (). The tree $ACDE$, whose last dropped dimension is B and the prefix dimension is A, computes the cuboids $ACDE$, ACD, and AC; they are all sub-groups of some a_i. The cuboids ADE and AD computed by further dropping C of $ACDE$ sub-tree are also sub-groups of some a_i.

```
Global Input: GB-tree $T$ of a $k$-dimensional dataset, $c$ =constraint.
                //$T$.prefixCuboid= $\emptyset$ and $T$.last-droppedD = 0
Output: Bound-cubing($T$)
Algorithm Bound-cubing($T$) {
    foreach ($g \in T$)// $g$ is a group on a node of $T$
        if(constraintCheckingOK($g$)) output $g$;
    subTrees = $T$.dropDimensions();
    foreach ($sT \in$ subTrees) Bound-cubing($sT$);
}
Procedure dropDimensions() {
    subTrees = $\emptyset$;
    foreach ($i \in$ {last-droppedD +1, ..., $k$} )
        create sub-tree $sT_i$;
        foreach ($g_p \in$ prefixCuboid)// foreach prefix group
            if(boundCheckingOK($g_p$))
                $sT_i$.amalgamate($g_p$, $D_i$);//amalgamate branches of $g_p$ below $D_i$
        $sT_i$.last-droppedD= $i$;
        $sT_i$.prefixCuboid= prefixCuboid $\cup$ {$\forall$ $D_j$ | $j \in$ {1 ... $i$−1}}
        subTrees = subTrees $\cup$ $sT_i$;
    return subTrees;
}
```

Fig. 3. Bound-Cubing Algorithm

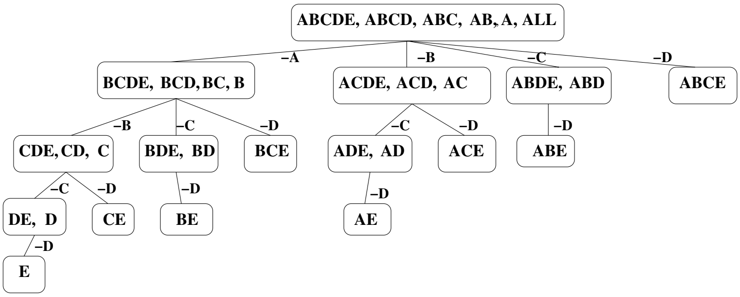


Fig. 4. The Aggregation Strategy for all cuboids

The recursive construction of sub-trees by dimension-dropping computes all cuboids. The Bound-cubing Algorithm is shown in Figure 3.

The computation strategy diagram for CUBE(A, B, C, D, E) is shown in Figure 4. The cuboids in each rectangle are computed simultaneously from a single GB-tree.

Stringent Bounding for Pruning. A sub-GB-tree computes sub-groups of some prefix group g_p. The leaf nodes of g_p are the g_c groups of g_p and its sub-groups. Pruning using g_c groups of g_p can be directly applied when building a sub-GB-tree. The branches under failed g_p are trimmed so that they do not participate in generating sub-trees. To ensure effective pruning, when constructing the next level sub-trees, the prefix groups are updated and the new g_c groups are used for pruning. To illustrate, the prefix groups on $ACDE$ tree is A; by dropping D, ACE tree is formed which computes the cuboid ACE. The prefix groups become AC and the ACE groups at the leaf level become the new g_c groups.

4 Related Work

We discuss the two most related work, the Divide-and-Approximate [9] and Star-cubing [4] iceberg cubing algorithms.

Divide and Approximate-Cubing (D&A Pruning). While bounding can handle some algebraic functions with local aggregations of both signs in × and / operations, they cannot be handled by D&A [9]. The constraint " SUM/MAX" cannot be approximated by D&A pruning when SUM and MAX can be both positive and negative values, but they can be bounded. Moreover, in D&A pruning, the approximator can only be derived from some group g with a sub-group of g, g_s, that is grouped by all dimensions and can prune only the groups that are sub-groups of g and super-groups of g_s. This suggests that in the simultaneous computation of multiple cuboids, it is difficult to locate the specific groups that can be pruned by the derived approximator on the tree before the groups are actually computed.

Star-Cubing. Both the Star-tree and the GB-tree are inspired by the H-tree [7]. Star-cubing uses star-nodes which represent all non-frequent dimension-values to reduce the size of the Star-tree. It is shown in [4] that for monotone constraints, star nodes significantly improve the efficiency of iceberg cubing. It is briefly mentioned that star nodes can also be incorporated for non-monotone constraints such as those involving AVG. However, this may involve considerable additional cost. Star nodes can be easily incorporated in Bound-cubing for monotone constraints.

Bound-cubing and Star-cubing share the spirit of multiway computation of cuboids [10]. In Star-cubing, with a traversal of the Star-tree, only the group at the root and the groups of the cuboid at the leaf level are computed. Bound-cubing computes multiple cuboids at multiple levels of the GB-tree at once while multiple Star-trees need to be built in Star-cubing.

5 Experiments

Two datasets are used for our experiments. One is real world data consisting of 88443 tuples taken from the 1990 US census [12]. The dataset has 61 attributes. We extracted the 10 attributes with discrete domains as dimensions such as group-quarters-type, marital-status, and occupation, with cardinalities between

6 to 10. The measure is total-income. The other is the OLAP benchmark relational tables *TPC-R* [1]. We constructed a single joined relation to derive a multidimensional dataset consisting of 1 million tuples with 10 dimensions, with cardinalities between 2 and 25.

We examine two aspects of Bound-cubing separately. First, the aggregation strategy of Bound-cubing is compared with Star-cubing. Second, the bounding techniques are compared with D&A for pruning effectiveness. For fair comparison, we have implemented all three algorithms and applied the same implementation optimisations whenever applicable. Our experiments were conducted on a PC with an Intel Pentium R 4, 1.70 GHz CPU and 512M main memory, running Red Hat Linux7.2. All programs are written in C++. All data structures required can fit in memory. The runtime measured excludes the I/O time.

5.1 Bound-Cubing vs. Star-Cubing

To evaluate the aggregation cost of Bound-cubing, we compare it with Star-cubing using the monotone constraint "$\mathsf{COUNT}(*) \geq \delta$". The runtime of both algorithms on both datasets are recorded at different count thresholds and shown in Figure 5. Bound-cubing is consistently faster than Star-cubing on both datasets at all count thresholds. Bound-cubing is slightly more efficient than Star-cubing. This can be attributed to the more simultaneous aggregation in Bound-cubing.

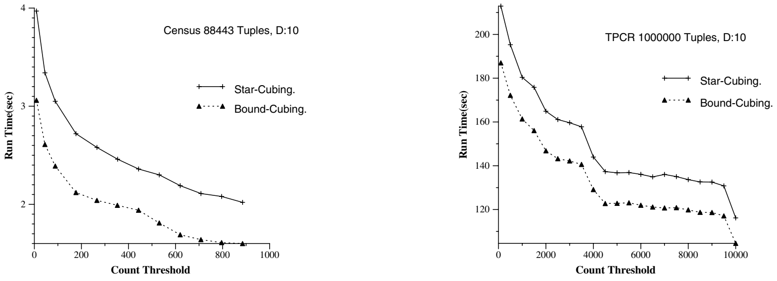


Fig. 5. Bound-cubing vs Star-cubing for Count Threshold

5.2 Bound-Cubing vs. D&A Cubing

We compare Bound-cubing with D&A cubing on the non-monotone constraint "$\mathsf{AVG}(m) \geq \alpha$". Their performance is shown in Figure 6 Two observations are made: (1) Bound-cubing is always significantly more efficient than D&A cubing at all thresholds. The improvement is 2 to 15 times on census dataset and 7 times on TPCR dataset. The performance gain is attributed to the simultaneous aggregation and bounding for pruning. (2) While the threshold of the constraint increases, Bound-cubing becomes faster. Surprisingly, the runtime of D&A does not decrease with larger thresholds. This suggests that the overhead of pruning increases with the increase in the constraint threshold. In contrast,

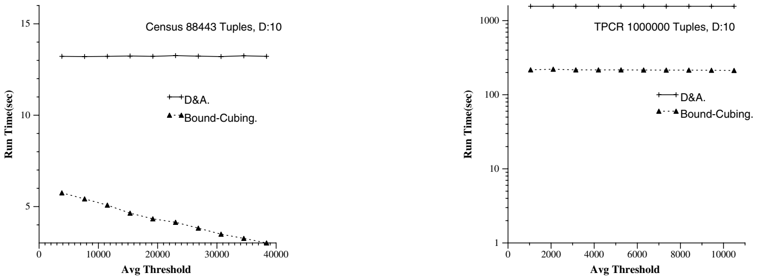


Fig. 6. Bound-cubing vs D&A for AVG Threshold

bounding techniques have low overhead and the efficiency gains always outweigh the overhead incurred in pruning.

6 Conclusions and Future Work

We have developed effective techniques that bound the aggregate values of groups. The bounding techniques are general and can prune for complex non-monotone constraints defined with distributive and algebraic functions. We have also developed an efficient computation strategy on the Group-Bound tree that computes multiple cuboids simultaneously. In terms of cube computation, our contribution is a general approach for dealing with complex non-monotone constraints. The approach incurs little overhead and fits nicely with the multiway aggregation strategy.

Acknowledgements

We thank Justin Zobel for his helpful comments.

References

1. The TPC-R benchmark, http://www.tpc.org/tpcr/.
2. Beyer and Ramakrishnan. Bottom-up computation of sparse and Iceberg CUBE. SIGMOD'99.
3. Agarwal et al. On the computation of multidimensional aggregates. VLDB'96.
4. Dong et al. Star-cubing: Computing iceberg cubes by top-down and bottom-up integration. VLDB'03.
5. Fang et al. Computing iceberg queries efficiently. VLDB'98.
6. Gray et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. J. Data Mining and Knowledge Discovery, 1997.
7. Han et al. Efficient computation of iceberg cubes with complex measures. SIGMOD'01.
8. Sarawagi et al. On computing the data cube. Tech. report, IBM Almaden Research Center, 1996.
9. Wang et al. Pushing aggregate constraints by divide-and-approximate. ICDE'02.
10. Zhao et al. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD'97.
11. Ross and Srivastava. Fast computation of sparse datacubes. VLDB'97.
12. Historical Census Projects. ftp://ftp.ipums.org/ipums/data/ip19001.Z, 1997.

An Aggregate-Aware Retargeting Algorithm for Multiple Fact Data Warehouses

Karin Beckera and Duncan Dubugras Ruiz

Pontifical Catholic University of Rio Grande do Sul, Faculty of Informatics
Porto Alegre - Brazil
{kbecker,duncan}@inf.pucrs.br

Abstract. The use of pre-computed aggregate data is a common technique to address performance in Data Warehouse systems (DW), but the adoption of aggregates must be transparent. An aggregate retargeting mechanism redirects a query to available aggregate table(s) whenever possible. In this work, we present an aggregate retargeting mechanism that deals with multiple fact table schemas (MFTS), which are composed of many fact tables related by conformed dimensions. The algorithm provides two types of transparency: a) aggregate unawareness, and b) MFTS join complexity unawareness. The algorithm requires simple metadata, and it is non-intrusive. The paper presents the retargeting algorithm, required metadata and performance experiments.

1 Introduction

Data warehouses (DW) are analytical databases aimed at supporting decision-making. A DW is modeled in terms of fact and dimension tables, which can be arranged in a variety of schema types, such as star, snowflake or multiple fact tables [7][12][6]. A *multiple fact tables schema (MFTS)*, or *fact constellation schema*, is composed of several fact tables, related through a set of conformed dimensions [7], i.e. dimensions that have the same meaning at every possible fact table. MFTSs are commonly used to model a DW as a set of multiple, interrelated subjects [6][12]. In relational databases, On-line Analytical Processing (OLAP) operations typically result in SQL queries in which aggregation functions (e.g. SUM) are applied to fact table attributes, using dimension table attributes as grouping columns. MFTSs impose restrictions on the joining of facts and dimensions: distinct fact tables can only be joined if they represent data at the same granularity level.

Performance is a critical issue in DW systems. Pre-computed aggregation is the most efficient strategy to reduce the time needed to access a large numbers of rows [4][9][15]. An aggregate, also referred to as *summary table*, is a materialized view that stores redundant, consolidated information. The adoption of aggregates presupposes the resolution of a number related problems, among them [9]: a) finding the best set of aggregates under space or update overhead constraints; b) efficiently computing these aggregate views from raw data; c) efficiently maintaining the aggregates when new data is shipped to the data warehouse; and d) exploring the aggregates to answer queries. This paper addresses the last problem.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 118–128, 2004.

The adoption of aggregates should be completely transparent to DW users for two main reasons [7]: a) users typically do not have the necessary technical skills to make the appropriate choices on how to conveniently use aggregates to improve performance, and b) as decision requirements change constantly, aggregates may have to change accordingly (e.g. removal of obsolete aggregates). An aggregate retargeting mechanism provides this transparency, by redirecting a query, if possible, to the available aggregate table(s) that can answer the query [9].

This paper presents an aggregate retargeting algorithm that handles MFTS, providing users with two types of transparency: aggregate unawareness and MFTS complexity unawareness. The algorithm requires simple metadata, and it is non-intrusive. The paper discusses the retargeting algorithm, underlying metadata, and presents performance tests. This paper improves previous work [2] by considering aggregates that represent a subset of the available data (i.e. by restricting the values of dimension attributes) and by presenting performance experiments.

The remainder of this paper is structured as follows. Section 2 discusses a motivation scenario. Section 3 presents the algorithm and the required metadata. Section 4 describes the experiments and Section 5 discusses the results. Section 6 presents related work. Conclusions and future work are addressed in Section 7.

2 Joining Tables in MFT Schemas

To illustrate the problem tackled, let us consider the example depicted in Figure 1.(a). The Costs table is considered along the dimensions Product(Code, Class, Group, Family, Line, Division) and Time(Month, Quarter, Year), whereas the Sales table is considered along Product, Time and Channel(Base).

Sales

Prod_ID	Time_ID	Channel_ID	DollarSold
D1P1	Jan/2001	C1	500
D1P1	Jan/2001	C2	500
D1P1	Feb/2001	C1	1000
D1P1	Feb/2001	C2	1000
D2P1	Feb/2001	C1	5000

Costs

Prod_ID	Time_ID	DollarCost
D1P1	Jan/2001	100
D1P2	Feb/2001	200
D1P1	Mar/2001	300
D2P1	Jan/2001	400
D2P1	Feb/2001	500

(a) An MFTS extension.

Quarterly Analysis

Division	Quarter	DollarSold	DollarCost
D1	1Q/2001	3000	600
D2	1Q/2001	5000	900

(b) Quarterly Sales/Costs Analysis.

Fig. 1. Example of Facts and resulting Analysis.

Suppose that a critical query is to compare quarterly sales and costs of product divisions. Assuming that product D1P1 and D1P2 belong to division D1, whereas product D2P1 belongs to division D2, the corresponding analysis is presented in Figure 1.(b). The user, unaware of the subtleties involved, might formulate the query as in Figure 2, that only produces correct results when Costs(Prod_ID, Time_ID) and Sales(Prod_ID, Time_ID) have both identical set of instances. Since this condition is not met by the tables depicted in Figure 1.(a), this query produces incorrect results. First, an outer join is necessary, because not all combinations of month/product present in Costs are included in Sales. Second, and most importantly, due to the Channel dimension, all sales corresponding to the same month/product must be aggregated before they can be joined to the corresponding costs fact for that same month/product, otherwise the same Costs record will be joined with many Sales records, one for each Channel value. Most users do not have the technical skills for realizing these problems, and their implications in query formulation. Therefore, queries involving MFTS need to be hidden through specific interfaces and applications.

"the sales amount and respective costs, per quarter and product division"

```
Select Division, Quarter, SUM(DollarSold) DollarSold, SUM(DollarCost) DollarCost
From Time, Product, Sales S, Costs C
Where Month= S.Time_ID and Month= C.Time_ID and Code= S.Prod_ID and Code=
C.Prod_ID
Group by Division, Quarter
```

Fig. 2. Naive input SQL query.

3 The Retargeting Algorithm

The retargeting algorithm handles MFTS with the use of aggregates, providing total transparency for users. Transparency in this context has a twofold meaning: a) *aggregate unawareness*, and b) MFTS *join complexity unawareness*. This retargeting service is non-intrusive, and it is intended to lie between the front-end tool and the DBMS. It considers as input a query written by a user through a proper user interface (e.g. a graphical one intended for naive users). The algorithm assumes that:

- Users are unaware of MFTS joining complexities, and always write their queries in terms of desired facts and dimensions in a single query. The retargeting service is responsible for rewriting the query to produce correct results despite of the DW extension, assuming as a premise that it is necessary to bring each fact table to the same summarization level before joining them (as explained in Section 2).
- Users are unaware of the existence of aggregates, formulating queries in terms of the original base tables and dimensions. To improve performance, the retargeting service is responsible for rewriting the query in terms of aggregates, if possible.
- Retargeting queries involving a single fact table (Star Schema) is a special case of MFTS. Other types of schemas (e.g. Snowflake) are not considered.

3.1 Metadata

The algorithm requires very few metadata, which is easily supplied by the DBA. Metadata, depicted in Figure 3 using a UML diagram, is divided into three main groups: tables, star schemas and relationship between star schemas. A *MFTSchema* is modeled as a set of individual *StarSchema*s, which can be related by derivation relationships (*derived/deriving*). Aggregating the fact table of its deriving schema(s) produces a derived schema, also referred to as an aggregate schema, or simply aggregate. Notice that an aggregate schema can have more than one deriving schema. Thus, derivation relationships define a partial ordering among schemas [9], meaning that any derived schema can be computed from the Cartesian product of its deriving schemas. Only direct derivations are maintained. For the algorithm, the derivation relationship is represented by a directed acyclic graph G(V, E), where V represents a set of star schemas, and E corresponds to the set of derivation relationships between any two schemas [13]. Figure 4 exemplifies a derivation graph, where Sales and Costs are the base schemas. With respect to its deriving schema(s), an aggregate schema can eliminate dimensions or be related to shrunken dimensions. A shrunken dimension [7] has either fewer attributes with regard to the dimension it shrinks (e.g. dimension TimeSh(Year) shrinks Time(Month, Quarter,Year)), fewer tuples (e.g. TimeSh is a subset of Time where Year > 1999) or both. We assume that the names of the attributes are not modified in shrunken dimensions/aggregates with regard to the original ones (i.e. from the respective deriving schemas).

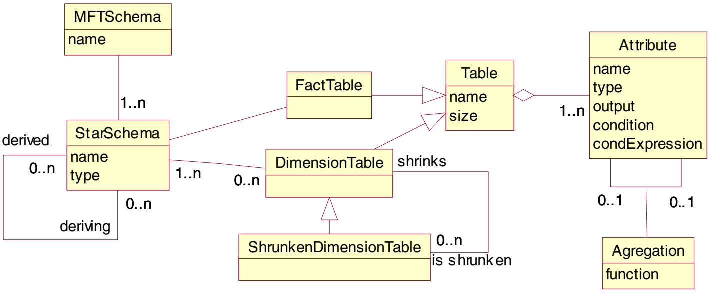


Fig. 3. UML representation of Metadata.

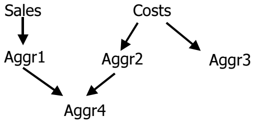


Fig. 4. Derivation graph for the MFTS.

3.2 The Algorithm

The input is a user query with the following restrictions: a) no nested queries; b) distributive aggregation functions [9]; c) all dimensions listed in the *from clause* apply to all fact tables; d) the *where* and *having* clause are a conjunction of primitive Boolean conditions. The algorithm is divided into 4 steps, presented below. The output of each step is illustrated in Figure 5, using the input query of Figure 2 and the derivation graph of Figure 4. The algorithm is an extension of the one presented in [7], to deal with multiple fact tables.

Step 1: Division into Component Queries

```
For each fact table $F_i$ listed in the from clause of input query $Q$, create a component
    query $C_i$ such that:
    $C_i$ from clause := $F_i$ and all dimensions listed in the from clause of Q;
    $C_i$ where clause := all join conditions of $Q$ necessary to relate $F_i$ to the dimensions,
        together with any conditions involving these dimensions;
    $C_i$ group by clause := all attributes used in the group by clause of $Q$;
    $C_i$ having clause := all conditions involving aggregation functions applied to $F_i$ attributes;
    $C_i$ select clause := all attributes used in the group by clause of $Q$, in addition to all
        aggregation function(s) applied to $F_i$ attributes;
```

Step 2: Selection of Candidates for Component Queries

This step generates the **candidate set CS_i** for each component query, which are schemas (base or aggregate) that answer C_i. For each component query, the graph is traversed recursively starting from the corresponding base schema, i.e. the one containing the fact table F_i listed in the ***from clause*** of C_i. Notice that when the algorithm detects that a schema cannot answer the component query, the search stops because no schema derived from it would be able to answer it as well. In the worst case, C_i contains the respective base schema.

```
For each component query $C_i$, generated in Step 01:
    Let n:= the node that corresponds to the base schema of $C_i$;
    Let $CS_i$ := Explore($C_i$, n);
```

The recursive function **Explore** is defined as follows:

```
Explore($C_i$, n)
    ResultSet := {};
    If compatible_attributes($C_i$ , n) and compatible_conditions($C_i$ , n) and compatible_aggregate_functions($C_i$ , n)
    Then begin
            Let ResultSet:= {n};
            For each dn, where dn is a derived schema of n
                Let ResultSet:= ResultSet ∪ Explore($C_i$, dn);
        end;
    Return ResultSet.
```

where *compatible_attributes(q, n)* is a function that verifies if all attributes of the component query *q* (clauses ***select***, ***where***, ***having***) are described for the schema *n*; *compatible_conditions(q, n)* verifies if the set of conditions specified in the ***where*** clause of *q* is subsumed by the conjunction of conditions described over the attributes of the schema *n* [1]; and *compatible_aggregate_functions(q, n)* verifies whether the aggregation functions used in the ***select*** and ***having*** clauses of *q* are compatible with the ones described over the numeric attributes of the schema *n* [9].

Step 1: Component Query
C1 = *Select* Division, Quarter, SUM (S.DollarSold) DollarSold
From Time, Product, Sales S
Where Month = S.Time_ID and *and* Code = S.Prod_ID
Group by Division, Quarter

C2 = Select Division, Quarter, SUM (C.DollarCost) DollarCost
From Time, Product, Costs C
Where Month = C.Time_ID and Code = C.Prod_ID
Group by Division, Quarter

Step 2: Candidate Set
$\mathbf{CS_1}$ = {Sales, Aggr1, Aggr4} for **component query $\mathbf{C_1}$**;
$\mathbf{CS_2}$ = {Costs, Aggr2, Aggr4} for **component query $\mathbf{C_2}$**.

Step 3: Best Candidate
t-min = (Aggr4, Aggr4)

Step 4: Query Reformulation
Select Division, Quarter, SUM (DollarSold) DollarSold, SUM (DollarCost) DollarCost
From Sh_Time, Sh_Product, Aggr4 A4
Where Quarter = A4.Time_ID and Line = A4.Prod_ID
Group by Division, Quarter

Fig. 5. Output of Algorithm Steps.

Step 3: Selection of Best Candidates

This step selects the best combination of candidates to rewrite the query, and this choice is based on the ***minimum accumulated size*** criterion, defined as the minimum volume of records that need to be manipulated to answer the query.

Let **T** be a set of tuples ($\mathbf{e_1}$, …, $\mathbf{e_n}$), where $\mathbf{e_1} \in \mathbf{CS_1}$, …, $\mathbf{e_n} \in \mathbf{CS_n}$, (n>0), representing the Cartesian product of candidate sets $\mathbf{CS_1}$ X .. X $\mathbf{CS_n}$. Let **t** be a tuple of **T**. The accumulated size of **t**($\mathbf{e_1}$, …, $\mathbf{e_n}$), **AS(t)**, is a function that returns the sum of records that must be handled if the query were rewritten using **t**. **AS(t)** computes only once the size of a given table, in case it is included in more than one candidate set $\mathbf{CS_i}$.

Consider all $\mathbf{CS_i}$ sets generated in Step 2,

Generate **T** = $\mathbf{CS_1}$ X .. X $\mathbf{CS_n}$

Let **t-min** := **min(** **AS(t)** for all **t** $\in$ **T**);

Step 4: Query Reformulation

Once the best candidate for each component query is determined, the query is rewritten. If the set of best candidate resulting from Step 3 has a single element, i.e. a common aggregate for all component queries, a single query is written using that aggregate and the respective shrunken dimensions. Otherwise, the query is rewritten in terms of views that summarize the best aggregates individually, which are subsequently joined. Figure 5 (Step 4) displays the rewritten query for our example, where Sh_Time and Sh_Product are shrunken dimensions related to aggregate Aggr4. This algorithm is trivial and it is not presented due to space limitations.

4 Experiments

The purpose of our experiments was to verify if the algorithm leads to best choice (in terms of query execution time) to answer a multi-fact query, considering different configurations of aggregates. The set of aggregates includes both aggregates that summarize individual fact tables, and aggregates that summarize and join different fact tables (referred to as joined aggregates). We performed tests considering: (a) different sizes for the output, (b) different levels of aggregation (compression factor) and (c) four different manners of implementing the data warehouse (base and aggregate schemas). The four alternatives are a combination of sorting and indexing properties, namely: (i) unsorted and pre-sorted tables, considering the best and the worst possible sorting order, (ii) tables with and without proper indexes. Best sorting and proper indexing are considered with regard to the target input query. A table presents the best sorting to solve a target query if its ordering matches the sequence of retrieving the records by the DBMS, saving I/O operations. To be realistic, we consider that the tables of base schemas always have proper indexes.

We adopted the MFTS schema of the APB-1 OLAP Benchmark [11], from which we selected tables Inventory and Sales, considered along 3 common dimensions: Product(Code, Class, Group, Family, Line, Division), Customer(Store, Retailer) and Time(Month, Quarter, Year). Sales is additionally considered along the dimension Channel(Base). Figure 8 depicts the hierarchy of aggregates defined, where Inventory and Sales are the base tables, I_i are aggregates summarizing facts of Inventory, aggregates S_i summarize facts of Sales, and IS_i are joined aggregates that summarize and join Sales and Inventory facts. Our target input queries are the ones used to generate IS_4, IS_3, IS_2 and IS_1. For each of them, we executed a set of tests considering the presence of different aggregates. Thus, we tested how the query was answered using the joined aggregate of its respective level, as well as using combinations of aggregate/base schemas from the previous levels. For instance, *Query One* retrieves the content of aggregate $IS_{4.}$ Therefore, *Query One* was answered by: (a) aggregate IS_4 , (b) summarizing joined aggregates from the IS_4 derivation relationship (IS_3, IS_2, IS_1), and (c) summarizing and joining schemas in the derivation hierarchy of I_4 and S_4, including base schemas Inventory and Sales. This rationale applies to the experiments related to *Query Two* (IS_3), *Query Three* (IS_2) and *Query Four* (IS_1).

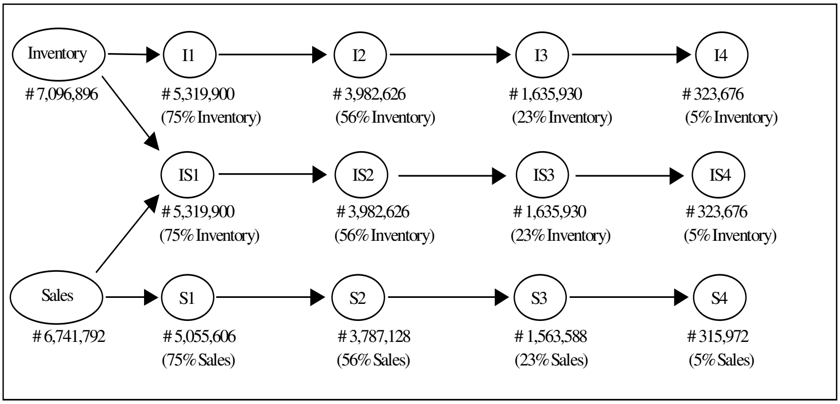


Fig. 8. Derivation Graph.

Experiment data was generated using the APB.EXE program, executed in a Sun Ultra II - Solaris, using a market leader database management system. APB.EXE generates randomly data for the fact/dimension tables, given two input parameters: *channel* (used to populate dimension tables) and *density* (percentage that determines how many of the possible dimensions combinations will have corresponding facts). They were settled as 17 and 0.1%, respectively, resulting in fact tables with a very large population. Aggregate hierarchy levels are defined to obtain aggregates that represent approximately 75, 50, 25 and 5 percent of the base fact tables population (levels 1, 2, 3 and 4, respectively), chosen as representative values for verifying execution behavior considering different summarization rates. These are obtained by eliminating the Channel dimension of fact table Sales, and by recursively shrinking the Product dimension. Figure 8 displays the fact table population (#) for each base or aggregate schema, as well as the corresponding percent value w.r.t. the base fact table.

Table 1. Execution times to answer *Query One.*

Source Tables	Accumulated Size	Indexed / Sorted	Without Indexes/Sorted	Indexed/Unsorted	Without Indexes/Unsorted
Inventory/Sales	13838688	3:17:51		4:06:51	
I1 × S1	10375506	0:46:42	1:39:37	0:48:30	1:52:48
I2 × S2	7769754	0:40:45	1:13:48	0:35:32	1:28:15
IS1	5319900	0:26:44	0:28:06	0:19:47	0:21:05
IS2	3982626	0:24:01	0:23:13	0:17:17	0:17:41
I3 × S3	3199518	0:19:14	0:31:50	0:18:21	0:39:45
IS3	1635930	0:12:32	0:12:48	0:08:17	0:09:46
I4 × S4	639648	0:05:27	0:09:04	0:06:36	0:08:59
IS4	323676	0:04:52	0:04:46	0:05:32	0:05:09

Table 1 details data related to the 34 experiments executed for *Query One*. Column *Source Tables* presents the tables used to solve the target query. Column *Accumulated Size* sums population of used tables. Each cell represents an execution of the target query in terms of the *Source Tables*, considering one of the 4 specific DW implementations considered. For example, the execution time of *Query One* on I1 × S1 is 0:46:42, considering that I1 and S1 are properly indexed and present the best sorting. Previous tests were done on the DBMS to determine the best and worst sorting, for each table, given a target query. Recall that base schemas are always indexed.

Figure 9 presents Table 1 data graphically. The x-axis presents the Accumulated Size of the Source Tables and the y-axis displays execution times. Each curve corresponds the execution time according to a DW implementation. We produced similar data for the 3 other queries (*Two*, *Three* and *Four*), so as to test performance considering different output population. Although absolute times changed, the patterns displayed in Figure 9 are maintained in all 3 sets of experiments.

5 Experiments Discussion

Considering execution times displayed in Table 1 and plotted in Figure 9, it is possible to observe that the criterion accumulated size always adopted make the best choice in the following situations:

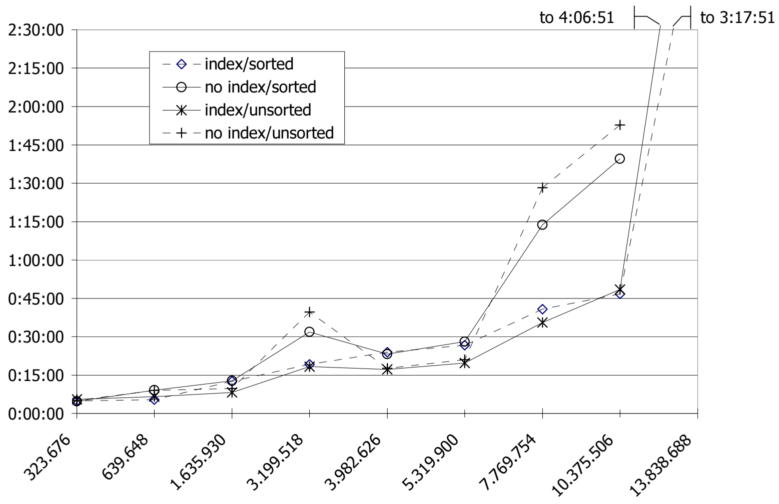


Fig. 9. Query One Performance Graph

- all candidates are indexed by proper indexes;
- except for the base schemas (which are always candidate), none of the candidates are properly indexed, but they are all joined aggregates;

The accumulated size may not lead to the best decision when it is necessary to join fact tables, or when some candidates have indexes whereas others (possibly smaller ones) do not. Indeed, the existence of indexes may be more significant than accumulated size alone, particularly if fact tables must be joined. For instance, the time necessary for executing the query based on I3 × S3 without proper indexes (Figure 9, x = 3,199,518) is superior than the time required to compute it from IS2 (Figure 9, x = 3,982,626), which is larger, when IS2 has proper indexes. Notice that it is even comparable to I2 × S2 candidate, which is significantly larger (Figure 9, x = 7,769,754), if both I2 and S2 are indexed. It should be observed however that the difference in time is not always significant and this situation does not constitute a pattern (e.g. see I4 × S4 vs. IS3). Properly sorted tables normally produce better results, but table sorting does not have a significant impact on time execution.

The results with regard to indexing are not surprising, for index role is well known [9]. In spite of that, the inadequate choice of our algorithm did not imply a severe performance degradation by considering exclusively the accumulated size. In our experiments, the worst case represented approximately 40 minutes, which corresponds to 89% of time increase (i.e. it has not even doubled the execution time). Also, the impact of indexes is directly related to table population. The main advantage of the accumulated size criterion is that it is possible to develop a non-intrusive algorithm, which does not require a detailed understanding of DBMS characteristics for proper parameterization and functioning.

6 Related Work

The problem of exploiting aggregates for efficient query processing is addressed in works such as [3][5][8][14][15]. The focus of [3][5] is on the optimization of the query plan, with the use of new operators. [8] addresses cache management, which is suitable for supporting the interactive, ad hoc analyses typical of DW environments. [15] presents an algorithm that rewrites queries to make the best use of aggregates. The algorithm detects the overlapping part between an input query and one or more aggregates, providing a compensation for the non-overlapping parts. With regard to [5][14][3], the main contribution of [15] is the complexity of input queries addressed (i.e. complex expressions, multidimensional aggregation and nested queries). None of these works address MFTS. Also, they are meant to be embedded into the database engine or OLAP server, and compute aggregates dynamically.

The algorithm presented here is an extension of one proposed in [7], in order to allow users to deal with multiple fact tables in their input queries. The algorithm of [7] is used to implement a query navigator, focused on Star Schemas. Notice that this navigator allows a user to deal with MFTS if he/she produces two or more aggregate tables, each one based on a single Start Schema, and then joins the results using SQL. This approach does not provide the transparency with regard to the problems related to MFTS, since all problems highlighted in Section 2 may happen. MS Analysis [10] also allows joining virtual cubes, thus handling MFTS. However, virtual cubes can be joined only if they share at least one common dimension, provided that both cubes have along these common dimensions identical set of members. Thus, only a limited set of MFTS extensions are supported. Unlike these works, the algorithm proposed in this paper provides total transparency on the problems related to input queries submitted to MFTS, imposes no constraints on DW extension and additionally improves performance by exploiting aggregates. To the best of our knowledge, other market leaders such as DB2 and Oracle do not handle MFTS.

7 Conclusions

The algorithm presented in this work deals with MFTS, providing two types of transparency: a) aggregate unawareness, and b) MFTS query complexity unawareness. The algorithm requires simple metadata, and it is non-intrusive. Thus it can be implemented as a layer in between the user front-end tool and DBMS engine.

To test the algorithm's performance, the experiments considered four alternative DW implementation scenarios that could influence the results, different input and output populations, and various types of aggregates (joined aggregates vs. aggregates of a single fact table, different compression factors). The population of the base fact tables is also representative of real DW environments. We conclude that the criterion of accumulated size produces good results in almost all the cases. The significant exceptions were related to the existence of indexes, particularly in situations when distinct fact tables must be joined. In spite of that, the inadequate choice of the algorithm did not imply a severe performance degradation.

The main advantage of the accumulated size criterion is that it is not necessary to take into account DBMS internals characteristics. For instance, one has to study the optimization plans and statistical data provided by a specific DBMS in order to understand the heuristics and function costs applied, and how these criteria influence each other. This can result in a difficult algorithm parameterization, which would be specific to DBMS, platform configuration and volume of records at hand. This parameterization is error-prone, and could lead to questionable results if not adequately performed. In opposition, the metadata required by our algorithm can be easily collected with simple SQL standard statements.

Future work includes extending performance tests (e.g. more than two fact tables, indexing related to Boolean conditions that restrict aggregates), improving the algorithm (e.g. other cost metrics, union of distributed schemas), the integration of the current algorithm implementation into a DW architecture, the use of the proposed algorithm in the context of dynamic aggregate computation, among other topics.

References

1. ABITEBOUL, S., DUSCHKA, O.M. Complexity of Answering Queries Using Materialized Views. In: ACM PODS'98, 1998. Proceedings… pp. 254-263
2. BECKER, K.; DUNCAN, D.; SANTOS; S. Aggregate Awareness in Multiple Fact Table Schema Data Warehouses. In: ADBIS 2002. Proceedings… pp. 41-50.
3. CHAUDHURI, S., SHIM, K. An overview of cost-based optimization of queries with aggregates. Bulletin of the technical committee of data engineering v. 18, n. 3, Sept. 1995.
4. CHAUDHURI, S., DAYAL, U. An Overview of Data Warehousing and OLAP Technology. SIGMOD Record, v.26, n.1, 1997. pp. 65-74
5. GUPTA, A., HARINARAYAN, V., QUASS, D. Aggregate-query Processing in Data Warehousing Environments. In: VLDB'95, 1995. Proceedings… pp. 358-369.
6. HAN, J.; KAMBER, M. Data mining. Morgan Kaufmann, 2001.
7. KIMBALL, R. et al. The Data Warehouse Lifecycle Toolkit : expert methods for designing, developing, and deploying data warehouses. John Wiley & Sons, 1998.
8. KOTIDIS, Y., ROUSSOPOULOS, N. A Case for Dynamic View Management. ACM Transactions on Database Systems. v. 26, n. 4, 2001. pp. 388-423
9. KOTIDIS, Y. Aggregate view management in data warehouses. In: Handbook of massive data sets. Kluwer Academic Publishers, 2002. pp. 711 – 741
10. Microsoft Corp. OLE DB Programmer's reference: Join Compatibility. *Online.* Available at: http://msdn.microsoft.com/library/en-us/oledb/htm/olappr_chapter25_21.asp.
11. OLAP Council. APB-1 OLAP Benchmark (Release II). *Online.* Available at : http://www.olapcouncil.org/research/bmarkco.htm.
12. POE, V., KLAUER, P., BROBST, S. Building a Data Warehouse for Decision Support 2nd edition. Prentice Hall, 1998.
13. SOUZA, M., SAMPAIO, M. Efficient Materialization and Use of Views in Data warehouse. SIGMOD Record, v. 28 n. 1, 1999. pp. 78-83
14. SRIVASTAVA, D.; DAR, S. et al. Answering queries with aggregation using views. In: VLDB'96, 1996. Proceedings… pp. 318-329
15. ZAHARIOUDAKIS, et al. Answering complex SQL queries using automatic summary tables. In: SIGMOD´00, 2000. Proceedings… pp. 105-1161

A Partial Pre-aggregation Scheme for HOLAP Engines

Wo-Shun Luk and Chao Li

School of Computing Science, Simon Fraser University, Canada
woshun@sfu.ca, clij@cs.sfu.ca

Abstract. This paper describes a scheme for partial pre-aggregation to speed up the response time of queries that are posed for the array-like interface, subject to the constraint that all pre-computed aggregates must fit into storage of a pre-determined size. The target query workload consists of all base and aggregate cells that are stored in a multidimensional array (i.e. cube). These queries are actually range queries pre-defined by users. Due to the huge size of all possible aggregate cells, the emphasis of our scheme is to reduce the overhead for query compilation. An efficient and effective query decomposition method is devised, which works well with a pre-aggregation scheme whereby pre-computed aggregates form a sub-cube of the full cube. A greedy algorithm is devised is to derive such a sub-cube. A HOLAP engine which implements this partial pre-aggregation scheme is described. Experimental results using both synthetic and real-life datasets are presented to demonstrate that the partial pre-aggregation scheme is viable, and for some complex queries, accelerates query execution by close to 300 times.

1 Introduction

In the recent years, the database industry has settled on HOLAP (Hybrid) OLAP as the main approach for implementation of OLAP (On-Line Analytic Processing) applications. A HOLAP is a combination of a MOLAP engine over aggregated data (the cube) and a Relational DBMS over base data. It is more scalable than a pure Multidimensional OLAP (MOLAP), but it keeps the array-like interface from its MOLAP engine, which offers superior performance for many OLAP applications in comparison to SQL, or its extended versions. According to a group of Oracle researchers, SQL suffers, in a fundamental way, from (i) lack of language constructs to treat relations as arrays and to define formulas over them, and (ii) lack of efficient random access methods for array access ([Wit03]).

This paper is concerned with construction of an HOLAP engine that directly supports an array-style querying interface. Due to a well-documented phenomenon called *database explosion* [Pendse03], only a small portion of the set of all possible aggregated data is generated and stored in the cube. This practice is called partial pre-aggregation. To the best of our knowledge, this paper is the first to study partial pre-aggregation for MOLAP/HOLAP systems[1]. Indeed, there have been assertions that that every aggregate in MOLAP database must be pre-computed (e.g. [PS99], [H98]).

[1] Some MOLAP/HOLAP vendors may have implemented their partial pre-aggregation schemes, but we are unaware of any description of these schemes in the literature.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 129–137, 2004.

Several research articles have been published ([BPT97], [HCKL00], [HRU96], [GHRU97], [SDN98], [GM99]), which studied the partial pre-aggregation problem for ROLAP systems. However, the research issues about partial aggregation for HOLAP engines are very different because of its array-like interface. Unlike views as defined by the CUBE-BY operator ([GBPL96]), cells in the cube are far more numerous. We define *query compilation* as the process to search for relevant pre-aggregated data which would assist in computation of the answer for a query. The query compilation time is generally negligible in the case of views, because they can be connected as a lattice ([HRU96]). In case of HOLAP engines, the query compilation time could be so large that it can offset any advantage in reduction in access time. Much of this paper is about finding an efficient query compilation scheme, an emphasis that actually drives the optimization process.

The rest of this paper is organized as follows. The basic concepts of HOLAP, together with the special terms we use in this paper, are explained in Section 2. The data structure (i.e. B-tree) to organize the non-empty cells of the cube is also discussed there. In Section 3, the query compilation method, which is essentially a query decomposition method, is described. In Section 4, we describe the Greedy Algorithm, which selects cells whose measures will be pre-computed. In Section 5, experimental results are presented to show the overhead of the query processing, and to justify this overhead in view of reduction in query response time. Section 6 is the conclusion.

2 Cube and HOLAP Engine

The (full) *cube* is defined in this paper to be a k-dimensional array, where k is a positive integer greater than zero. Each dimension of a cube has D_i members, 1<=i<=k, which are organized as a *hierarchy*. The members at the leaf level are called *primary* members. All other members in a higher level of the dimension hierarchy are called *group* members. The hierarchy is a *tree* hierarchy, where a member is assumed to have exactly one parent, except for the root. A *sub-cube* is a k-dimensional array, which has all primary members and a subset of group members for each dimension of the full cube. The *base cube*, as a sub-cube of the cube, is the k-dimensional array, each dimension of which has only the primary members. This is the smallest sub-cube.

Fig. 1 shows dimension hierarchies of a 2-dimensional OLAP database, i.e., all sub-cubes built from this database are 2-dimensional cubes. All members, which are numbered within the interval [1,4] are primary members, and the remaining ones are group members. The cube contains all 7 members on each dimension. The base cube contains all 4 primary members, but no group members, on each dimension.

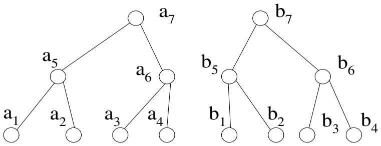


Fig. 1. Dimensional Hierarchies of an OLAP Database

A cell in the cube is labeled by a k-tuple, which is composed of its coordinates along the k dimensions. A cell is an *aggregate* cell if at least one coordinate of the cell is a group member of some dimension, otherwise it is a *primary* cell. A cell stores a single numeric value, which is called the *measure*, although the results of this paper are equally valid for multiple values stored in each cell. Measures of all primary cells are input from a data source. The measure of an aggregate cell may be calculated according to the method to be discussed later in this section. The calculation can take place on demand, usually when its value is required for computation of the answer of a query, or, it can be pre-computed, i.e., before the query time. In the latter case, the cell is called a *pre-aggregated* cell. A cube is *fully pre-aggregated* if all aggregate cells in the full cube are pre-aggregated. A PPA (*Partially Pre-aggregated*) cube is a sub-cube, where all aggregate cells in the sub-cube are pre-aggregated. A group member that is included in some dimension of a PPA cube is called a *pre-aggregated* member of this sub-cube. A PPA cube is uniquely defined by the set of all pre-aggregated members.

The input to the HOLAP engine is a query, which points to a cell of the (full) cube. The purpose of our HOLAP engine is to determine the measure of the cell, and then return it as the answer of the query. In this paper, we assume that all aggregated data form a PPA cube. Consequently, the query whose corresponding cell is not found in the PPA cube, must be computed on the fly. Hopefully, cells in the PPA cube would help to reduce the response time of this query. The goal of this research is to find a PPA cube, whose storage size is not to exceed a certain pre-determined threshold, such that the response time of an average query is minimized. In Section 4, we will introduce a greedy algorithm which essentially selects group members from all dimensions that may form the PPA cube. For the time being, some PPA cube is assumed to exist, in order that we may proceed to the discussion of the HOLAP engine.

The HOLAP engine has two major components: the query processor and an access method that is based on a B-tree. The query processor is to decompose a query into a number of queries, or sub-queries, whose answers are readily retrievable from the PPA cube. The contents of the cells in the PPA cube are organized into a disk-based B-tree. Query decomposition is covered in Section 3. For the rest of this section, we are concerned with the B-tree, or more broadly, the data storage and retrieval issues.

We adopt an addressing scheme such that all possible cells in the full cube are visualized as a linear array. Let D_i be the total number of primary and group members for dimension i. The dimensions are so named such that $D_i <= D_{i+1}$, for $0<= i <= k-1$. Thus there are all together $D_0*\ldots*D_{k-1}$ cells in the cube. The address of a cell is the *offset* from the top of the linear array, and is called the *key*. For simplicity, all members of each dimension, say dimension i, are mapped, on 1-to-1 basis, into a range of integers, $\{0, 1, \ldots, D_i-1\}$. Given the coordinates of a cell, we can now turn them into integers, say, $(v_0, \ldots, v_{k-1})$. The key of this k-dimensional cell is then equal to $v_0*D_1*\ldots*D_{k-1} + v_1*D_2*\ldots*D_{k-1} + \ldots + v_{k-1}$. Conversely, given the key, the coordinates also may be computed. With this addressing scheme, a cube is actually a set of records, in the form of <key, measure>, where measure is non-zero. Empty cells are not stored. A B-tree is built, with these records stored in the leaf nodes. To retrieve the measure in a cell, its coordinates are first converted into a key, and then locate its measure from the B-tree.

3 Query Decomposition

A *query* is submitted to the HOLAP engine as a tuple T (t_1, …, t_k) where t_i, 1<=i<=k, is the coordinate of the cell in the ith dimension. The terms of query and cell are used interchangeably in this paper. A *descendant* of a tuple T(t_1, …, t_k) is another tuple T'(t_1', …, t_k') such that each t_i' is a descendant of t_i in the ith dimension hierarchy. T' is an *immediate child* of T, if their components are identical except for one dimension, say i, such that t_i' is an immediate child (i.e. the closest descendant) of t_i. There could be as many immediate children of T as there are dimensions. In particular, T' is *the* immediate child of T *along the dimension i.*

We define the *measure* of an aggregate tuple T to be a non-holistic aggregate function ([GBLP96]) of all measures of tuples which are collectively represented by the aggregate tuple. This measure is recursively defined as follows. If the immediate children of T are primary members, the measure of T is a function of measures of all these immediate children. If the immediate children of T are group members, the measure of T is a function of measures of all its immediate children of T along any one single dimension.

The following result, which follows from the above definition, and is proved in [Luk01], offers an alternative way to compute the measure of a tuple.

Lemma 1: The measure of T is an aggregate function of measures of all primary members that are also descendants of T.

The *primary tuple set* (*pts*) of T is defined to be the set of all primary tuples that are descendant of T. To compute the answer to T, one needs to access all cells that are associated with the tuples in pts(T), retrieve their measures and then compute the aggregation function. There is a straightforward method to locate this set of tuples. Let us define a *primary member set* (or pms, in short) of a group member G in a dimension hierarchy as the set of primary members that are descendants of G. For the tuple T above, we can easily show that the set of tuples, which is the Cartesian product of the primary member sets of each component of T, i.e. pms(t_1) × pms(t_2) ×…× pms(t_k), is a superset of pts(T). It is a superset because many of the tuples included in the Cartesian product simply do not exist. Nonetheless, assuming no prior knowledge of the cube, these non-existent tuples must be generated, and attempts must be made to access their cells.

We define the *query cost* for T as the number of tuples that must be accessed for the computation of its answer. In the case of zero pre-aggregation, the cost of process a query T (t_1, …, t_k) is $|pms(t_1)|*\ldots*|pms(t_k)|$. With pre-aggregated cells in the PPA, our query processing strategy is to decompose a given query Q into a set of queries, whose answers are stored in the PPA cube. The query cost of Q is the cardinality of this set. For the rest of this section, we will describe a process which will give the lowest query cost for any Q.

Consider an arbitrary dimension hierarchy, say DH. Let *m* be an arbitrary member in the hierarchy. Let S be the set of all members in the PPA cube belonging to the dimension. A *cover* for *m* is defined to be a set of members in DH, e.g. {p_1, …, p_i}

such that (i) every member in the set is a member of S, (ii) every member is a descendant of *m* in DH, and (iii) pms(*m*) = pms(p_1) ∪ … ∪ pms(p_i).

Note that every member *m* in DH has at least one cover, i.e. the set of primary members in DH, which are also descendants of *m*. A *mincover* of *m*, given S, mincover(*m*, S), is defined to be a cover with the smallest cardinality. The following algorithm computes the mincover C, which is initially null. We assume that *m* is not in S, because otherwise *m* itself is the mincover.

Algorithm 1 (FindMincover)

```
FOR each primary member of $m$, say $r$,
    1. Locate all members on the path between $m$ and $r$ in the dimension hierarchy
    2. Let $p$ be the closet member on the path to m that is pre-aggregated (which
       could be $r$ itself)
    3. IF $p$ is not already in C THEN include $p$ into C
END
```

Lemma 2: The set C derived from Algorithm 1 is a cover of m and has the smallest cardinality among all covers of *m*, i.e. C = mincover(*m*, S).
Proof: see [Li03].

Consider now a query Q (q_1, …, q_k). Let S_i, $1 \leq i \leq k$, be the set of members on the ith dimension hierarchy of the PPA cube. This query will be decomposed into a set of pre-aggregated queries, according to the mincover of each coordinate. In particular, this is an optimal decomposition, as the following Lemma shows.

Lemma 3: The Cartesian product: mincover(q_1, S_1) × mincover(q_2, S_2) ×…× mincover(q_k, S_k) is the smallest set of sub-queries that must be processed in order to derive the answer for Q.
Proof: see [Li03].

To conclude this Section, we have shown that if we are given a PPA cube, we can produce a query decomposition plan that will guarantee the lowest query cost. The processing overhead in produce this plan is minimal, if mincover(q_i, S_i) for every i, $1 \leq i \leq k$, have been pre-computed before query time, although I/O may be involved in retrieving the pre-computed mincovers.

4 Construction of PPA Cube

We construct a PPA cube as follows. We begin with a base cube, and select members from all k dimension hierarchies for inclusion into the PPA Cube, one at a time, until the estimated size of PPA Cube reaches a pre-determined threshold. Then the PPA Cube is computed, using any of the many published algorithms ([Luk01], [BR99]). For the rest of this Section, we describe the member selection process using the 'greedy' approach.

The Greedy Algorithm looks for a member that would give maximum benefit per unit of storage, among all remaining yet-to-be-selected members. Our job here is to compute the benefit that the selection of a certain member would bring, and the amount of storage the pre-aggregation would take up. The *benefit* due to inclusion of a new member in the PPA Cube is the reduction of the total query cost for all queries. This may seem to be a lot of work involved to calculate the benefit at each step of the Greedy Algorithm. Fortunately, it is not as difficult as it seems.

We denote Total-QC(S_1, …, S_k) as the total query cost for the PPA cube where S_i, $1 \leq i \leq k$, is the set of members for the ith dimension of the cube. Let M(m_1, …, m_k) be an arbitrary tuple in the full cube. By Algorithm 1, we compute mincover(m_i, S_i), $1 \leq i \leq k$, the mincover of m_i in the ith dimension hierarchy with S_i as the set of pre-aggregated members. Then,

(1) $$\text{Total-QC}(S_1, \ldots, S_k) = \Sigma \ldots \Sigma|\text{mincover}(m_1, S_1)|*\ldots*|\text{mincover}(m_k, S_k)|$$
$$= \Sigma(|\text{mincover}(m_1, S_1)|*\ldots* \Sigma|\text{mincover}(m_k, S_k)|)$$
$$= \text{Dim-QC}(S_1) * \ldots * \text{Dim-QC}(S_k)$$

where each summation ranges over all members in a dimension hierarchy, and Dim-QC(S_i), $1 \leq i \leq k$, as the sum total of all mincover cardinalities in the dimension i, is the ith component of the total query cost.

Suppose now a new member p is to be added to S_i, so that $S_i' = p \cup S_i$. The benefit, which is dependent on p, and (S_1, …, S_k), can be computed as follows: To compute Total-QC(S_1, …S_i', …, S_k), we need only compute Dim-QC(S_i').

(2) $$\text{Benefit}(p, S_1, \ldots, S_k)$$
$$= \text{Total-QC}(S_1, \ldots, S_i, \ldots, S_k) - \text{Total-QC}(S_1, \ldots, S_i', \ldots, S_k)$$
$$= (\text{Dim-QC}(S_i) - \text{Dim-QC}(S_i'))*(\text{Dim-QC}(S_1)* \ldots*\text{Dim-QC}(S_{i-1})*\text{Dim-QC}(S_{i+1})*\ldots*\text{Dim-QC}(S_k))$$

By (2), we now know that the benefit of choosing a member p for inclusion into the PPA cube can be calculated by re-computing the mincovers of the members in the same dimension hierarchy. Specifically, we need only re-compute those mincovers which may be affected if p is made a pre-aggregate member.

In general, we may compute the benefit per unit storage if a member p is made an aggregated member, as follows. We assume, without loss of generality, that p is member of ith dimension hierarchy, where S_i is the current set of all aggregated members. Let $S_i' = p \cup S_i$. Then, from (2),

(3) $$\text{Benefit}(p, S_1, \ldots, S_k)$$
$$= (n*(|\text{mincover}(p, S_i)| - 1))*$$
$$(\text{Dim-QC}(S_1)* \ldots*\text{Dim-QC}(S_{i-1})*\text{Dim-QC}(S_{i+1})*\ldots*\text{Dim-QC}(S_k))$$

where *n* is the number of ancestors of p in the ith dimension hierarchy between p and the closest ancestor of p that is already a pre-aggregated member. Next, we will compute the extra storage that may be incurred by the inclusion of p to the PPA cube. This may be estimated using the method described in [SDNR96].

We will now describe the Greedy Algorithm. Initially, the base cube as the input database, i.e. each side of the PPA Cube, S_i contains only primary members of the dimension.

Algorithm 2 (Greedy Algorithm)

```
Repeat
    1. Find $m_i$, $1<=i<=k$, in ith dimension hierarchy, $m_i$ not being in $S_i$, such that Benefit($m_i$, $S_1$, …, $S_k$) per unit storage is the largest (see benefit calculation in Equation (3));
    2. Choose $m_i$ such that Benefit($m_i$, $S_1$, …, $S_k$) per unit storage is the largest
    3. $S_i = m_i \cup S_i$;
    4. Update the storage size of PPA Cube;
    5. Update the mincovers of all ancestors of $m_i$ in the ith hierarchy;
Until the size of the PPA cube is larger than a pre-set limit.
```

5 Experimental Results

Our experiments are performed on an Intel Pentium IV machine running Windows 2000. The machine has a physical memory of 1.5G bytes.

We use a synthetic and a real-life dataset for experiments. We will discuss first the experimental results with the synthetic dataset. It has 5 dimensions with identical 6-level dimension hierarchy. Beginning from the top of the hierarchy, the 6 levels of a hierarchy have in total 1, 2, 5, 25, 50, and 100 members at each respective level. Thus, each dimension has 83 group members and 100 primary members. (The same dimension hierarchy is used for experiments in [SDN98].) The base cube has 100,000 non-empty cells, which is expanded into the full cube, by including about 365,000,000 non-empty aggregate cells.

The Pre-Aggregation Ratio is the estimated size of the PPA cube relative to estimated size of the full cube. We implement the Greedy Algorithm, and the Random Algorithm for construction of PPA cubes, given a pre-set value of the Pre-Aggregation Ratio. The Random Algorithm, which is to select members randomly, is implemented for comparison purposes.

In the query processing phase, sets of 10,000 queries are posed to the HOLAP engine. The queries are cells randomly selected from the full cube, our query workload. The response time for answering a query is the total of two time components: (i) the query decomposition time, and (ii) the time to retrieve from the PPA cube the aggregated values for all sub-queries. The first component includes time to locate the relevant pre-computed mincovers, and compute the final query result. The second component is mostly the total response time for all sub-queries, which is roughly proportional to the query cost, i.e., the number of sub-queries resulted from query composition. The query response time for each query is so short that we opt for measuring the query response time of the whole query set, instead of individual queries. All 10,000 queries in a query set will have the same query complexity, i.e. a predetermined number of group members in a query (out of 5).

Our experiments are designed to examine the feasibility of the scheme proposed in this paper. We will first show the efficiency of the query decomposition strategy. As

Table 1 shows, the *relative* query decomposition time is significant (12.3% - 34.3%), but not high. This is due to mostly the I/O cost in fetching the mincovers from disk, since every experiment has a "cold" start – initially, nothing relevant in the system buffer. As pre-aggregation ratio increases, this relative query decomposition time increases, indicating that the response time for each sub-query decreases. The opposite happens in case of query complexity. As the query complexity increases, the relative query decomposition time decreases, indicating that the response time for each sub-query increases.

Table 1. The Query Decomposition Time as a Percentage of Query Response Time (Synthetic Dataset)

Query Complexity (# of group members in a query)	Pre-Aggregation Ratio 10%	Pre-Aggregation Ratio 15%
1	27.1%	34.3%
3	23.6%	29.4%
5	12.3%	20.2%

The next set of experiments is designed to answer the following question quantitatively: How well does the whole idea of Partial Pre-aggregation work? The data in Table 2 show that it clearly works well. If no pre-aggregation is performed, or equivalently, the answers for the query set are computed directly from the base cube, the system could be close to 300 times slower than if a good partial pre-aggregation scheme is adopted. Table 2 also sheds some light on the final question: how good is the Greedy algorithm? It shows that Greedy consistently beats Random, and the performance gap widens as the query complexity increases.

Table 2. Query Response Time in (ms) by Algorithms – Pre-Aggregation Ratio 10% (Synthetic Dataset)

Query Complexity	No Pre-Aggregation	Greedy	Random
1	112	44	61
3	1708	109	188
5	65166	223	1039

We repeat roughly the same set of experiments with a real-life dataset, on SFU enrolment. The 5-dimensional dataset, with 2 measures, has about 20,000 records. Despite the significant differences between the datasets, the results are very similar. A detailed account of the experiments with this dataset and associated results can be found in [Li03].

6 Conclusion

After much debate on MOLAP vs. ROLAP, the database industry has settled on the hybrid of the two, HOLAP. Despite its name, it actually features two engines: a HOLAP engine to directly support an array-style query interface, e.g. a spreadsheet, and an expanded SQL that facilitates OLAP applications running on the RDBMS. This paper addresses the question of how to support efficiently array-style query interface with a partial pre-aggregation scheme. From this preliminary study, we may conclude that partial pre-aggregation for array-style query interfaces is feasible, and could significantly accelerate query response time. This study also highlights the importance of an efficient query processing strategy, which will generate a good query compilation plan. We demonstrate that an efficient and effective query processing strategy is indeed possible if we decide in advance how the pre-aggregated cells may be organized. We show in this paper that one effective organization of pre-aggregated cells is to pack them as a sub-cube, i.e. a PPA (partially pre-aggregated) cube.

We note that the query workload consists of range queries, which are defined by the users as they define dimension hierarchies. Ad-hoc range queries can be decomposed into a number of these range queries. Generally speaking, if the number of these decomposed queries is very large, the response time may be too long. But we are encouraged by what we saw in our experiments the tremendous speedup in query execution even for large query sets. One of the future research directions is to study the extent to which the HOLAP engine, as presented here, would support the ad-hoc range queries vis-a-vis the ROLAP engine.

References

[BPT97] E. Baralis, S. Parabolschi, E. Teniente, “Materialized Views Selection in a Multidimensional Database”, in Proc. of VLDB, 1997

[BR99] K. Beyer, and R. Ramakrishnan, **“**Bottom-Up Computation of Sparse and Iceberg CUBEs”, in Proc. of SIGMOD, 1999.

[GBLP96] J. Gray, A. Bosworth, A. Layman, H. Prahesh, “Data Cube: A Relational Aggregation Operator Generalizing group-BY, Cross-Tabs, and Sub-Totals”, Proc. of ICDE ’96, New Orleans, February, 1996

[GHRU97] H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman, “Index Selection for OLAP”, In Proc. 13th ICDE, Manchester, UK, 1997

[GM99] H. Gupta, I.S. Mumick, Selection of Views to Materialize Under a Maintenance-Time Constraint, Proc. International Conf. on Database Theory, 1999.

[H98] J. Hellerstein, “Data Warehousing, Decision Support & OLAP”, http://redbook.cs.berkeley.edu/lec28.html

[HCKL00] E. Hung, D. Cheung, B. Kao, Y. Liang (2000) “An optimization problem in data cube system design”. Knowledge Discovery and Data Mining, Lecture Notes in Artificial Intelligence 1805, 74-85.

[HRU96] V. Harinarayan, A. Rajaraman, J. Ullmna, “Implementing Data Cubes Efficiently”, Proc. of ACM SIGMOD, 1996.

[Li03] C. Li, “A Partial Pre-Computation of Aggregates for OLAP Databases”, M.Sc. thesis, Simon Fraser University, July, 2003.

[Luk01] W. Luk, “ADODA: A Desktop Online Data Analyzer”, Proc. of DASFAA 2001, Hong Kong, 2001

[Pendse03] N. Pendse, “Datadase Explosion”, OLAP Reports, http://www.olapreport.com/DatabaseExplosion.htm

[PS99] P. Vassiliadis, T. Sellis, “A Survey of Logical Models for OLAP Databases”, SIGMOD Record, Vol. 28, No. 4, 1999.

[SDN98] A. Shukla, P. Deshpande, J. Naughton, “Materialized View Selection for Multidimensional Datasets”, Proc. of the 24th VLDB Conf., New York, 1998

[Wit03] A. Witkowski et al, Spreadsheets in RDBMS for OLAP, Proc. of ACM SIGMOD, San Diego, 2003.

Discovering Multidimensional Structure in Relational Data

Mikael R. Jensen, Thomas Holmgren, and Torben Bach Pedersen

Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E, 9220 Aalborg Ø, Denmark
{mrj,thm,tbp}@cs.auc.dk

Abstract. On-Line Analytical Processing (OLAP) systems based on multidimensional databases are essential elements of decision support. However, most existing data is stored in "ordinary" relational OLTP databases, i.e., data has to be (re-) modeled as multidimensional *cubes* before the advantages of OLAP tools are available.

In this paper we present an approach for the automatic construction of multidimensional OLAP database schemas from existing relational OLTP databases, enabling easy OLAP design and analysis for most existing data sources. This is achieved through a set of practical and effective algorithms for discovering multidimensional schemas from relational databases. The algorithms take a wide range of available metadata into account in the discovery process, including functional and inclusion dependencies, and key and cardinality information.

1 Introduction

A prerequisite for using OLAP tools is that structures in the database are expressed in the multidimensional paradigm. Defining the structural aspects of a multidimensional database is an expert task, demanding knowledge of logical database design. Motivated by the increasing use of OLAP tools for analyzing business data and because existing OLAP design methods are not suited for non-professionals, this paper propose an automatic approach for discovering multidimensional structure from relational databases, thereby easing the process of designing a multidimensional database. This is achieved by developing practical methods for automatically discovering multidimensional schemas from relational databases. The discovery of OLAP structure falls into four general steps: (1) metadata is obtained from the data source and is (2) annotated with data, which divides the attributes into three OLAP data categories. Next, (3) interrelationships among tables are discovered and from these relationships dimensions are constructed. Finally, (4) hierarchies are discovered in each dimension.

We believe this paper to be the first to address the issue of automatic construction of multidimensional structure from existing relational databases. In [1, 5] it is described how to semi-automatically discover a multidimensional conceptual schema from an XML data source, which is easier than discovering it from relational data, because the hierarchical XML structure can often be used directly as a hierarchically organized dimension. In [11], techniques for discovering an OLAP cube based on a priori specification of queries written by the user is presented, such that the cube contains the data

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 138–148, 2004.

(and *only* the data) needed to answer all queries. In comparison, our approach requires no a priori knowledge and we instead discovers a more general structure. In [13], an approach for automatically discovering a conceptual data warehouse schema from an OLTP schema is presented, where the focus is on *conceptual* correctness, archived by generating several candidate ME/R [13] schemes and evaluating them against user requirements. Our focus is on *logical* correctness and conceptual correctness is only a secondary goal. Moreover, our approach demands no knowledge of user requirements. The approach taken in this paper uses among other things functional- and inclusion dependencies (FDs and INDs) in the data source to discover multidimensional structure. In [3, 7, 9, 10, 12] algorithms for inferring *all* such constraints are described, but as our focus is on OLAP, more efficient algorithms are constructed by only inferring constraints interesting w.r.t. OLAP. A prototype using the approach described in this paper has been implemented in Borland Delphi 6 and initial experiments shows promising results.

The remainder of this paper is organized as follows. Section 2 gives an overview of the process of discovering multidimensional structure automatically. Section 3 defines a model capturing the content of a source database. Section 4 presents algorithms for discovering inter-relationships among entities and Section 5 presents algorithms for constructing a complete multidimensional schema. Section 6 summarizes and points to topics for future research.

2 Process Overview

Our aim is to automatically discover multidimensional structure in relational data and present the structure as a *snowflake schema* [8]. The techniques are to be used prior to (or instead of) the multidimensional modeling performed in a modeling tool, thereby making the process of deducing multidimensional meaning from a database easier.

Informally, a snowflake schema is a rooted tree of tables where the root is termed a *fact table*. The fact table contains a number of *measures* which can be aggregated. Each subtree of the root is termed a *dimension* and each path from root to leaf in a dimension is termed a *hierarchy*. Each hierarchy contains one or more *levels*, where level $i + 1$ is a higher level of abstraction of the data at level i [5]. For the OLAP user, the hierarchies describe the basic navigation constructs for drill-down and roll-up operations.

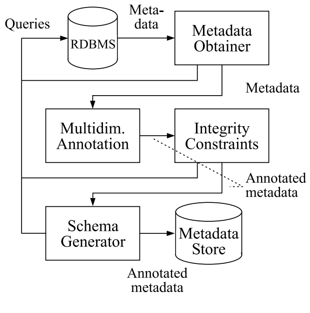


Fig. 1. System architecture

The process of discovering multidimensional structure is performed in four steps depicted in Fig. 1: First, the "Metadata Obtainer" collects metadata, such as table and attribute names, cardinality of attributes, etc. Second, the "Multidimensional Annotation" annotates the metadata with information w.r.t. OLAP; in this paper, data is divided into three categories termed the *role* of the attribute: *keys* (used for joining tables), *measures* (used for aggregation), and *descriptive* data (characterizing the data in some way, i.e., dimension data). The component "Integrity Constraints" discovers FDs and

INDs between attributes and tables and the “Schema Generator” constructs a complete snowflake schema. Finally, the snowflake schema is stored in the metadata repository “Metadata Store”.

Our working hypothesis is threefold: (1) the database does not contain composite keys, (2) the name of a foreign key should resemble the name of the primary key/table it references, and (3) each table in the database only participates in one dimension. Arguments for our working hypothesis will be given later in the paper.

3 Metadata Model

In the following a metadata model is presented giving a precise description of the metadata associated with structures in the data source. The model consists of the following sets: *Metadata* = $\{$(*table, attribute, type, null, unique, primary_key, ForeignKey, index, cardinality, distincts, frequency, Role*)$\}$; *Tables* = $\{t \mid t$ is a user-defined table$\}$; *Attributes* = $\{a \mid a$ is an attribute from a table$\}$; *Types* = $\{t \mid t$ is an SQL type$\}$; *ForeignKey* = $\{$(*table, pk*)$\}$; *Roles* = $\{$*key, measure, descriptive*$\}$; *Role* = $\{$(*role, confidence*) $\mid$ *role* $\in$ *Roles* $\wedge\ 0 \leq$ *confidence* $\leq 1\}$.

Metadata contains 12-tuples for each attribute in the data source: *table* is a name identifying the table, *attribute* is a name identifying the attribute and *type* states the type of the attribute. *null*, *unique*, and *primary_key* are booleans identifying whether or not NULL values are allowed, if the attribute is declared unique, and if the attribute has a primary key constraint. *ForeignKey* is a set stating which attribute in which table explicitly is referenced by *attribute*. The element *index* is a boolean stating whether or not the attribute has any index declarations and *cardinality*, *distincts*, and *frequency* describes the number of tuples in *table*, the number of distinct values for *attribute*, and the number of tuples satisfying an equality predicate on *attribute*. The sets *Tables* and *Attributes* describes all the names of tables defined in the data source by a user and all names of attributes in the tables. Each attribute has a type and the set *Types* consists of all the data types defined by the SQL standard. *Roles* is a set consisting of the three types of roles characterizing the attributes w.r.t. OLAP. The set *Role* contains two-tuples describing the confidence of an attributes role and always contains exactly three elements; one element for each type of role. Common for all *Role* sets is that they define a probability distribution. Initially, the set *Role* for each attribute contains (*key*, $1/3$), (*measure*, $1/3$), and (*descriptive*, $1/3$), indicating that nothing is known about the role of the attribute.

Multidimensional Annotation. When metadata is obtained, each attribute is annotated with multidimensional information. The roles for each attribute are determined using a Bayesian network [6] (depicted in Fig. 2) consisting of eight information variables (IVs) (representing the way each attribute is measured) and one hypothesis variable (HV) which contains the states “Key”, “Measure”, “Descriptive” corresponding to the three types of roles. IVs are directly connected to the HV (“Role”) and are not related in

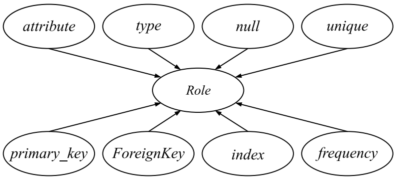


Fig. 2. Bayesian network

any way indicating that nothing can be said about the causal relations between the ways attributes are measured. Causal relations between IVs are unimportant here, because hard evidence is always given for each variable. Associated with each IV is an a priori probability distribution, i.e., one exist for attribute name, one for type, etc. See [4] for details.

Due to the simple structure of the network, the probabilities are effectively calculated as: $P(Role \mid V) = (\sum_{i=1}^{|V|} V_i[key]\ ;\ \sum_{i=1}^{|V|} V_i[measure]\ ;\ \sum_{i=1}^{|V|} V_i[descriptive])/x$ [6], where V is a set of all IVs and V_i denotes the i'th IV with $V_i[state]$ being the probability of V_i being in state *state*.

4 Discovering Database Structure

FDs and INDs [7] are useful integrity constraints w.r.t. OLAP, as they represent many-to-one relationships, where FDs are used for determining key attributes within one table and INDs are used to interrelate tables. In this paper, INDs and FDs are used to determine dimensions and hierarchies in the snowflake schema.

FDs and INDs are deduced from the the data *instance*, i.e., it is not possible to distinguish integrity constraints *holding* on the schema from integrity constraints *satisfying* the schema [14]. In this paper no distinction is made; if an integrity constraint is found in the data instance, it is assumed to hold. It is furthermore assumed that keys are not composite, which is a valid assumption due to the use of surrogate keys in real-world databases [14]. In the following it is described how candidate keys (CKs, a special case of FDs [14]) and foreign keys (FKs) are discovered and how this information is used to discover INDs.

Discovering CKs. CKs assist in acquiring INDs; in order to construct the multidimensional structure, FK/CK relationships must be resolved, such that the generated schema fulfills the structural constraints of a snowflake schema. Information regarding CKs is stored in *CandKeys* = $\{$(*table, attribute*) $\mid$ *attribute* is a CK in *table*$\}$, where each tuple describes which attribute is believed to be a CK in which table. Notice, that one table can have several CKs.

It is easy to discover CKs given the set *Metadata*: if an attribute $a \in$ *Metadata* has $a[frequency] = 1$, then *CandKeys* is updated with the tuple $(a[table], a[attribute])$. Notice, that if a table does not have an attribute uniquely identifying a complete tuple the table is not subject for further processing regarding FK/CK relationships, i.e., it is not part of the final multidimensional structure, since only tables represented in the set *CandKeys* are used for building the schema. It is not possible to use tables having no CK in the construction process, because such tables violates the strict cardinality requirements associated with snowflake schemas. The exclusion of tables without CKs does not regard the fact table, as the requirements for a snowflake schema does not dictate that a fact table must have a CK. Obviously, a fact table should not be excluded from the schema and later in this section techniques for identifying the fact table are described.

Discovering FKs. As was the case for CKs, FKs must be identified in order to discover INDs. The set *ForKeys* = $\{$(*table, attribute*) $\mid$ *attribute* is a possible FK in *table*$\}$ describes which attribute is a possible FK in which table.

```
(1) discoverPossibleFKs(Metadata, CandKeys, $\vartheta$):
(2) $\forall a \in \textit{Metadata} \setminus \{b \mid b \in \textit{Metadata} \wedge (b[\textit{table}], b[\textit{attribute}]) \in \textit{CandKeys}\}$ :
(3)   if $P_{max} - P_{min} \leq \vartheta$ then $\textit{ForKeys} = \textit{ForKeys} \cup \{(a[\textit{table}], a[\textit{attribute}])\}$ else
(4)     if $((P_{max} - P_{mid}) > (P_{mid} - P_{min})$ and $P_{max}$ value for "Key") then
(5)         $\textit{ForKeys} = \textit{ForKeys} \cup \{(a[\textit{table}], a[\textit{attribute}])\}$
(6)     else if $((P_{max} - P_{mid}) \leq (P_{mid} - P_{min})$ and $P_{max}$or$P_{mid}$ values for "Key") then
(7)         $\textit{ForKeys} = \textit{ForKeys} \cup \{(a[\textit{table}], a[\textit{attribute}])\}$
```

`discoverPossibleFKs` populates the set *ForKeys*. The pairs to consider (line 2) are all pairs in the set *Metadata* except the pairs in *CandKeys*. Notice, that *ForKeys* is a set of *possible* FKs. It is later determined whether or not these actually are *true* FKs. In the following, P_{min}, P_{mid}, and P_{max} represents the lowest, middle, and highest probability values in $a[\textit{Role}]$, respectively. The first thing to consider (line 3) is the dispersion of the values, that represents the credibility of the probability distribution; if the difference is "large", the values are not close to each other and therefore it is reasonable to assume that the multidimensional annotation of metadata have "guessed" right and vice versa (what "large" means is determined by the user-specified input parameter ϑ). In general, if the database design is "good", i.e., all primary keys are declared PRIMARY KEY, FOREIGN KEY declarations exists for all FKs, etc., then ϑ should be lower than if the design is "bad". If the values are close to each other $(a[\textit{table}], a[\textit{attribute}])$ is included in the set *ForKeys* (line 3). If the difference is "large" further checks are needed: if the difference between the highest and middle values is larger than the difference between the middle and lowest values (line 4) it is known that P_{max} is much larger than any of the two other values. If P_{max} is the value for "Key", then *ForKeys* is updated (line 5). If instead the difference between the highest and middle values is less than the difference between the middle and lowest values (line 6) it is known that P_{max} and P_{mid} are close. If P_{max} or P_{mid} are the values for "Key" then *ForKeys* is updated (line 7).

Identifying Fact Tables. The fact table (often) does not contain CKs and thus are not processed by the above algorithms. Of course, INDs linking fact table to dimensions has to be found. However, due to the size of the fact table (which is often several orders of magnitude larger than the other tables), excluding attributes from the fact table when discovering INDs improves performance. To avoid joins between the fact table and other tables when discovering INDs, the fact table has to be identified before discovering INDs. Identifying the fact table(s) is a semi-automatic process involving the user: they are identified by cardinality and by the possible presence of measures. The set *Facts* = $\{\textit{table} \mid \textit{table}$ is a fact table$\}$ contains information about which table(s) to be used as fact table(s) in the snowflake schema.

Discovering INDs. Discovering INDs is necessary in order to determine relationships between tables and forms the basis from which the different parts of the snowflake schema is constructed. INDs must be precisely determined due to the need for referential integrity between the components of a snowflake schema. For FK $\texttt{A} \in r_1$ to reference CK $\texttt{B} \in r_2$ the following must hold: $\textit{cardinality}(r_1) = \textit{cardinality}(r_1 \bowtie_{r_1.\texttt{A}=r_2.\texttt{B}} r_2)$. Information regarding INDs is stored in *Links* = $\{(\textit{source, target}) \mid \textit{source}, \textit{target} \in \{(\textit{table}, \textit{attribute}) \mid \textit{table} \in \textit{Tables} \wedge \textit{attribute} \in \textit{Attributes} \vee \textit{attribute} =$ NULL $\} \vee \textit{source}$ = NULL $\wedge\ \textit{source}[\textit{table}] \neq \textit{target}[\textit{table}] \wedge \textit{source}[\textit{attribute}]$ references $\textit{target}[\textit{attribute}]\}$, where a tuple means that *source*[*attribute*] is a FK in *source*[*table*] referencing CK *target*[*attribute*] in *target*[*table*]. Notice, that *source* can be NULL as opposed to a (*table*, *attribute*)-pair and also that the *attribute* field of either *source* or *target* can be NULL (the usage of NULL values for the tuples is described below). INDs (accidentally) holding within *one* table is not included in the set, i.e., the generated snowflake schema will not contain self-referencing hierarchy levels, which is a violation of the constraints in such a schema. In general, the set represents a non-connected, possibly cyclic digraph. The algorithm presented shortly ensures that a cycle consist of minimum two distinct tables, thus avoiding self-referencing hierarchy levels. In Section 5 it is described how a cyclic digraph (if present) is transformed into a directed tree, such that no cycles exists in the final snowflake schema.

```
(1) discoverINDs($CandKeys, ForKeys, Facts, Metadata$):
(2) $\forall a \in Metadata$ :
(3)   $\forall b \in a[ForeignKey]$ : $Links = Links \cup \{((a[table], a[attribute]), b)\}$
(4) $\{(c, d) \mid c \in ForKeys \setminus \{e \mid e \in ForKeys \wedge e[table] \in Facts\} \wedge$
(5)         $d \in CandKeys \setminus \{f \mid f \in CandKeys \wedge f[table] \in Facts\}\} \setminus Links$
(6) Sort $(c, d)$ descending by output of compare($c[attribute], d[attribute], d[table]$)
(7) $\forall (c, d)$ : if refIntTest($(c, d), Metadata$) then
(8)         $Links = Links \cup \{(c, d)\}$
(9)         $CandKeys = CandKeys \setminus \{g \mid g \in CandKeys \wedge g[table] = d[table]\}$
(10) $\forall h \in CandKeys$ : $Links = Links \cup \{(\text{NULL}; (h[table], \text{NULL}))\}$
```

discoverINDs describes how INDs are discovered. Of course, if any explicit FK declarations exist in *Metadata* they are included in *Links* before any other relationships are tested (lines 2-3). Next, all permutations of CKs and FKs are constructed, except for the combinations already added in line 3 (and *Facts* are excluded also).

To accommodate that similar names are good candidates for a FK/CK relationship (which is described below), the function compare is invoked in line 6 for all pairs, and the pairs are sorted descending according to the output of the function (a large output value means a strong name similarity between the names). Of course, all elements in the list are candidates for FK/CK relationships, but testing attributes having similar names early is likely to decrease overall processing time of the algorithm in case of match. This list is then tested for referential integrity (line 7) by invoking the function refIntTest (which is described below): if it returns true, referential integrity exists and *Links* is updated with (c, d).

If a table containing a CK has been referenced by a table containing a FK, then all elements g in *CandKeys* having the same *table* as the element d is removed from the set (line 9). The exclusion of these elements means that the table represented by d can only participate in one FK/CK relationship; in snowflake schemas, relationships between the dimensions are not allowed and the exclusion of the elements g renders relationships between dimensions impossible.

When all elements in *ForKeys* and *CandKeys* have been processed (lines 4-9) it might be the case that not all elements in *ForKeys* have been paired with elements in *CandKeys*. But since the fact table is not processed yet, the remaining elements in *CandKeys* might be paired with this table. Thus, the remaining elements in *CandKeys* are included in the set *Links* (line 10). Since no *source* is referencing the remaining elements, the field is set to NULL. Furthermore, since nothing is known about what CK is perhaps referenced (a table can have multiple CKs), the field *target*[*attribute*] is set to NULL.

Determining Similar Names. discoverINDs uses similarity between attribute names and tables as a measure for which attributes references other attributes. The idea is that if two attributes in different tables have similar names, there is a chance that one of the attributes references the other. The function compare : $str \times str \times str \longrightarrow \mathbb{R}$ compares three strings and returns a value between 0 and 1 indicating the strength of the match. When invoked, the function is given as input the name of a FK, the name of a CK, and the name of the table from which the CK originate, respectively. The name of the FK is compared to the name of the CK directly, and also to the concatenation of the name of the table and the name of the CK. We have chosen to perform both tests because we believe it is a common way of designing databases to use the name of the referred table and the referred key as the name for the FK [14]. Many off-the-shelf methods for comparing strings exist [2]. We have used a methodology based on comparing bigrams (two consecutive letters within a string) in the input strings. To some extend the strings being compared should resemble each other (e.g., one being a substring of the other) in order to get a good score.

refIntTest. As described above, INDs are discovered for distinct tables. A condition for having an IND is that the types of the attributes are the same. Furthermore, for referential integrity between two tables to hold, it is required that the table containing the CK has at least as many distinct values for the attribute as the table containing the FK. There is no need to actually check if referential integrity holds if these sub-requirements does not hold. If they hold the comparison of whether or not a FK A references a CK B is done by issuing the SQL statement "SELECT COUNT(*) FROM r_1, r_2 WHERE r_1.A = r_2.B" and comparing the result to the cardinality of relation r_1: if the numbers are equal then A references B and the function returns true, otherwise false.

5 Constructing the Multidimensional Schema

This section describes how to construct a snowflake schema based on the information discovered in the previous section. The process is divided into two parts: (1) discover elements in the set *Links* that can be used as dimensions, (2) discover hierarchies for each dimension.

Discover Dimensions. The set *Links* contains a disconnected (possibly cyclic) digraph and each *connected* graph in *Links* (denoted a *Subgraph*) represents constructs which *might* be used as dimensions (after some modification of the algorithm presented later in this section). A connected graph is a dimension if the fact tables references a "root"-node, i.e., if an IND between an attribute in a fact table and an attribute in a node of the graph exists.

We find that confining tables to be used in only *one* dimension is justified by the way databases are designed. Most "reasonable" databases aims at complying to some normal form, in far the most cases at least 3NF, which states that tables should be decomposed to avoid replication of data, but decomposed in a way such that dependencies are preserved and thus the lossless-join property is achieved [14]. This means that information closely related in nature is contained within tables and it is unlikely that a designer would construct one table describing both e.g., customers and products. This means that data contained in a single table is most likely to belong to a single dimension and not several dimensions.

The set $FactDim = \{(fact, dim) \mid fact \in \{(factTable, fk) \mid factTable \in Facts\} \wedge dim \in \{(dimTable, ck) \mid dimTable \notin Facts\} \wedge fact[fk]$ references $dim[ck]\}$ describes that $fact[fk]$ is a FK in $fact[factTable]$ referencing CK $dim[ck]$ in $dim[dimTable]$.

`discoverDimensions` describes how dimensions are identified. The algorithm is invoked once for each *Subgraph* in *Links*. The algorithm is divided into two distinct cases, depending on whether or not *Subgraph* contains a cycle. It is first described how to process a *Subgraph* containing a cycle (lines 2-11). If the graph contains a cycle, the cycle has to be "broken" before the dimension can be build, since cycles in dimensions are not allowed. All pairs (a, b) are considered, such that a is a pair $(fact, fk)$, where *fact* is a fact table and *fk* is a possible FK (line 3). Likewise, b is a pair $(table, ck)$, where *table* is a table from *Subgraph* and *ck* is a CK (line 4). Furthermore, $(table, ck)$ is participating in the cycle in *Subgraph*. The intention is to try to construct the largest dimension from *Subgraph*, thus the root of *Subgraph* must be tested first. Due to the structures that can be generated by `discoverINDs`, the cycle (if present) constitute the "root" of *Subgraph*, thus b must be part of the cycle (line 5). To accommodate that similar names are good candidates for an IND, the function compare is invoked, and the pairs are sorted descending according to the output of the function (line 6). Each pair (a, b) is then tested by the function refIntTest (line 7); if the function returns true, then $a[fk]$ references $b[ck]$ and *FactDim* is updated with the element (a, b) (line 8), and the algorithm is terminated. Notice, that having found an attribute in the cycle referenced by an attribute in the fact table triggers the removal of an edge from the cycle (line 9). Removing the edge referencing the table containing the attribute referenced by the fact-table (i.e., the newly designated root) ensures that the largest possible *tree* is used as a dimension. If a match is not found for any pairs, the algorithm is called recursively for all the subtrees pointed to by tables participating in the cycle, i.e., the input parameter *Subgraph* is now a subtree pointed to by a table in the cycle, while the tables constituting the cycle are discarded. This means that each subtree is now considered as a separate dimension instead of a part of the same dimension. In this way, one connected graph in *Links* can represent either a single dimension or a set of independent dimensions. If *Subgraph* does not contain any cycles, the second case of the algorithm (line 13) is invoked. This part is a special case of the first part, consisting in that in the first part a root for the tree had to be identified.

```
(1) discoverDimensions(Facts, ForKeys, CandKeys, Metadata, Subgraph):
(2)  if Subgraph contains a cycle then
(3)    Construct $\{(a, b) \mid a \in \{(fact, fk) \mid fact \in$ Facts $\wedge (fact, fk) \in ForKeys\} \wedge$
(4)                  $b \in \{(table, ck) \mid table \in Subgraph \wedge (table, ck) \in CandKeys \wedge$
(5)                                $(table, ck)$ is part of the cycle$\}\}$
(6)    Sort $(a, b)$ descending by output of compare($a[fk]$, $b[ck]$, $b[table]$)
(7)    $\forall (a, b)$ : if refIntTest($(a, b)$, $Metadata$) then
(8)          $FactDim = FactDim \cup \{(a, b)\}$
(9)          Transform sequence of tables $b, \ldots, i, b$ to $b, \ldots, i$
(10)         Terminate algorithm
(11)   Invoke discoverDimensions on all children of the "root"
(12) else
(13)   Construct $\{(a, b) \mid a \in \{(fact, fk) \mid fact \in Facts \wedge (fact, fk) \in ForKeys\}$
(14)                 $b \in \{(table, ck) \mid table \in Subgraph \wedge (table, ck) \in CandKeys \wedge$
(15)                               $table$ is the root of $Subgraph$ $\}\}$
(16)   Sort $(a, b)$ descending by output of compare($a[fk]$, $b[ck]$, $b[table]$)
(17)   $\forall (a, b)$ : if refIntTest($(a, b)$, $Metadata$) then
(18)         $FactDim = FactDim \cup \{(a, b)\}$
(19)         Terminate algorithm
(20)   Invoke discoverDimensions on all children of the root
```

Discover Hierarchies. Discovering the hierarchical structure of each dimension is necessary to ensure that data is aggregated correctly when performing drill-downs or roll-ups. For level l_i to roll up to l_{i+1} (denoted $l_i \rightsquigarrow l_{i+1}$) each value in l_i must be present in exactly one group made from the values of l_{i+1}. A way to test if a hierarchy generated by sorting by the number of distinct values of each attribute is correct, is to perform this grouping for each consecutive pair of hierarchy levels. If no values from level l_{i+1} are duplicated or lost when grouping on level l_i then $l_i \rightsquigarrow l_{i+1}$. This test is performed by issuing the SQL statement "SELECT COUNT(*) FROM (SELECT DISTINCT l_i, l_{i+1} FROM r)" and comparing the result to the number of distinct values on level l_i: if the numbers are equal $l_i \rightsquigarrow l_{i+1}$. Roll-up relationships are stored in *Levels* = $\{(l_i, l_{i+1}) \mid l_i, l_{i+1} \in \textit{Attributes} \wedge l_i \rightsquigarrow l_{i+1}\}$.

```
(1)  discoverHierarchies(Dim, Metadata):
(2)    Construct list $\mathcal{A}$ and sort $\mathcal{A}$ descending by the number of distincts for each attribute
(3)    for $i$ := 1 to $|\mathcal{A}|$ do
(4)      if test($n_i, n_{i+1}$) then Levels $=$ Levels $\cup \{(n_i, n_{i+1})\}$ else
(5)        if $i + 2 \leq |\mathcal{A}|$ then
(6)          for $k$ := $i + 2$ to $|\mathcal{A}|$ do
(7)            if test($n_i, n_k$) or $k$ = $|\mathcal{A}|$ then
(8)              for $h$ := $i - 1$ downto 1 do
(9)                if test($n_h, n_{i+1}$) then Levels $=$ Levels $\cup \{(n_h, n_{i+1})\}$
(10)                 if test($n_i, n_k$) then Levels $=$ Levels $\cup \{(n_i, n_k)\}$
(11)                 break inner for-loop
(12)             break outer for-loop
```

discoverHierarchies describes how hierarchy levels are discovered for each dimension. Initially a list $\mathcal{A}$ is constructed (line 2) containing all the attributes present in the dimension *Dim* given as input. $\mathcal{A}$ is sorted (line 2) in descending order according to the number of distinct values. In the following n_i denotes the attribute at position i in the list $\mathcal{A}$, and test(n_i, n_j) is a function returning either true or false depending on whether or not $n_i \rightsquigarrow n_j$, respectively (test is described below). In line 3 a for-loop is invoked; it iterates through all the elements present in $\mathcal{A}$. It is first tested (line 4) if $n_i \rightsquigarrow n_{i+1}$ and if true, then *Levels* is updated to include the pair (n_i, n_{i+1}) (line 4). If $n_i \not\rightsquigarrow n_{i+1}$ further tests are needed: The for-loop in line 6 is invoked in order to test if n_i rolls up to any attributes following n_{i+1}. If an attribute is found that n_i rolls up to (line 7), a for-loop is invoked (line 8), that ensures that the partially hierarchy structures found so far is also part of an *underlying* hierarchy, i.e., if $n_{i-1} \rightsquigarrow n_{i+1}$ (line 9). If this test is passed, *Levels* is updated with the element (n_{i-1}, n_{i+1}) (line 9) if not, the next element *below* n_{i-1} is tested. The or-part of the if-statement in line 7 is included in order to ensure that if the last attribute $a_{\text{top}} \in \mathcal{A}$ is tested and no adjacent attribute rolled up to a_{top}, perhaps some non-adjacent attribute rolls up to a_{top}. If the if-statement in line 7 is true, it is either because the roll-up test succeeded or because k is index of the last attribute in $\mathcal{A}$. In order to ensure that perhaps $n_i \rightsquigarrow n_k$, the test is performed again in line 10 and if $n_i \rightsquigarrow n_k$, *Levels* is updated (line 10). If the roll-up tests succeeded in line 7 or 9, the for-loops in line 6 and 8 are "broken" (lines 11 and 12) and the next attributes in $\mathcal{A}$ are used as n_i and n_{i+1}.

The function test SQL-tests its input-attributes as described above, but since the function is used extensively and SQL-tests are expensive, dynamic programming techniques are used to speed up evaluation: for every invocation, the result of testing attributes (i,j) is placed in a two-dimensional array at position (i, j), where the position contains *true* if $i \rightsquigarrow j$, *false* if $i \not\rightsquigarrow j$, and *N/A* otherwise. It is then first checked if the array contains a result for the attributes in question and if so, the result is returned without performing an SQL-test. Furthermore, the array is populated with implicit knowledge of roll-up relationships based on the discovered CKs and INDs. For every table, the CK is known and trivially functionally determines every attribute in the table [14], i.e., the CK is known to roll-up to all the other attributes in the table. Moreover, if a FK is referencing a CK then trivially FK $\rightarrow$ CK and thereby FK $\rightsquigarrow$ CK. Based on this implicit knowledge, further knowledge is discovered: from the known FDs, all FDs logically implied are calculated and these new relationships are stored in the array. In this way, `discoverHierarchies` utilizes information about roll-up relationships and many SQL-tests are avoided as opposed to not using the array structure.

6 Conclusions and Future Work

Motivated by the increasing use of OLAP tools for analyzing business data and because existing OLAP design methods requires knowledge of multidimensional design techniques and extensive knowledge of the database being modeled, this paper proposed a practical approach for discovering multidimensional structure from OLTP databases. The approach ensures the general quality of the generated schema by discovering only correct multidimensional structure in the data source.

The discovery of OLAP structure falls into four general steps: (1) obtain metadata from the data source and (2) annotate it with OLAP data category information. Next, (3) discover inter-relationships among tables and construct dimensions from these relationships. Finally, (4) discover hierarchies within each dimension. Based on experiments with a prototype using the approach described in this paper, we believe that the approach is well suited for automatic OLAP DB construction, even for large databases.

In future work, it should be investigated how to also take composite keys into account and which combinations of attributes to use as possible composite keys. Furthermore, the a priori probabilities used in the Bayesian network should be based on experiments on a large number of real-world databases. An incremental discovery approach should be investigated in case that changes made to the source data gives rise to a schema change for the multidimensional database. The next immediate step is to conduct qualitative analysis of resulting schemas in order to make sure our method is applicable to a broad range of different data sets.

References

1. M. Golfarelli, S. Rizzi, and B. Vrdoljak: Data Warehouse Design from XML Sources. In Proc. of *DOLAP*, pp. 40-47, 2001
2. P. A. V. Hall, and G. R. Dowling: Approximate String Matching. *ACM Computing Surveys 12(4)*, pp. 381-402, 1980
3. Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen: Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In Proc. of *ICDE*, pp. 392-401, 1998
4. M. R. Jensen and T. Holmgren: Discovering Multidimensional Structure in Relational Data. *Thesis Report, Department Of Computer Science, Aalborg University*, 2002

5. M. R. Jensen, T. H. Møller, and T. B. Pedersen: Specifying OLAP Cubes On XML Data. *JIIS 17(2-3)*: 255-280, 2001
6. F. V. Jensen: *Bayesian Networks and Decision Graphs*, Springer-Verlag, 2001
7. M. Kantola, H. Mannila, K. Räihä, and H. Siirtola: Discovering Functional and Inclusion Dependencies in Relational Databases. *JIIS 7*: 591-607, 1992
8. R. Kimball: *The Data Warehouse Toolkit*, 2. Ed., Wiley, 2002
9. H. Mannila and K. Räihä: Algorithms for inferring functional dependencies from relations. *DKE 12(1)*: 83-99, 1994
10. F. Marchi, S. Lopes, and J. Petit: Efficient Algorithms for Mining Inclusion Dependencies. In Proc. of *EDBT*, pp. 464-476, 2002
11. T. Niemi, J. Nummenmaa, and P. Thanisch: Constructing OLAP Cubes Based on Queries. In Proc. of *DOLAP*, pp. 9-15, 2001
12. N. Novelli and R. Cicchetti: FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies. In Proc. of *ICDT*, pp. 189-203, 2001
13. C. Phipps and K. C. Davis: Automating Data Warehouse Conceptual Schema Design and Evaluation. In Proc. of *DMDW*, pp. 23-32, 2002
14. A. Silberschatz, H. Korth, and S. Sudarshan: *Database System Concepts*, 4. Ed., Mc Graw Hill, 2001

Inductive Databases as Ranking

Taneli Mielikäinen

HIIT Basic Research Unit
Department of Computer Science
University of Helsinki, Finland
Taneli.Mielikainen@cs.Helsinki.FI

Abstract. Most of the research in data mining has been focused on developing novel algorithms for specific data mining tasks. However, finding the theoretical foundations of data mining has recently been recognized to be even greater concern to data mining.

One promising candidate to form a solid basis for data mining is known as inductive databases. The inductive databases are databases with a tight integration to data mining facilities. However, it is not clear what inductive databases actually are, what they should be and whether inductive databases differ notably from the usual databases with slightly broader notions of queries and data objects.

In this paper we aim to show that the viewpoint offered by inductive databases differs from the usual databases: the inductive databases can be seen as databases with ability to rank data manipulation operations. We describe how several central data mining tasks can be naturally defined by this approach and show that the proposed inductive databases framework offers conceptual benefits by clarifying and unifying the central data mining tasks. We also discuss some challenges of inductive databases based on query ranking and grading.

1 Introduction

Data mining aims to extract useful knowledge from large collections of (observational) data [1, 2]. The largest effort in data mining has been devoted on developing novel methods to reach for several correctives of this goal. For example, there has been considerable amount of activity to invent techniques for pattern discovery, classification, clustering and density estimation.

However, although the general goal of data mining is not very well-defined, much less work has been devoted on finding out what that goal actually means, i.e., whether there exists the "theory of data mining" that captures the central tasks but is not too broad, essentially an empty concept. Recently this has increasingly been recognized to be one of the most important current challenges for the data mining community.

Although the main focus in data mining research has been in the development of data analysis algorithms, there has also been some work on examining possibilities to theoretical foundations of data mining [2, 3]. There are two main requirements for the theory of data mining. First, it should capture most of the

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 149–158, 2004.

central data mining tasks naturally. Second, it should clarify and unify different data mining tasks. Thus, the theory of data mining should be useful in practice but still the theory should actually say something about the commonalities of the vast methods known in data mining and support the knowledge discovery process.

One promising candidate to serve as the theory of data mining is known as *inductive databases* [4, 5]. The basic idea of inductive databases is to extend database technologies in such way that it enables the databases to support data mining and the knowledge discovery process. However, it is not clear what this actually means. In this paper try to clarify this by suggesting that the most essential difference of inductive databases compared to usual databases is the ability to rank or grade queries, i.e., to suggest promising queries based on the preference function determined by the data analyst.

The rest of the paper is organized as follows. In Section 2 we discuss about the foundations of data mining and the current view to inductive databases. In Section 3 we describe how inductive databases can be seen as an extension of usual databases with ranking abilities. In Section 4 we show how the central data mining tasks can be naturally expressed by the proposed inductive databases framework and discuss about some of the challenges they raise. Section 5 concludes the paper.

2 On Foundations of Data Mining and Inductive Databases

Data mining is currently a very popular research topic. However, relatively small effort has been put on finding a general frameworks for data mining.

Since data mining has immediate commonalities with statistics and machine learning, one might claim that data mining does not exist but it is just another buzzword for getting more funding to apply techniques of statistics and machine learning. However, this is not exactly true since there are also several distinctive features [3]. For example, summarizing a given data set (e.g., by means of data reduction [6] or data compression [7]) even without making any statistical assumptions is an important task in data mining but this task does not fit very well to statistics nor to machine learning. Furthermore, data mining is typically iterative and interactive, an exploratory process of data analysis. This viewpoint is not emphasized in statistics nor in machine learning. In fact, it could even cause some troubles for them.

Although the Grand Unified Theory of Data Mining would be useful to support the existence of data mining as a scientific discipline, there is even greater need for that because of practical reasons: systematizing the field of data mining would benefit practical data mining of being more than a bunch of techniques.

A very promising candidate for the theory of data mining is offered by the inductive databases [4, 5]. Inductive databases are databases that contain inductive generalizations about the data, in addition to the usual data [8]. These inductive generalizations can be e.g. clusterings, classifications or patterns. Clearly, the inductive databases can naturally support exploratory data analysis in terms of ad-hoc querying to the inductive database [9]. The high-level notion of inductive databases captures all major data mining tasks, such as pattern discovery, density estimation, clustering, and prediction. However, it is not clear how these tasks should be expressed in the inductive databases framework and what the inductive databases framework should be to express these tasks in a fruitful way.

Most of the existing work on inductive databases has been rooted on pattern discovery and the proposed inductive databases are mostly extensions of usual database query languages [4, 10–12]. The typical model for inductive databases consists of a data component and a pattern component, and both components can be queried [4]. Restricting inductive databases to pattern discovery can be motivated by the fact that there are already a large number of practical primitives for pattern discovery available and also applicable general frameworks have been sketched [13, 14].

However, the theory of data mining should capture all major data mining tasks naturally in order to become commonly accepted and supported. Although pattern discovery can be expressed quite naturally in the current models for inductive databases, expressing some other data mining tasks such as clustering in a natural way is not so easy. In the next section we propose a notion of inductive databases that can be used to naturally describe all central data mining tasks in a uniform way.

3 Inductive Databases as Ranking

A usual database consists of two components: a *data model* and *data* [15]. Also the data model consists of two components: a *data schema* which essentially restricts the collection $\mathcal{D}$ of possible data, and the collection $\mathcal{Q}$ of *data manipulation operations* such as queries and update operations. Thus, for our purposes a usual database can be considered as a triplet $(\mathcal{D}, \mathcal{Q}, D)$ where $\mathcal{D}$ is the collection of possible data, $\mathcal{Q}$ is the collection of possible data manipulation operations and $D \in \mathcal{D}$ is the actual data in the database. The collections $\mathcal{D}$ and $\mathcal{Q}$ are considered to be implicitly represented.

The central question in data mining can be expressed as "I have this data. What should I ask about it?" The results of most data mining tasks can be seen also as queries and their evaluation results. (See Section 4 for examples.) Thus, if the user has a database, the question become even more concrete: "I have this database. Which queries are relevant for this data?" Without any knowledge about the interests of the user there are no meaningful answers. Fortunately, expressing the preferences (in some precision) is usually possible even when choosing the most relevant queries directly from often enormously large collection of all queries is too difficult for the user.

This immediately gives rise to the following notion of inductive databases. An inductive database consists of the components of a usual database and a collection $\mathcal{R}$ of *rankings* of the data manipulation operations. (Most of the cases it would be sufficient to consider only rankings of queries but conceptually there is no need for that restriction.) Thus, an *inductive database* is a 4-tuple $(\mathcal{D}, \mathcal{Q}, \mathcal{R}, D)$.

A ranking is a list of data manipulation operations in $\mathcal{Q}$ where each data manipulation operation occurs at most once. The most important operation for rankings is popping the head of the list, i.e., fetching the next best data manipulation operation w.r.t. the preference function induced by the ranking.

Already this simple operation is sufficient for many data mining tasks. For example, also popping the last operation from the (possibly infinite) list would be many times useful (see Section 4.1 for an example of this in the context of frequent itemset mining) but conceptually there is no need for any new primitives since popping the last element from the list is conceptually equivalent to popping the first element from the reverse of the list, i.e., fetching the next worst operation from the ranking is conceptually equivalent to fetching the next best operations from the inverse of that ranking.

Sometimes, however, there is need for also a few other operations. For example, it would occasionally be very useful to be able to compare the usefulness of the data manipulation operations and also to compute the actual usefulness values. Fortunately, these operations are often defined as a side product of the ranking. The reason why they are not required in the proposed inductive databases framework is that computing them can be much more difficult that ranking: Comparing two arbitrary data manipulation operations can be very difficult even when fetching the next best operation is trivial. Also, meaningful usefulness values do not always even exist. For example, the ranking can be partially done by a human expert that can only express her preferences and not the actual usefulness values. This situation occurs sometimes also due to privacy issues. In practice, the rankings will be augmented with some additional information. Especially the information whether two consecutive operations in the ranking are equally good will often be of interest.

One particularly nice aspect of the proposed inductive databases framework is that the inductive database preserves the closure properties of the underlying usual database since the essential difference between the inductive and the usual databases is the ranking function.

4 Expressing Data Mining Tasks by Rankings

Although the inductive databases framework proposed in the previous section seems to be quite clean and simple, it is not yet clear whether the standard data mining tasks can be naturally expressed by it. In this section we show how the central tasks in data mining can be expressed in a valuable way by the proposed inductive databases framework. We describe how different data mining tasks can be seen as queries and query evaluations. We also highlight some challenges of defining and computing the rankings in data mining.

4.1 Pattern Discovery

Pattern discovery is an important subfield in data mining where the goal is to find all interesting patterns in the given data [16, 17]. The interesting patterns can be e.g. local regularities in the data, or parsimonious summaries of subsets of data [18].

The most prominent examples of interesting patterns are *frequent itemsets* and *association rules* [19, 20]. Let D be a bag (i.e., a multi-set) consisting of finite subsets (called *transactions*) of a set I of *items.* The bag D called a *transaction database.*

An *itemset* X is a subset of I. The frequency $\mathit{fr}\,(X, D)$ of the itemset $X \subseteq I$ in a transaction database D is the fraction of sets in D containing X, i.e., $\mathit{fr}\,(X, D) = |\{Y \in D : X \subseteq Y\}| \,/\, |D|$ where all collections of sets are interpreted as bags. The frequency $\mathit{fr}\,(X, D)$ can be seen as the empirical probability of X. The interestingness of an itemset is considered to be its frequency in the given transaction database.

An *association rule* $X \Rightarrow Y$ consists of two itemsets $X, Y \subseteq I$. The accuracy of an association rule $X \Rightarrow Y$ in D is the fraction of transactions containing Y that also contain X, i.e., $\mathit{acc}\,(X \Rightarrow Y, D) = \mathit{fr}\,(X \cup Y, D)/\mathit{fr}\,(Y, D)$. From the viewpoint of probabilities the accuracy $\mathit{acc}\,(X \Rightarrow Y, D)$ can be interpreted as an the conditional empirical probability of X given Y.

However, computing all frequent itemsets or association rules is not feasible since the number of them is exponential in the number of items in I. Instead, the practical goal has been finding all itemsets $X \subseteq I$ that are σ-frequent in D, i.e., the collection $\mathcal{F}\,(\sigma, D) = \{X \subseteq I : \mathit{fr}\,(X, D) \geq \sigma\}$, and σ-frequent δ-accurate association rules $X \Rightarrow Y$ with X and Y being σ-frequent in D and having no common items.

In practice, several minimum threshold values σ and δ have to be tried since it is difficult to find a good trade-off between the understandability of the pattern collection and its descriptive power. Instead of the minimum threshold values, one can specify the upper bound for the number of patterns to be produced but still the same problem of finding a suitable parameter value exists. Although the association rules are usually considered as the actual end product, most of the attention has been devoted for finding collection of frequent itemsets [21] since association rules can be produced from them as a simple post-processing step [19].

The proposed inductive databases framework fits perfectly for mining frequent itemsets and association rules: Itemsets and association rules can be seen as queries. The query evaluation result for an itemset $X \subseteq I$ in D corresponds to the bag $\mathit{cover}\,(X, D) = \{Y \in D : X \subseteq Y\}$ and the query evaluation result for an association rule $X \Rightarrow Y$ corresponds to two bags $\mathit{cover}\,(Y, D)$ and $\mathit{cover}\,(X \cup Y, D)$. Conceptually the association rule query can be seen to consist of first selecting the transactions of $\mathit{cover}\,(Y, D)$ from D and second $\mathit{cover}\,(X \cup Y, D)$ from $\mathit{cover}\,(Y, D)$ since $\mathit{cover}\,(X \cup Y, D) \subseteq \mathit{cover}\,(Y, D)$.

The ranking of itemsets based on their frequencies can be claimed to solve the frequent itemset mining task from the user point of view even better than finding all σ-frequent sets since the inductive database naturally supports the interactive examination of the itemsets. It also gives a slightly different viewpoint to frequent itemset mining by interpreting the frequent itemset mining as generating frequent itemsets on demand in the order of decreasing frequencies instead of computing the collection for different threshold values.

Instead of the itemsets with high frequencies (i.e., regularities in the data), one can be interested also in the very rare itemsets, i.e., the surprising itemsets. As mentioned in the previous section, conceptually this is only inverting the ranking. Also in practice it is feasible to look for infrequent itemsets in addition to the frequent ones [22].

Clearly, mining also other kinds of interesting patterns fits immediately to this framework: the patterns are listed in the order of decreasing interestingness. Thus, it can be said that ranking suits quite well for pattern discovery.

4.2 Density Estimation

In addition to local modeling, also global modeling is important in data mining. The central global modeling tool is the estimation of probability distributions.

Based on Section 4.1 it is clear that we could present probability distributions of a given class in decreasing order of their likelihood w.r.t. the data. The ranking approach bends naturally also for other kinds of density estimation tasks. In this section we shall show how it can be used to construct refining representation of a probability distribution as a mixture (i.e., a convex combination) of simpler distributions. This kind of hierarchical modeling occurs frequently in data summarization since defining the right trade-off between understandability and accuracy of the model in advance is usually very difficult.

Let us consider the situation where the database D consists of points in $\mathbb{R}^d$ and we try to model the data by a mixture of d-dimensional Gaussians. Let

$$\mathbb{P}\left(p \mid G_1,\ldots,G_m,\alpha_1,\ldots,\alpha_m\right) = \sum_{i=1}^{m} \alpha_i \mathbb{P}\left(p \mid G_i\right)$$

be the likelihood of $p \in D$ given the Gaussians $G_1,\ldots,G_m$ with weights $\alpha_1,\ldots,$ α_m such that $\sum_{i=1}^{m} \alpha_i = 1$ and $\alpha_i \geq 0$ for all $i = 1,\ldots,m$. The joint likelihood of all points in D is similarly

$$\mathbb{P}\left(D \mid G_1,\ldots,G_m,\alpha_1,\ldots,\alpha_m\right) = \prod_{p \in D} \mathbb{P}\left(p \mid G_1,\ldots,G_m,\alpha_1,\ldots,\alpha_m\right).$$

Thus, the queries are mixtures of Gaussians and the query evaluation in D results the associated likelihoods for all points in D.

The refining mixture of Gaussians for the data can be computed as follows. The first Gaussian G_1 is the one with the maximum likelihood w.r.t. the data D. The second Gaussian G_2 is the one with maximum likelihood w.r.t. the data points $p \in D$ weighted by $1 - \mathbb{P}\left(p \mid G_1, 1\right)$ and the weights α_1 and α_2 are chosen in such a way that they maximize the likelihood $\mathbb{P}\left(D \mid G_1, G_2, \alpha_1, \alpha_2\right)$. Similarly the kth Gaussian G_k is the maximum likelihood estimate w.r.t. the data D weighted by $1 - \mathbb{P}\left(p \mid G_1,\ldots,G_{k-1},\alpha_1,\ldots,\alpha_{k-1}\right)$ and α_k is chosen and $\alpha_1,\ldots,\alpha_{k-1}$ are scaled in such way that the updated values $\alpha_1,\ldots,\alpha_k$ maximize the likelihood $\mathbb{P}\left(D \mid G_1,\ldots,G_k,\alpha_1,\ldots,\alpha_k\right)$.

This way a refining description of the data can be represented by ranking. The previous queries (i.e., the previous mixture of $k-1$ Gaussians) are exploited when computing the next best query (i.e., the mixture of k Gaussians). A similar rankings occur also when ordering patterns w.r.t. their informativeness [23].

4.3 Clustering

Maybe the most important task in data mining is clustering, i.e., grouping data into groups consisting of similar data. For clusterings, the proposed inductive databases framework seems to be especially suitable: Defining clusterings unambiguously is not usually very easy and thus several alternative suggestions, possibly with associated quality values for the clusterings, are most welcome.

Let us consider k-center clustering of points in $D \subset \mathbb{R}^d$ and let the goodness of the clustering be measured by Euclidean distance. Then the queries correspond to k points in $\mathbb{R}^d$ and the query evaluations result partitions of D into k groups. (Note that also hierarchical clustering can be described by ranking, similarly to the refining mixtures in Section 4.2.)

An important aspect in ranking clusterings is what we would really like to see is the listing of clusterings in such order that each prefix of the list contains a representative collection of clusterings. For example, in the case of k-center clustering w.r.t. Euclidean distance, the straightforward application of the distance function would typically produce quite uninformative ranking: all good clusterings would be essentially the same. Thus, after finding the best clustering, we should be able to rule out the clusterings that are too similar to it. In practice, this goal can be reached for by stochastic search techniques.

Although listing the clusterings in a very good order is not trivial, it is clear that the rankings offer a natural way to describe the clustering tasks.

4.4 Prediction

Another data mining task with high practical relevancy is prediction of outcomes of some unknown function for a given data based on examples of outcomes for some other data points. A well-known special case of the prediction task is classification, i.e., labeling unlabeled data based on another set of labeled data (called the *training data*). The query in this case is the classifier and its evaluation on D results the classification of the data points in D which is typically a partition of D.

There are several established methods for the classification task, see e.g. [24]. Furthermore, the representations of classifiers can differ very much. This is not desirable for the inductive databases since one of the goals of the inductive databases is to have only a small number of primitives succinct for data mining. A conceptually simple solution to this problem of different representations would be to explicitly compute labelings. However, this does not make sense in general, as the goal of classification is to *generalize* the knowledge extracted from the training data to possibly infinite collections of data [25]. Also, finding a common, understandable representation for different classifiers can be very difficult. For example, representing a hyperplane with a decision tree with splits of one variable at a time is usually very awkward (and vice versa).

Based on the above observations it might seem that inductive databases can hardly be useful in prediction tasks since there is need for possibility to represent many kinds of predictors. However, the goal of prediction does not depend on the method. For example, in the case of classification, the goal is to find classifiers that are accurate also for still unclassified (and possibly unseen) data. Thus, although the actual predictors can differ much from each other, the same ranking functions can be used for all predictors solving the same prediction task at hand. Of course, sometimes the ranking function has to be slightly modified based on e.g. the preferences of the user w.r.t. different classifiers. For example, one could prefer Bayesian methods or decision trees with small depth regardless of their estimated prediction performance.

Thus, prediction tasks can be comfortably expressed in the proposed inductive databases framework and the inductive databases framework can support also the construction of the actual inductive database since same ranking functions can be used for all different methods solving the same prediction task.

5 Conclusions

In this paper we have described how inductive databases can be seen as ranking data manipulation operations of usual databases. We have proposed an inductive databases framework based on ranking that gives clean and simple definition what the inductive databases essentially could be. Furthermore, we have shown that the framework is also very suitable for expressing the central data mining tasks naturally and it can give even some new insights to them. We believe that the concept of ranking is relevant for data mining in general due to the exploratory nature of data mining.

However, this paper is still far from concrete implementations of general purpose inductive databases. There are many important open questions relevant for the suggested concept of inductive databases ranging from technical database theory questions to understandability and relevancy questions:

- How to summarize the ranked queries? There is often a need for summarizing even the rankings. For example, recently pattern discovery research has been focused on summarizing the collection of interesting patterns under the flag of condensed representations, see e.g. [26, 27] and the references therein.
- What kind of ranking languages are useful and usable? It is clear that rankings should be described by languages that are suitable for that task. Furthermore, it would be desirable if the expressive power and the computational complexity of the language used for expressing the rankings are low.
- What kind of trade-offs there are between generality, efficiency and usefulness of the ranking languages?
- Are there non-trivial trade-offs and relations between expressive powers of data definition languages, data manipulation languages and ranking languages?

- Is there a need for higher order queries? Higher order queries, e.g. queries of queries, can be expressed by the rankings. However, there might be situations where the real higher order queries would be truly beneficial.

Acknowledgments

I wish to thank Floris Geerts, Bart Goethals, Matti Kääriäinen and Heikki Mannila for exciting discussions on foundations of data mining and the nature of inductive databases.

References

1. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)
2. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Academic Press (2001)
3. Mannila, H.: Theoretical frameworks for data mining. SIGKDD Explorations **1** (2000) 30–32
4. De Raedt, L.: A perspective on inductive databases. SIGKDD Explorations **4** (2003) 69–77
5. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of The ACM **39** (1996) 58–64
6. Barbará, D., DuMouchel, W., Faloutsos, C., Haas, P.J., Hellerstein, J.M., Ioannidis, Y.E., Jagadish, H.V., Johnson, T., Ng, R.T., Poosala, V., Ross, K.A., Sevcik, K.C.: The new jersey data reduction report. IEEE Data Engineering Bulletin **20** (1997) 3–45
7. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. 3rd edn. Texts in Computer Science. Springer-Verlag (1997)
8. Mannila, H.: Inductive databases and condensed representations for data mining. In Maluszynski, J., ed.: Logic Programming, Proceedngs of the 1997 International Symposium, Port Jefferson, Long Island, N.Y., October 13-16, 1997. MIT Press (1997) 21–30
9. Boulicaut, J.F., Klemettinen, M., Mannila, H.: Modeling KDD processes within the inductive database framework. In Mohania, M.K., Tjoa, A.M., eds.: Data Warehousing and Knowledge Discovery, First International Conference, DaWaK '99, Florence, Italy, August 30 - September 1, 1999, Proceedings. Volume 1676 of Lecture Notes in Artificial Intelligence. Springer (1999) 293–302
10. Giannotti, F., Manco, G.: Querying inductive databases via logic-based used-defined aggregates. In Zytkow, J.M., Rauch, J., eds.: Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings. Volume 1704 of Lecture Notes in Artificial Intelligence. Springer (1999) 125–135
11. Imieliński, T., Virmani, A.: MSQL: A query language for database mining. Data Mining and Knowledge Discovery **3** (1999) 373–408
12. Jeudy, B., Boulicaut, J.F.: Constraint-based discovery and inductive queries: Application to association rule mining. [28] 110–124
13. De Raedt, L., Jaeger, M., Lee, S.D., Mannila, H.: A theory of inductive query answering. In Kumar, V., Tsumoto, S., eds.: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan. IEEE Computer Society (2002) 123–130

14. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery **1** (1997) 241–258
15. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
16. Hand, D.J.: Pattern detection and discovery. [28] 1–12
17. Mannila, H.: Local and global methods in data mining: Basic techniques and open problems. In Widmayer, P., Ruiz, F.T., Bueno, R.M., Hennessy, M., Eidenbenz, S., Conejo, R., eds.: Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings. Volume 2380 of Lecture Notes in Computer Science. Springer (2002) 57–68
18. Fayyad, U., Uthurusamy, R.: Evolving data mining into solutions for insights. Communications of the ACM **45** (2002) 28–31
19. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993. ACM Press (1993) 207–216
20. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996) 307–328
21. Goethals, B., Zaki, M.J., eds.: Proceedings of the Workshop on Frequent Itemset Mining Implementations (FIMI-03), Melbourne Florida, USA, November 19, 2003. Volume 90 of CEUR Workshop Proceedings. (2003) http://CEUR-WS.org/Vol-90/.
22. Mielikäinen, T.: Intersecting data to closed sets with constraints. [21]
23. Mielikäinen, T., Mannila, H.: The pattern ordering problem. [29] 327–338
24. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer-Verlag (2001)
25. Vapnik, V.N.: The Nature of Statistic Learning Theory. Statistics for Engineering and Information Science. Springer-Verlag (2000)
26. Calders, T., Goethals, B.: Minimal k-free representations of frequent sets. [29] 71–82
27. Mielikäinen, T.: Separating structure from interestingness. In Dai, H., Srikant, R., Zhang, C., eds.: Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings. Volume 3056 of Lecture Notes in Artificial Intelligence. Springer (2004) 476–485
28. Hand, D.J., Adams, N.M., Bolton, R.J., eds.: Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK, September 16-19, 2002, Proceedings. Volume 2447 of Lecture Notes in Computer Science. Springer (2002)
29. Lavrac, N., Gamberger, D., Blockeel, H., Todorovski, L., eds.: Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings. Volume 2838 of Lecture Notes in Artificial Intelligence. Springer (2003)

Inductive Databases of Polynomial Equations

Sašo Džeroski, Ljupčo Todorovski, and Peter Ljubič

Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract. Inductive databases (IDBs) contain both data and patterns. Here we consider IDBs where patterns are polynomial equations. We present a constraint-based approach to answering inductive queries in this domain. The approach is based on heuristic search through the space of polynomial equations and can use subsumption and evaluation constraints on polynomial equations. We evaluate this approach on standard regression problems. We finally consider IDBs containing patterns in the form of polynomial equations as well as molecular fragments, where the two are combined in order to derive QSAR (Quantitative Structure-Activity Relationships) models.

1 Introduction: Constraints in Inductive Databases

Inductive databases [7] embody a database perspective on knowledge discovery, where knowledge discovery processes are considered as query processes. In addition to normal data, inductive databases contain patterns (either materialized or defined as views). Data mining operations looking for patterns are viewed as queries posed to the inductive database. In addition to patterns (which are of local nature), models (which are of global nature) can also be considered.

A general formulation of data mining [9] involves the specification of a language of patterns and a set of constraints that a pattern has to satisfy with respect to a given database. The set of constraints can be divided in two parts: language and evaluation constraints. The first only concern the pattern itself, the second concern the validity of the pattern with respect to a database.

Inductive queries consist of constraints. The primitives of an inductive query language include language constraints (e.g., find association rules with item A in the head) and evaluation primitives. The latter are functions that express the validity of a pattern on a given dataset. We can use these to form evaluation constraints (e.g., find all item sets with support above a threshold) or optimization constraints (e.g., find the 10 association rules with highest confidence).

Constraints thus play a central role in data mining and constraint-based data mining is now a recognized research topic [1]. The use of constraints enables more efficient induction as well as focussing the search for patterns on patterns likely to be of interest to the end user. Several approaches exist that use constraints for predictive models, such as size and accuracy constraints in decision trees [5].

Most work on constraint-based induction, however, is concerned with the discovery of frequent patterns. Different types of data and patterns have been considered, including itemsets, episodes, Datalog queries, and graphs. Designing

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 159–168, 2004.

inductive databases for these types of patterns involves the design of inductive query languages and solvers for the queries in these languages. For each type of pattern, or pattern domain, a specific solver is designed, following the philosophy of constraint logic programming [2].

Here we consider IDBs that contain models in the form of polynomial equations: constrains on these are considered and a heuristic solver is proposed. We evaluate the use of this solver on standard regression problems. We finally consider IDBs containing both the pattern domains of equations and molecular fragments, as well as combining them in order to derive QSAR (Quantitative Structure-Activity Relationships) models.

2 The Pattern Domain of Polynomial Equations

Here we consider the pattern domain of polynomial equations. We first define the language of polynomial equations, then consider syntactic/subsumption constraints on these. We next define several evaluation primitives for equations and finally discuss inductive queries in this domain.

2.1 The Language of Polynomial Equations

Given a set of variables V, and a dependent variable $v_d \in V$, a polynomial equation has the form $v_d = P$, where P is a polynomial over $V \setminus \{v_d\}$. A polynomial P has the form $\sum_{i=1}^{r} c_i \cdot T_i$, where T_i are multiplicative terms, and c_i are real-valued constants. Each term is a finite product of variables from $V \setminus \{v_d\}$, $T_i = \prod_{v \in V \setminus \{v_d\}} v^{d_{v,i}}$, where $d_{v,i}$ is (a non-negative integer) degree of the variable in the term. The degree of 0 denotes that the variable does not appear in the term. The sum of degrees of all variables in a term is called the degree of the term, i.e., $\deg(T_i) = \sum_{v \in V \setminus \{v_d\}} d_{v,i}$. The degree of a polynomial is the maximum degree of a term in that polynomial, i.e., $\deg(P) = \max_{i=1}^{r} \deg(T_i)$. The length of a polynomial is the sum of the degrees of all terms in that polynomial, i.e., $\text{len}(P) = \sum_{i=1}^{r} \deg(T_i)$.

2.2 Syntactic Constraints

We will consider two types of syntactic constraints on polynomial equations: parametric and subsumption constraints. Parametric constraints set upper limits for the degree of a term (in both the LHS and RHS of the equation), as well as the number of terms in the RHS polynomial. For example, one might be interested in equations of degree at most 3 with at most 4 terms. Such parametric constraints are taken into account by the equation discovery system Lagrange [4].

Of more interest are subsumption constraints, which bear some resemblance to subsumption/generality constraints on terms/clauses in first-order logic. A term T_1 is a sub-term of term T_2 if the corresponding multi-set M_1 is subset of the corresponding multi-set M_2. For example, xy^2 is sub-term of x^2y^4z.

The sub-polynomial constraint is defined in terms of the sub-term constraint. Polynomial p_1 is a sub-polynomial of polynomial p_2 if each term in p_1 is a sub-term of some term in p_2. There are two options here: one may, or may not, require that each term in p_1 is a sub-term of a different term in p_2.

In looking for interesting equations, one might consider constraints of the form: LHS is a sub-term of t, t is a sub-term of LHS, RHS is a sub-polynomial of p, and p is a sub-polynomial of RHS. Here t and p are a term and a polynomial, respectively. These set upper and lower bounds on the lattice of equation structures, induced by the relations sub-term and sub-polynomial.

Consider the following constraint: LHS is a sub-term of x^2y and both xy and z are sub-polynomials of RHS. The equation $xy = 2x^2y^2 + 4z$ satisfies this constraint, under both interpretations of the sub-polynomial constraint. The equation $x^2y = 5xyz$, however, only satisfies the constraint under the first interpretation (different terms in p_1 can be sub-terms of the same term in p_2).

2.3 Evaluation Primitives

The evaluation primitives for equations calculate different measures of the degree of fit of the equation to a given dataset/table. Assume that i is an index that runs through records/rows of a database table. Denote with y_i the value of the LHS of a given equation on record i; with $\hat{y_i}$ the value of the RHS as calculated for the same record; and with $\overline{y}$ the mean value of y_i over the table records.

Six measures for the degree of fit for an equation to a dataset, well known from statistics, are: multiple correlation coefficient R and normalized standard deviation S ($R^2 = 1 - \frac{\sum_{i=1}^N (y_i-\hat{y}_i)^2}{\sum_{i=1}^N (y_i-\overline{y})^2}$; $S^2 = \frac{\frac{1}{N}\sum_{i=1}^N (y_i-\hat{y}_i)^2}{\overline{y}^2+e^{-\overline{y}^2}}$), mean/maximum absolute error ($MeanAE = \frac{1}{N}\sum_{i=1}^N |\hat{y}_i - y_i|$, $MaxAE = \max_{i=1}^N |\hat{y}_i - y_i|$), mean/root mean square error ($MSE = \frac{1}{N}\sum_{i=1}^N (\hat{y}_i - y_i)^2$, $RMSE = \sqrt{MSE}$).

2.4 Inductive Queries in the Domain of Equations

Inductive queries are typicaly conjunctions of primitive constraints, e.g., language and evaluation constraints. Evaluation constraints set thresholds on acceptable values of the evaluation primitives: $M(e, D) < v$; $M(e, D) > v$, where v is a positive constant and M is one of the measures defined above. Instead of evaluation constraints one can consider optimization constraints. Here the task is to find (the n best) e so that $M(e, d)$ is maximal / minimal. Language constraints, as discussed above, can be parametric and/or subsumption constraints.

It is rarely the case that an inductive query consists of a single constraint. Most often, at least one syntactic and at least one evaluation or optimization constraint would be a part of the query. For example, we might look for the equations, where the LHS is sub-polynomial of X^2Y^3 and $X + Z$ is a sub-polynomial of the RHS, which have the highest multiple correlation coefficient.

3 Heuristic Search for Polynomial Equations

Using constraints in the discovery of polynomial equations helps the user focus on interesting hypotheses and can drastically reduce the space of possible equations. However, in realistic problems with many variables, these reductions are not enough to make the exhaustive search in Lagrange [4] feasible. This motivates us to consider heuristic search.

In this section, we will present a heuristic search algorithm that searches through the space of possible polynomial equations. First, a refinement operator will be introduced that orders the space of polynomial equations. Then, the heuristic function is defined that measures the quality of each equation considered during the search. Finally, the search algorithm based on a beam search strategy is presented.

3.1 The Refinement Operator

In order to apply heuristic search methods to the task of discovering polynomial equations, we first have to order the search space of candidate equations. We will define a refinement operator, based on the syntactic constraints above, that orders the space of equations according to their complexity. Starting with the simplest possible equation and iteratively applying the refinement operator, all candidate polynomial equations can be generated.

Assume we measure the complexity of the polynomial equation $v_d = P$ as $len(P)$. The refinement operator increases $len(P)$ by 1, either by adding a new linear term or by adding a variable to an existing term. Given an equation $v_d = \sum_{i=1}^{r} c_i \cdot T_i$, the refinement operator produces equations that increase r by adding a new linear term (at most one for each $v \in V \setminus v_d$): $v_d = \sum_{i=1}^{r} c_i \cdot T_i + c_{r+1} \cdot v$, where $\forall i : v \neq T_i$. It also produces refined equations that increase the degree of an existing term (at most one for each T_j and $v \in V \setminus v_d$): $v_d = \sum_{i=1, i \neq j}^{r} c_i \cdot T_i + T_j \cdot v$, where $\forall i \neq j : T_j \cdot v \neq T_i$. Special care is taken that the newly introduced/modified term is different from all the terms in the current equation.

Note that the refinements of a given polynomial P are superpolynomials of P. They are minimal refinements in the sense that they increase the complexity of P ($len(P)$ by one unit. The branching factor of the refinement operator depends on the number of variables $|V|$ and number of terms in the current equation r. The upper bound of the branching factor is $\mathcal{O}((|V| - 1)(r + 1)) = \mathcal{O}(|V|r)$, since there are at most $|V| - 1$ possible refinements that increase r and at most $(|V| - 1)r$ possible refinements that increase d.

The defined refinement operator is not optimal, in the sense that each polynomial equation can be derived in several different ways, e.g., $z = x + y$ can be derived by adding x then y or vice-vesa by adding y then x. This is due to the commutativity of the addition and multiplication operators. An optimal refinement operator can be easily obtained by taking into account the lexical ordering of the variables in V. Then, only variables and/or terms) with higher lexical rank (than those already in) should be added to the terms and/or equations.

While an optimal refinement operator is desired for complete/exhaustive search (to avoid redundancy), it may prevent the generation of good equations in greedy heuristic search. Suppose the polynomials x and z have low heuristic value, while y has a high heuristic value and $x + y$ is actually the best. Greedy search would choose y and the optimal refinement operator that takes into account lexicographic order would not generate $x + y$.

3.2 The Search Heuristic

Each polynomial equation structure considered during the search contains a number of generic constant parameters (denoted by c_i). In order to evaluate the quality of an equation, the values of these generic constants have to be fitted against training data consisting of the observed values of the variables in V. Since the polynomial equations are linear in the constant parameters, the standard linear regression technique can be used, just as in Lagrange [4].

The quality of the obtained equation is usually evaluated using an evaluation primitive, such as mean squared error (MSE), defined in the previous section. We will use a MDL (minimal description length) based heuristic that combines the degree of fit with the complexity of the equation, defined as $\mathrm{MDL}(v_d = P) = len(P) \log N + N \log \mathrm{MSE}(v_d = P)$, were N is number of training examples. The second term introduces a penalty for the complexity of the equation. Thus, the MDL heuristic function introduces a preference toward simpler equations.

3.3 The Search Algorithm

Our algorithm Ciper (Ciper stands for Constrained Induction of Polynomial Equations for Regression) employs beam search through the space of possible equations. The algorithm takes as input the training data D, i.e, the training examples, each of them containing the measurements of the observed (numerical) system variables, and a designated dependent variable v_d. In addition, a set of language constraints C can be specified. The output of Ciper consists of the b polynomial equations for v_d that satisfy the constraints C and are best wrt the data D according to the MDL heuristic function defined in the previous section.

```
   procedure Ciper($D$, $v_d$, $C$, b)
 1 $E_0$ = simplest polynomial equation ($v_d = c$ )
 2 $E_0$.MDL = FitParameters($E_0$, $D$)
 3 $Q = \{E_0\}$
 4 repeat
 5   $Q_r$ = {refinements of equations in Q
     that satisfy the language constraints $C$}
 6   foreach equation structure $E \in Q_r$ do
 7     $E$.MDL = FitParameters($E$, $D$)
 8   endfor
 9   $Q$ = {best $b$ equations from $Q \cup Q_r$
     according to MDL }
10 until $Q$ unchanged during the last iteration
11 print $Q$
```

Before the search procedure starts, the beam Q is initialized with the simplest possible polynomial equation of the form $v_d = c$. The value of the constant parameter c is fitted against the training data D using linear regression and the MDL heuristic function is calculated (lines 1-3). In each search iteration, the refinements of the equations in the current beam are computed first and checked for consistency with the specified language constraints C. Those that satisfy the constraints are collected in Q_r (line 5).

In case when redundant equations are generated due to the inoptimality of the refinement operator, the duplicate equations are filtered out from the set Q_r. The refinements are computed using the refinement operator defined in Section 3.1. In case when redundant equations are generated due to the inoptimality of the refinement operator, the duplicate equations are filtered out from the set Q_r.

Linear regression is used to fit the constant parameters of the refinements against the training data D, and the MDL heuristic function is calculated for each refinement (lines 6-8). At the end of each search iteration, the best b equa-

Table 1. The regression datasets' characteristics (left four columns). The performance of CIPER in terms of RE, as compared to three other regression approaches: linear regression (LR), model trees (MT), and regression trees (RT) (right four columns).

Dataset	Exs	Vars	NERPV	CIPER	LR	MT	RT
autoprice	159	16	3.24	0.15	0.23	0.15	0.32
baskball	96	5	4.87	0.61	0.67	0.63	0.78
bodyfat	252	15	2.22	0.03	0.03	0.03	0.11
elusage	55	3	7.56	0.18	0.23	0.28	0.44
fruitfly	125	5	3.94	1.01	1.10	1.03	1.01
housing	506	14	1.24	0.19	0.29	0.17	0.28
mbagrade	61	3	6.97	0.78	0.83	0.83	1.05
pollution	60	16	7.06	0.55	0.55	0.42	0.77
pwLinear	200	11	2.68	0.15	0.25	0.11	0.33
quake	2178	4	0.35	1.00	1.00	0.99	1.00
sensory	576	12	1.11	0.89	0.87	0.75	0.85
strike	625	7	1.04	0.95	0.84	0.83	0.87
veteran	137	8	3.66	0.90	0.92	0.88	0.91
vineyard	52	4	7.89	0.29	0.43	0.48	0.73
Average RE				0.55	0.59	0.54	0.67

tions, according to the MDL heuristic function, are kept in the beam (line 10). The search proceeds until the beam remains unchanged during the last iteration.

4 Polynomial Equations on Standard Regression Datasets

We take 14 regression datasets from the UCI repository, listed in the left part of Table 1. In addition to the number of examples (data points) and variables, we also list the reduction of MSE (in percent) necessary to counterbalance an increase in $len(P)$ of 1 in the MDL heuristic (Needed Error Reduction Per Variable - NERPV).

We perform ten-fold cross-validation to evaluate the performance of our approach, measured in terms of RE (where $RE^2 = 1 - R^2$). We compare the performance of our approach to the performance of linear regression (with no feature selection, no feature elimination), model trees, and regression trees (all of these within WEKA [10]). The results are given in the right part of Table 1.

Note that a beam of 16 was used in CIPER. Our approach performs better than linear regression and regression trees and comparably to model trees. We should note that this is achieved using a single equation/model over the entire instance space, rather than a piece-wise model (as in model trees).

It is interesting to compare the complexity of the equations found by our approach to that of model trees. The complexity of model trees can be measured as the number of parameters/constants in them, including the thresholds in internal nodes and the coefficients in linear regressions in the leaves. The complexity of equations can be measured in several ways ($d + r$, $len(P)$), including the number of coefficients (where we count the degree of a variable in a term as a coefficient). The average total number of parameters in model trees is 16.83

and in regression trees 13.72. The average d, r and $len(P)$ are 1.64, 4.93 and 6.00, respectively. The polynomial equations discovered thus compare favorably in terms of complexity to regression trees and model trees and are about the same level with linear regression (which has on the average 11.93 parameters).

5 Towards Inductive Databases for QSAR

Here we first describe the pattern domain of molecular fragments. We then proceed with a proposal of how to integrate the pattern domains of equations and molecular fragments in order to obtain an inductive database for QSAR (Quantitative Structure-Activity Relationships). Preliminary experiments in the domain of predicting biodegradability, illustrating how the two domains can be combined, are presented.

5.1 The Pattern Domain of Molecular Fragments

The pattern domain of molecular fragments was introduced by Kramer and De Raedt [8]. The system MolFea, which implements a solver for this pattern domain, looks for interesting molecular fragments (features) in sets of molecules. Interestingness is defined as a conjunction of primitive constraint that the fragment has to satisfy.

A molecular fragment is defined as a sequence of linearly connected atoms. For instance, $o-c=o$ is a fragment meaning: "an oxygen atom with a single bond to a carbon atom with a double bond to an oxygen atom". In such expressions 'c', 'n', 'o', etc. denote elements, and '$-$' denotes a single bond, '$=$' a double bond, '$\#$' a triple bond, and '$\sim$' an aromatic bond. Only non-hydrogen atoms are considered. Fragments are partially ordered by the relation "is more general than"/"is a subsequence of": when fragment g is more general than fragment s, one writes $g \leq s$.

The primitive language constraints that can be imposed on unknown target fragments f are of the form $f \leq p$, $p \leq f$, $\neg(f \leq p)$ and $\neg(p \leq f)$, where f is the unknown target fragment and p is a specific pattern. This type of primitive constraint denotes that f should (not) be more specific (general) than the specified fragment p. E.g., the constraint '$c=o$' $\leq f$ specifies that f should be more specific than '$c=o$', i.e., that f should contain '$c=o$' as a subsequence.

The evaluation primitive $freq(f, D)$ denotes the relative frequency of a fragment f on a set of molecules D. The relative frequency is defined as the percentage of molecules (from D) covered (by f). Evaluation constraints can be defined by specifying thresholds on the frequency or relative frequency of a fragment on a dataset: $freq(f, D_1) \leq t$ and $freq(f, D_1) \geq t$ denote that the relative frequency of f on D should be larger than (resp. smaller than) or equal to t. For example, the constraint $freq(f, Pos) \geq 0.95$ denotes that the target fragments f should have a minimum relative frequency of 95% on the set of molecules Pos. The two types of primitive constraints defined above can be conjunctively combined in order to declaratively specify the target fragments of interest.

5.2 Combining Molecular Fragments and Equations

The basic idea of our proposal is to consider the pattern domains of equations and molecular fragments in a single inductive database. One could then easily use the results of one inductive query (e.g., the set of interesting features resulting from a MolFea query) as input to another inductive query (e.g. to find a QSAR equation for biodegradability). This is of interest as QSAR models most often take the form of equations and molecular fragments are often used as features.

Each of the two pattern domains offers potentially useful functionality. On one hand, equations are the approach of choice for QSAR modeling. While non-linear transformations of bulk features are sometimes used, most often linear equations are sought. Our constraint-based approach to equation discovery would allow the QSAR modeler to pose inductive queries that focus the search on interesting equations, such as: "find the best equation that involves featureX", supposing featureX is of interest to the QSAR modeler. On the other hand, molecular fragments (or in general substructures) are often used as features in QSAR modeling. Using feature mining in the pattern domain of molecular fragments, the QSAR modeler can find patterns involving substructures of special interest. These can then be used in QSAR models.

The basic form of exploiting the connection between the two pattern domains is to use the molecular fragments found by MolFea as input for a data mining query looking for QSAR equations. Here one can exploit the subpolynomial constraints in the equations pattern domain to ask for equations that contain a specific molecular fragment: the fragment in question should be a subpolynomial of the RHS of the equation sought. However, additional constraints can be defined. For example, one may look for equations that involve a subfragment or superfragment of a given fragment, rather than just the fragment itself. We have implemented the latter in our system CIPER as an extension of the subpolynomial/superpolynomial constraint.

At present, we assume all frequent fragments of interest are generated beforehand by MolFea and look for equations that involve the given fragments and satisfy the constraints. For example, in the biodegradability dataset, we may be interested in the two best equations that contain only one feature that is a superfragment of '$c = o$'. The equations $logHLT = 6.9693 - 1.19013 * c = o$ and $logHLT = 6.91524 - 1.24286 * c - c = o$ are returned as a result of this query.

Our constraint-based approach to equation discovery is meant for regression problems (continuous class variable). However, we can easily adapt it to classification problems where the features are numeric (including binary features). We can use CIPER to perform classification via regression (multi-response polynomial regression), an approach shown to perform well in many cases.

5.3 Experiments on the Biodegradability Dataset

The QSAR application that we consider here is the one of predicting biodegradability of compounds [3]. The database used was derived from the data in the handbook of degradation rates [6] and contains the chemical structure and measured (or estimated) degradation rates for 328 chemicals. In our study we focus

Table 2. The equation generated by CIPER on the biodegradability dataset.

```
logHLT = 0.0645738*logP - 0.825858*c=o + 0.374477*logP*cl + 0.487245*c-n
       + 2.43385*c~c~c~c~c~c~c~c~c~c~c~c~c~c - 0.529246*c~c~c~c~c
       + 0.757922*n=o + 0.570323*c-c-c~c~c~c~c~c  - 0.632581*c-c-c-o
       + 0.817581*c-o-c - 0.621152*c-o + 0.00231708*MolWeight + 5.94176
```

Table 3. The predictive performance of CIPER on the biodegradability dataset, estimated by 10-fold cross validation, as compared to several other learning algorithms.

Data version / Algorithm	J48	(Class via) LinReg	(Class via) M5	(Class via) CIPER
2 class (acc.)	71.34	74.09	78.05	78.66
4 class (acc.)	56.40	52.24	54.57	56.40
continuous (RE)		0.627	0.560	0.576

on aqueous biodegradation half life time (HLT) in aerobic conditions. The target variable (denoted logHLT) is the natural logarithm of the arithmetic mean of the low and high estimate of the HLT for aqueous biodegradation in aerobic conditions, measured in hours. Two discretized versions (2-class and 4-class) of logHLT were also considered.

The molecular weight of a compound, as well as logP, the logarithm of the compound's octanol/water partition coefficient, were used as features in constructing QSAR models, in addition to the molecular fragments discovered by MolFea. The logP feature is a measure of hydrophobicity and can be expected to be important since we are considering biodegradation in water. The fragments were generated by Kramer and De Raedt [8], who used the two-class version of the biodegradability dataset. They looked for features that are above a prespecified minimum frequency threshold (0.95) for compunds that degrade fast and below a maximum threshold (0.05) for compounds that degrade slow. MolFea found 124 fragments, of which 69 are distinct and were used as features.

Taking the 69 fragments, logP and MolWeight as the independent and logHLT (or the discretization thereof) as the dependent variable, we applied our system for equation discovery CIPER to the biodeg dataset. For the continuous version, the equation in Table 2 was generated by CIPER (with beam 4). For the two-class and four-class version, classification via regression was performed: as many equations were produced as the number of classes.

The predictive performance of CIPER, estimated by 10-fold cross validation, was also measured (Table 3). It was compared to the performance of three other algorithms: J48, Linear Regression and M5. For the latter two, comparison was done for both the continuous (regression) and the discrete (classification via regression) versions of the dataset.

CIPER performs much better than J48 for the two-class version and the same for the four-class version. It performs better than linear regression (higher accuracy, lower error / RE). It performs better than M5 for classification via regression and slightly worse for regression. Overall, CIPER performs best. In addition, the equation produced can be easily interpreted (e.g., three fused aromatic rings greatly contribute to longer degradation time).

6 Summary and Further Work

Here we have considered the problem of predictive modeling via polynomial equations within the framework of inductive databases (IDBs). We have defined primitives for the pattern domain of polynomial equations, including language constraints (such as sub-polynomial), evaluation primitives, evaluation and optimization constraints. We have also shown how to combine such primitives to form inductive queries in this pattern domain and presented a heuristic solver (CIPER) for such queries. We have applied CIPER to standard regression datasets, where it performs competitively to existing regression methods while producing smaller models. We have then indicated how the pattern domains of equations and molecular fragments can be combined into an IDB for QSAR and illustrated the use of such an IDB on the problem of predicting biodegradability.

Many directions for further work exist at present. In the pattern domain of equations, one can consider the definition, implementation and use of similarity constraints: these allow queries of the form: "find the equation most similar to e, which has mean absolute error smaller than x". Also, the pattern domain of equations could be extended towards general equations (non-polynomial ones). Finally, extensions towards other types of regression models (e.g. model trees) as well as classification models would be in order.

Concerning the work towards IDBs for QSAR, we have not even completely and formally specified the language of queries involving both fragments and equations that can be posed to (and solved by) our current implementation: this is an immediate direction for further work. On the technical side, further exploration of the possibilities for integration between fragment and equation discovery deserves attention. Tighter integration seems possible and desirable.

References

1. R. Bayardo. Constraints in data mining. *SIGKDD Explorations*, 4(1), 2002.
2. De Raedt, L. Data mining as constraint logic programming. In *Computational Logic: From Logic Programming into the Future*. Springer, Berlin, 2002.
3. S. Džeroski, H. Blockeel, B. Kompare, S. Kramer, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. In *Proc. Ninth International Conference on Inductive Logic Programming*, pages 80–91. Springer, Berlin, 1999.
4. S. Džeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *J. Intelligent Information Systems*, 4:89–108, 1995.
5. M. Garofalakis and R. Rastogi. Scalable data mining with model constraints. *SIGKDD Explorations*, 2(2):39–48, 2000.
6. Howard, P.H., Boethling, R.S., Jarvis, W.F., Meylan, W.M., and Michalenko, E.M. 1991. *Handbook of Environmental Degradation Rates*. Lewis Publishers.
7. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
8. S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical applications. In *Proc. Eighteenth International Conference on Machine Learning*, pages 258–265. Morgan Kaufmann, San Francisco, 2001.
9. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
10. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, San Mateo, CA, 1999.

From Temporal Rules to Temporal Meta-rules*

Paul Cotofrei and Kilian Stoffel

University of Neuchatel
Pierre-a-Mazel 7, 2000 Neuchatel, Switzerland
{paul.cotofrei,kilian.stoffel}@unine.ch

Abstract. In this article we define a formalism to discover knowledge in the form of *temporal rules*, inferred from databases of *events* with a temporal dimension. The theoretical framework is based on first-order temporal logic and allows the definition of the main temporal data mining notions (event, temporal rule, constraint) in a formal way. The concepts of *consistent linear time structure* and *general interpretation* are fundamentals in the design of algorithms for inferring higher order temporal rules, (called *temporal meta-rules*), from local sets of temporal rules.

1 Introduction

The domain of temporal data mining focuses on the discovery of causal relationships among events that may be ordered in time and may be causally related. The contributions in this domain encompass the discovery of temporal rules, of sequences and of patterns. However, in many respects this is just a terminological heterogeneity among researchers that are, nevertheless, addressing the same problem, albeit from different starting points and domains.

The main tasks concerning the information extraction from time series database and on which the researchers concentrated their efforts over the last years may be divided in several domains. *Similarity/Pattern Querying* concerns the measure of similarity between two sequences or sub-sequences respectively. Different methods were developed, such as window stitching [1] or dynamic time warping based matching [12]. *Clustering/Classification* concentrates on optimal algorithms for clustering/classifying sub-sequences of time series into groups (classes) of similar sub-sequences. Different techniques were proposed, as Hidden Markov Models (HMM) [13], Dynamic Bayes Networks (DBNs) [7] or Recurrent Neural Networks [8]. *Pattern finding/Prediction* methods regard the search for periodicity patterns (fully or partially periodic) in time series databases. For full periodicity search there is a rich collection of statistic methods, like FFT [14]. For partial periodicity search, different algorithms were developed, which explore properties related to partial periodicity such as the max-subpattern-hit-set property [10] or point-wise periodicity [9]. *Temporal Rules' extraction* approach concentrated to the extraction of explicit rules from time series, like temporal association rules [4] or cyclic association rules [16]. Adaptive methods

* Supported by the Swiss National Science Foundation (grant N ° 2100-063 730).

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 169–178, 2004.

for finding rules whose conditions refer to patterns in time series were described in [6,11] and a general methodology for classification and extraction of temporal rules was proposed in [5].

Although there is a rich bibliography concerning formalism for temporal databases, there are very few articles on this topic for temporal data mining. In [2,3,15] general frameworks for temporal mining are proposed, but usually the researches on causal and temporal rules are more concentrated on the methodological/algorithmic aspect, and less on the theoretical aspect. In this article, we provide an innovative formalism based on first-order temporal logic, which permits an abstract view on temporal rules. The formalism allows also the application of an inference phase in which higher order temporal rules (called temporal meta-rules) are inferred from local temporal rules, the lasts being extracted from different sequences of data. Using this strategy, known in the literature as higher order mining [18], we can guarantee the scalability of our system (the capacity to handle huge databases), by applying statistical and machine learning tools.

The rest of the paper is structured as follows. In the next section, the first-order temporal logic formalism is extensively presented (definitions of the main terms – *event, temporal rules, confidence* – and concepts – *consistent linear time structure, general interpretation*). Section 3 contains a brief description of a general methodology for temporal rules extraction, through the frame of the proposed formalism. The notion of temporal meta-rules and the algorithms for inferring such higher order rules are described in Section 4. Finally, the last section summarizes our work and lists some possible future directions.

2 Formalism of Temporal Rules

Time is ubiquitous in information systems, but the mode of representation/perception varies in function of the purpose of the analysis. It can be based either on *time points* (instants) or on *intervals* (periods), may have a *discrete* or a *continuous* structure and may be *linear* or *nonlinear* (e.g. acyclic graph). For our methodology, we chose a temporal domain represented by linearly ordered discrete instants.

Databases being first-order structures, the first-order logic represents a natural formalism for their description. Consequently, the first-order temporal logic expresses the formalism of temporal databases. For the purposes of our methodology we consider a restricted first-order temporal language L which contains only n-ary function symbols ($n \geq 0$), n-ary predicate symbols ($n > 1$, so no proposition symbols), the set of relational symbols $\{=, <, \leq, >, \geq\}$, a single logical connective $\{\wedge\}$ and a temporal connective of the form X_k, $k \in \mathbf{Z}$, where k strictly positive means *next k times*, k strictly negative means *last k times* and $k = 0$ means *now.*

The syntax of L defines terms, atomic formulae and compound formulae. A Horn clause is a formula of the form $B_1 \wedge \cdots \wedge B_m \rightarrow B_{m+1}$ where each B_i is a positive (non-negated) atom. The atoms B_i, $i = 1, \ldots, m$ are called implication clauses, whereas B_{m+1} is known as the implicated clause. Syntactically, we cannot express Horn clauses in our language L because the logical connective $\rightarrow$ is not defined. However, to allow the description of rules, which formally look like a Horn clause, we introduce a new logical connective, $\mapsto$, which practically will represent a rewrite of the connective $\wedge$. Therefore, a formula in L of the form $p \mapsto q$ is syntactically equivalent with the formula $p \wedge q$. When and under what conditions we may use the new connective, is explained in the next definitions.

Definition 1 *An event (or temporal atom) is an atom formed by the predicate symbol E followed by a bracketed n-tuple of terms* $(n \geq 1)$ $E(t_1, t_2, \ldots, t_n)$*. The first term of the tuple,* t_1*, is a constant representing the name of the event and all others terms are function symbols. A short temporal atom (or the event's head) is the atom* $E(t_1)$*.*

Definition 2 *A constraint formula for the event* $E(t_1, t_2, \ldots t_n)$ *is a conjunctive compound formula,* $C_1 \wedge C_2 \wedge \cdots \wedge C_k$*, where each* C_j *is a relation implying one of the terms* t_i*.*

For a short temporal atom $E(t_1)$, the only constraint formula that is permitted is $t_1 = c$, where c is a constant. We denote such a constraint formula as *short constraint formula.*

Definition 3 *A temporal rule is a formula of the form* $H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$*, where* H_{m+1} *is a short constraint formula and* H_i *are constraint formulae, prefixed by the temporal connectives* X_{-k}*,* $k \geq 0$*. The maximum value of the index* k *is called the time window of the temporal rule.*

Remark. The reason for which we did not permit the expression of the implication connective in our language is related to the truth table for a formula $p \rightarrow q$: even if p is false, the formula is still true, which is unacceptable for a temporal rationing of the form *cause*$\rightarrow$ *effect.*

Practically, the only atoms constructed in L are temporal atoms and the only formulae constructed in L are constraint formulae and temporal rules. As a consequence of the definition 3, a conjunction of relations $C_1 \wedge C_2 \wedge \cdots \wedge C_n$, each relation prefixed by temporal connectives X_{-k}, $k \geq 0$, may be rewritten as $C_{\sigma(1)} \wedge \cdots \wedge C_{\sigma(n-1)} \mapsto C_{\sigma(n)}$, – σ being a permutation of $\{1..n\}$ – if and only if there is a short constraint formula $C_{\sigma(n)}$ prefixed by X_0.

The semantics of L is provided by an interpretation I over a domain D. The interpretation assigns an appropriate meaning over D to the (non-logical) symbols of L. Usually, the domain D is imposed during the discretisation phase, which is a pre-processing phase used in almost all knowledge extraction methodologies. Based on Definition 1, an event can be seen as a labelled (constant symbol t_1) sequence of points extracted from raw data and characterized by a finite set of features (function symbols $t_2, \cdots, t_n$). Consequently, the domain D is the union $D_e \cup D_f$, where the set D_e contains all the strings used as event names and the set D_f represents the union of all domains corresponding to chosen features.

To define a first-order linear temporal logic based on L, we need a structure having a temporal dimension and capable to capture the relationship between a time moment and the interpretation I at this moment.

Definition 4 *Given L and a domain D, a (first order) linear time structure is a triple $M = (S, x, \Im)$, where S is a set of states, $x : \mathbf{N} \rightarrow S$ is an infinite sequence of states $(s_0, s_1, \ldots, s_n, \ldots)$ and $\Im$ is a function that associates to each state s an interpretation $\Im(s)$ of all symbols defined at s.*

In the framework of temporal data mining, the function $\Im$ is a constant and it is equal to the interpretation I. In fact, the meaning of the events, constraint formulae and temporal rules is not changing over time. What is changing over time is the value of the meaning. Given a first order time structure M, we denote the instant i (or equivalently, the state s_i) for which $I(P) = true$ by $i \Rightarrow P$, i.e. at time instant i the formula P is true. Therefore, $i \Rightarrow E(t_1, \ldots, t_n)$ means that at time i an event with the name t_1 and characterized by the global features $t_2, \ldots, t_n$ started. A constraint formula is true at time i if and only if all relations are true at time i. A temporal rule is true at time i if and only if $i \Rightarrow H_{m+1}$ and $i \Rightarrow (H_1 \wedge \cdots \wedge H_m)$. (*Remark*: $i \Rightarrow P \wedge Q$ if and only if $i \Rightarrow P$ and $i \Rightarrow Q$; $i \Rightarrow X_k P$ if and only if $i + k \Rightarrow P$).

Now suppose that the following assumptions are true:

A. For each formula P in L, there is an algorithm that calculates the value of the interpretation $I(P)$ in a finite number of steps.
B. There are states (called incomplete states) that do not contain enough information to calculate the interpretation for all formulae defined at these states.
C. It is possible to establish a measure, (called *general interpretation*) about the degree of truth of a compound formula along the entire sequence of states $(s_0, s_1, \ldots, s_n, \ldots)$.

The first assumption expresses the calculability of the interpretation I. The second assumption expresses the situation when only the body of a temporal rule can be evaluated at time moment i, but not the head of the rule. Therefore, for the state s_i, we cannot calculate the interpretation of the temporal rule and the only solution is to estimate it using a general interpretation. This solution is expressed by the third assumption. (*Remark:* The second assumption violates the condition about the existence of an interpretation in each state s_i, from definition 4. But it is well known that in data mining sometimes data is incomplete or is missing. Therefore, we must modify this condition as "*$\Im$ is a function that associates to* `almost` *each state s an interpretation $\Im(s)$ of all symbols defined at s*").

However, to ensure that this general interpretation is well defined, the linear time structure must present some property of consistency. Practically, this means that if we take any sufficiently large subset of time instants, the conclusions we may infer from this subset are sufficiently close from those inferred from the entire set of time instants. Therefore,

Definition 5 *Given L and a linear time structure M, we say that M is a consistent time structure for L if, for every n-ary predicate symbol P, the limit $co(P) = \lim_{n\to\infty} \frac{\#A}{n}$ exists, where $A = \{i \in \{0, \ldots, n\} | i \Rightarrow P\}$ and # means "cardinality". The notation co(P) denotes the confidence of P*

Now we define the general interpretation for an n-ary predicate symbol P as:

Definition 6 *Given L and a consistent linear time structure M for L, the general interpretation I_G for an n-ary predicate P is a function $D^n \rightarrow true \times [0,1]$, $I_G(P) = (true, co(P))$.*

The general interpretation is naturally extended to constraint formulae, prefixed or not by temporal connectives. There is only one exception: for temporal rules the confidence is calculated as the limit of the ratio between the number of certain applications (time instants where both the body and the head of the rule are true) and the number of potential applications (time instants where only the body of the rule is true). The reason for this choice is related to the presence of incomplete states, where the interpretation for the implicated clause cannot be calculated.

Definition 7 *The confidence of a temporal rule $H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$ is the limit $\lim_{n \to \infty} \frac{\#A}{\#B}$, where $A = \{i \in \{0,\ldots,n\} | i \Rightarrow H_1 \wedge \cdots \wedge H_m \wedge H_{m+1}\}$ and $B = \{i \in \{0,\ldots,n\} | i \Rightarrow H_1 \wedge \cdots \wedge H_m\}$.*

For different reasons, (the user has not access to the entire sequence of states, or the states he has access to are incomplete), the general interpretation cannot be calculated. A solution is to estimate I_G using a finite linear time structure, i.e. a model.

Definition 8 *Given L and a consistent time structure $M = (S, x, \Im)$, a model for M is a structure $\tilde{M} = (\tilde{T}, \tilde{x})$ where $\tilde{T}$ is a finite temporal domain $\{i_1, \ldots, i_n\}$, $\tilde{x}$ is the subsequence of states $\{x_{i_1}, \ldots, x_{i_n}\}$ (the restriction of x to the temporal domain $\tilde{T}$) and for each $i_j, j = 1, \ldots, n$, the state x_{i_j} is a complete state.*

Now we may define the estimation for a general interpretation:

Definition 9 *Given L and a model $\tilde{M}$ for M, an estimator of the general interpretation for an n-ary predicate P, $\tilde{I}_G(P)$, is a function $D^n \rightarrow true \times [0,1]$, assigning to P the value* true *with a confidence equal to the ratio $\frac{\#A}{\#\tilde{T}}$, where $A = \{i \in \tilde{T} | i \Rightarrow P\}$. The notation $co(P, \tilde{M})$ will denote the estimated confidence of P, given $\tilde{M}$.*

The estimation of the confidence of a temporal rule, giving a model, is defined as:

Definition 10 *Given a model $\tilde{M} = (\tilde{T}, \tilde{x})$ for M, the estimation of the confidence of the temporal rule $H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$ is the ratio $\frac{\#A}{\#B}$, where $A = \{i \in \tilde{T} | i \Rightarrow H_1 \wedge \cdots \wedge H_m \wedge H_{m+1}\}$ and $B = \{i \in \tilde{T} | i \Rightarrow H_1 \wedge \cdots \wedge H_m\}$.*

Table 1. The first six states of the linear time structure M (example)

State	Events	State	Events	State	Events
s_1	$E(peak, 10, 1.5)$	s_3	$E(plateau, 3, 0.2)$	s_5	$E(valley, 15, 1.9)$
s_2	$E(peak, 12, 1.2)$	s_4	$E(plateau, 1, 0.5)$	s_6	$E(peak, 6, 1.1)$

Table 2. Two temporal rules extracted from two models $\tilde{M}$ using the induction process

Model	Temporal Rule
$s_1 \ldots s_{100}$	$X_{-4}(t_1 = peak) \wedge X_{-4}(t_2 < 11) \wedge X_{-3}(t_1 = peak) \mapsto X_0(t_1 = valley)$
$s_{300} \ldots s_{399}$	$X_{-2}(t_1 = peak) \wedge X_{-2}(t_3 < 1.1) \wedge X_{-1}(t_1 = plateau) \wedge \mapsto X_0(t_1 = valley)$

3 A General Methodology

A general methodology for temporal rules extraction may be structured in two phases. The first, called discretisation phase, transforms sequential raw data into sequences of events. Practically, this means to establish the set of events, identified by names (constant symbol t_1), and the set of features, common for all events (function symbols $t_2, \ldots, t_n$). As example, consider a database containing the daily price variations of a given stock. After the application of the first phase we obtain an ordered sequence of events. Each event has the form $(name, v_1,\ v_2)$, where the name is one of the strings {*peak, plateau, valley*} and $v_1,\ v_2$ represents the maximum variation, respectively, the standard error of the daily prices.

In the frame of our formalism, during this phase, we establish the set of temporal atoms which can be defined syntactically in L. In the above example, an event is defined as $E(t_1, t_2, t_3)$. Semantically, the domain D_e is defined as {*peak, plateau, valley*} and the domain D_f as $\Re^+$ (the prices are positives real numbers and the features are statistical functions). Furthermore, during this phase, a linear time structure M is defined: at each time moment i, the state s_i contains as information the set of events started at i (see Table 1).

The second phase, called inference phase, extracts temporal rules from the ordered sequence of events. This phase is divided in two steps. The first step consists in applying an induction process on a model $\tilde{M}$ to obtain temporal rules. This step can be repeated for different models $\tilde{M}_i$ (see Table 2). Different approaches – Association Rules, Inductive Logic Programming, Classification Trees [5] – can be applied to extract rules from a database of events.

The second step consists in applying an inference process, using the previously inferred temporal rules, to obtain the final set of temporal meta-rules. These rules are temporal rules in accordance with the Definition 3, but supposed to have a small variability of the estimated confidence among different models. Therefore, such a rule may be applied with the same confidence in any state, complete or incomplete. The process of inferring temporal meta-rules is related to a new approach in data mining, called higher order mining, i.e. mining from the results of previous mining results. The formalism we proposed does not impose what approach must be use to discover first order temporal rules (rules generated by the induction process). As long as these rules may be expressed according to the Definition 3, the strategy (including algorithms, criterions, statistical methods) developed to infer temporal meta-rules remains valid.

4 Temporal Meta-rules

Suppose that we dispose of a set of temporal rules corresponding to a given model $\tilde{M}$. It is very likely that some temporal rules contain constraint formulae that are irrelevant, i.e. by deleting these relations, the general interpretation of the rules remain unchanged. In the frame of a consistent time structure M, it is obvious that we cannot delete a relation from a temporal rule (noted TR) if the resulting temporal rule (noted TR^-) has a lower confidence. But for a given model $\tilde{M}$, we calculate an estimate of $co(TR)$, which is $co(TR, \tilde{M})$. Because this estimator has a binomial distribution, we may calculate a confidence interval for $co(TR)$ and, consequently, we accept to delete a relation from TR if and only if the lower confidence limit of $co(TR^-, \tilde{M})$ is greater than the lower confidence limit of $co(TR, \tilde{M})$.

The estimator $co(TR, \tilde{M})$ being a ratio, $\#A/\#B$, a confidence interval for this value is constructed using a normal distribution with mean $\pi = \#A/\#B$ and variance $\sigma^2 = \pi(1-\pi)/\#B)$. The lower limit of the interval is $L_\alpha(A, B) = \pi - z_\alpha\sigma$, where z_α is the quantile of the standard normal distribution for a given confidence level α (usually, 0.95). The following algorithm generalizes a single temporal rule TR, by deleting one relation.

Algorithm 1 ***Generalization 1-delete***

```
Step 1 Let $TR = H_1 \wedge \cdots \wedge H_m \mapsto H_{m+1}$. Let $\aleph = \cup C_j$, where $C_j$ are all relations
    that appear in the constraint formulae of the implication clauses. Rewrite TR, by
    an abuse of notation, as $\aleph \mapsto H_{m+1}$. If $n = \#\aleph$, denote by $C_1, \ldots, C_n$ the list of
    all relations from $\aleph$.
Step 2 For each $i = 1, \ldots, n$ do
    $\aleph^- = \aleph - C_i, \quad TR_i^- = \aleph^- \mapsto H_{m+1}$
    $A = \{i \in \tilde{T} | i \Rightarrow \aleph \wedge H_{m+1}\}, \quad B = \{i \in \tilde{T} | i \Rightarrow \aleph\}$
    $A^- = \{i \in \tilde{T} | i \Rightarrow \aleph^- \wedge H_{m+1}\}, \quad B^- = \{i \in \tilde{T} | i \Rightarrow \aleph^-\}$
    $co(TR, \tilde{M}) = \#A/\#B, \quad co(TR_i^-, \tilde{M}) = \#A^-/\#B^-$
    If $L_\alpha(A, B) \le L_\alpha(A^-, B^-)$ then store $TR_i^-$
Step 3 Keep the generalized temporal rule $TR_i^-$ for which $L_\alpha(A^-, B^-)$ is maximal.
```

The complexity of this algorithm is linear in n. Using the criterion of lower confidence limit, (or LCL), we define the temporal meta-rule inferred from TR as the temporal rule with a maximum set of relations deleted from $\aleph$ and having the maximum lower confidence limit greater than $L_\alpha(A, B)$. An algorithm designed to find the largest subset of relations that can be deleted will have an exponential complexity. A solution is to use Algorithm 1 in successive steps until no more deletion is possible, but without having the guarantee to get the global maximum.

As example, consider the first temporal rule of Table 1 and suppose that $\#A = 20$ and $\#B = 40$ (giving an estimate $co(TR, \tilde{M}) = 0.5$, with $L_{0.95}(0.5) =$

Table 3. Parameters calculated in Step 2 of Algorithm 1 by deleting one relation

Deleted relation	$\#A^-$	$\#B^-$	$co(TR_i^-, \tilde{M})$	$L_\alpha(A^-, B^-)$
$X_{-4}(t_1 = peak)$	24	49	0.489	0.349
$X_{-4}(t_2 < 11)$	30	50	0.60	0.464
$X_{-3}(t_1 = peak)$	22	48	0.458	0.317

0.345). Looking at Table 3, we find two relations which could be deleted (the first and the second) with a maximum $L_\alpha(A^-, B^-)$ given by the second relation.

Suppose now that we dispose of two models, $\tilde{M}_1 = (\tilde{T}_1, \tilde{x}_1)$ and $\tilde{M}_2 = (\tilde{T}_2, \tilde{x}_2)$, and for each model we have a set of temporal rules with the same implicated clause H (sets denoted S_1, respectively S_2). Let S be a subset of the union $S_1 \cup S_2$. If $TR_j \in S$, $j = 1, \ldots, n$, $TR_j = H_1 \wedge \cdots \wedge H_{m_j} \mapsto H$, then consider the sets $A_j = \{i \in \tilde{T}_1 \cup \tilde{T}_2 | i \Rightarrow H_1 \wedge \ldots \wedge H_{m_j} \wedge H\}$, $\mathbf{A} = \cup A_j$, $B_j = \{i \in \tilde{T}_1 \cup \tilde{T}_2 | i \Rightarrow H_1 \wedge \ldots \wedge H_{m_j}\}$, $\mathbf{B} = \cup B_j$ and $\mathbf{C} = \{i \in \tilde{T}_1 \cup \tilde{T}_2 | i \Rightarrow H\}$. The performance of the subset S can be summarized by the number of *false positives* (time instants where the implication clauses of each temporal rule from S are true, but not the clause H) and the number of *false negatives* (time instants where the clause H is true, but not at least one of the implication clauses of the temporal rules from S). Practically, the number of *false positives* is $fp = \#(\mathbf{B} - \mathbf{A})$ and the number of *false negatives* is $fn = \#(\mathbf{C} - \mathbf{B})$. The worth of the subset S of temporal rules is assessed using the Minimum Description Length Principle (MDLP)[17]. This provides a basis for offsetting the accuracy of a theory (here, a subset of temporal rules) against its complexity. The principle is simple: a Sender and a Receiver have both the same models $\tilde{M}_1$ and $\tilde{M}_2$, but the states of the models of the Receiver are incomplete states (the interpretation of the implicated clause cannot be calculated). The sender must communicate the missing information to the Receiver by transmitting a theory together with the exceptions to this theory. He may choose either a simple theory with a great number of exceptions or a complex theory with fewer exceptions. The MLD Principle states that the best theory will minimize the number of bits required to encode the total message consisting of the theory together with its associated exceptions.

To encode a temporal rule from S, we must specify its implication clauses (the clause H is the same for all rules, so is no need to encoded it). Because the order of the implication clauses is not important, the number of required bits is reduced by $\kappa \log_2(m!)$, where m is the number of implication clauses and κ is a constant depending on the encoding procedure. The number of bits required to encode the set S is the sum of encoding length for each temporal rule from S reduced by $\kappa \log_2(n!)$ (the order of the n temporal rules from S is not important). The exceptions are encoded by indicating the sets *false positive* and *false negative.* If $b = \#\mathbf{B}$ and $N = \#(\tilde{T}_1 + \tilde{T}_2)$ then the number of bits required is $\kappa \log_2 \left(\binom{b}{fp} \right) + \kappa \log_2 \left(\binom{N-b}{fn} \right)$, because we have $\binom{b}{fp}$ possibilities to choose the *false positives* among the cases covered by the rules and $\binom{N-b}{fn}$ possibilities to indicate the *false negatives* among the uncovered cases. The total number of bits required to encode the message is then equal to *theory bits + exceptions bits.*

Using the criterion of MDLP, we define as temporal meta-rules inferred from a set of temporal rules (implying the same clause and extracted from at least two different models), the subset S that minimizes the total encoding length. The algorithm that find this subset S has again an exponential complexity, but in practice we may use different non-optimal strategies (hill-climbing, genetic algorithms, simulated annealing), having a polynomial complexity.

Because the two definitions of temporal meta-rules differ not only in criterion (LCL, respectively MLDP), but also in the number of initial models (one, respectively at least two), the second inference process is applied in two steps. During the first step, temporal meta-rules are inferred from each set of temporal rules based on a single model. During the second step, temporal meta-rules are inferred from each set of temporal rules created during the step one and having the same implicated clause. There is another reason to apply firstly the LCL criterion: the resulted temporal meta-rules are less redundant concerning the set of implication clauses and so the encoding procedures, used by MLDP criterion, don't need an adjustment against this effect.

5 Conclusions

In this article we constructed a theoretical framework to discover knowledge, represented in the form of general Horn clauses, inferred from databases with a temporal dimension. This framework, based on first-order temporal logic, permits to define the main notions (event, temporal rule, constraint) in a formal way. The concept of consistent linear time structure allows us to introduce the notions of *general interpretation* and of *confidence*.

Also included in the proposed framework, the process of inferring temporal meta-rules is related to a new approach in data mining, called *higher order mining*, i.e. mining from the results of previous mining runs. According to this approach, the rules generated by the first induction process are first order rules and those generated by the second inference process (i.e temporal meta-rules) are higher order rules. Our formalism does not impose which methodology must be used to discover first order rules. As long as these rules may be expressed according to Definition 3, the strategy (here including algorithms, criterions, statistical methods), developed to infer temporal meta-rules may be applied.

It is important to mention that the condition of the existence of the limit, in the definition of a consistent linear time structure, is a fundamental one: it expresses the fact that the structure M represents a homogenous model and therefore the conclusions (or inferences) based on a finite model $\tilde{M}$ for M are consistent. However, at this moment, we do not know any methods which may certify that a given model is consistent. In our opinion, the only feasible approach to this problem is the development of methods and procedures for detecting the change points in the model. In this sense, the analysis of the evolution of temporal meta-rules seems to ba a very promising starting point.

References

1. R. Agrawal, K. I. Lin, H. S. Sawhney, and K. Smith. Fast Similarity Search in the Presence of Noise, Scaling and Transaction in Time-Series Databases. In *Proc. of the* 21^{th} *Conf. on VLDB*, pages 490–501, 1995.
2. S. Al-Naemi. A Theoretical Framework for Temporal Knowledge Discovery. In *Proc. of Int. Workshop on Spatio-Temporal Databases*, pages 23–33, Spain, 1994.
3. X. Chen and I. Petrounias. A Framework for Temporal Data Mining. In *Proc. of the* 9^{th} *Int. Conf. DESA*, pages 796–805, L.N.C.S. 1460, 1998.
4. X. Chen and I. Petrounias. Discovering Temporal Association Rules: Algorithms, Language and System. In *Proc. of the* 6^{th} *ICDE*, page 306, San Diego, USA, 2000.
5. P. Cotofrei and K. Stoffel. Classification Rules + Time = Temporal Rules. In *Proc. of ICCS*, pages 572–581. Lecture Notes in Computer Science, 2329, 2002.
6. G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule Discovery from Time Series. In *Proc. of the* 4^{th} *Conf. KDD*, pages 16–22, 1998.
7. N. Friedman, K. Murphy, and S. Russel. Learning the structure of dynamic probabilistic networks. In *Proc. of the* 14^{th} *Conf. on Uncertainty in AI*, pages 139–147. AAAI Press, 1998.
8. G. Guimares. Temporal Knowledge Discovery for multivariate time series with enhanced self-organizing maps. In *Proceedings of the IEEE-INNS-ENNS Int. Joint Conference on Neural Networks*, pages 165–170. IEEE Computer Society, 2000.
9. J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *Proc. of ICDE*, pages 106–115, Sydeny, Australia, 1999.
10. J. Han, W. Gong, and Y. Yin. Mining Segment-Wise Periodic Patterns in Time-Related Databases. In *Proc. of the* 4^{th} *Conf. KDD*, pages 214–218, 1998.
11. F. Hoppner. Learning Temporal Rules from State Sequences. In *IJCAI Workshop on Learning from Temporal and Spatial Data*, pages 25–31, Seattle, USA, 2001.
12. E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. Iterative Deepening Dynamic Time Warping for Time Series. In *Proc. of* 2^{th} *SIAM Conf. on Data Mining*, 2002.
13. W. Lin, M. A. Orgun, and G. J. Williams. Temporal Data Mining using Hidden Markov-Local Polynomial Models. In *LNCS*, volume 2035, pages 324–335, 2001.
14. H. Loether and D. McTavish. *Descriptive and Inferential Statistics: An introduction.* Allyn and Bacon, 1993.
15. D. Malerba, F. Esposito, and F. Lisi. A Logical Framework for Frequent Pattern Discovery in Spatial Data. In *Proc. of* 5^{th} *Conf. KDD*, pages 53–62, 2001.
16. B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proc. Of Int. Conf. on Data Engineering*, pages 412–421, Orlando, USA, 1998.
17. J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.
18. M. Spiliopoulou and J. Roddick. Higher Order Mining: Modeling and Mining the Results of Knowledge Discovery. In *Proc. Of the* 2^{nd} *Int. Conf. on Data Mining Methods and Databases*, pages 309–320, UK, 2000.

How Is BI Used in Industry?: Report from a Knowledge Exchange Network

Torben Bach Pedersen

Department of Computer Science, Aalborg University
Fr. Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark
tbp@cs.aau.dk

Abstract. After being born in industry, the topic of *business intelligence* (BI), has become an active research topic, spanning research areas such as data warehousing, On-Line Analytical Processing (OLAP), data mining, and data streams. Ideally, BI research should be driven by industry needs. However, only little information on how BI is used in industry is currently available to the BI research community.

This industrial paper presents the experiences that the author has obtained from running a number of knowledge exchange activities related to BI, e.g., network meetings and workshops, with partners in industry. The paper presents the network participants, the topics of the knowledge exchange activities, and the lessons learned over the last couple of years. For example, lessons relate to the popularity of OLAP and data mining tools, the time spent on different tasks in BI projects, and the importance of "soft" people-related issues. Finally, the paper makes a recommendation on a number of future research topics that are useful in industry.

1 Introduction

The topic of *business intelligence* (BI) was born in industry, but has now become an active research topic, spanning research areas such as data warehousing, On-Line Analytical Processing (OLAP), data mining, and data streams. Ideally, BI research should be driven by industrial needs. However, only little information on how BI is used in industry is currently available to the BI research community.

This industrial paper presents the most significant experiences that the author has obtained from running a number of BI-related knowledge exchange activities with a set of significant industrial BI users. The activities are a knowledge exchange network on Business Intelligence Technology (BIT) [8] and a workshop on BI issues. The paper presents the network participants, the topics of the knowledge exchange activities, and the lessons learned over the last couple of years. For example, lessons relate to the (relatively rare) use of OLAP and data mining tools, the time spent on different tasks in BI projects, legacy compatibility, and the importance of "soft" people-related issues. Finally, the paper used the presented to lessons to make a recommendation on a number of future research topics that will be useful in industry.

This paper complements previous work such as [12] by providing concrete details about experiences with the important BI issues as perceived by a number of representative companies, and presenting a number of interesting novel lessons.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 179–188, 2004.

A secondary aim of the paper is to inspire other researchers to start similar activities, as the task of knowledge transfer to industry is increasingly becoming a standard task for university researchers.

The remainder of the paper is structured as follows. Section 2 describes the knowledge exchange setup and the BIT network. Section 3 describes the topics covered in the activities, while Section 4 summarizes the lessons learned. Section 5 proposes a set of research topics with particular industrial relevance, while Section 6 concludes and points to future work.

2 The Knowledge Exchange Setup

In this section, the organization of the knowledge exchange network will briefly be described. The background is the increased focus in many countries on transferring the knowledge available in universities to industry. This is currently a focus area of the Danish government, and the new law governing universities stress knowledge exchange as the "third leg" of the universities, along with teaching and research. Thus, the purpose of this section is to serve as an inspiration for other researchers who wish to organize similar activities.

2.1 Nouhauz

In order to provide a forum for knowledge exchange, the Department of Computer Science at Aalborg University has founded *Nouhauz* [6] whose name suggests that it is a virtual organization (no house), and is about know-how. Nouhauz is a new forum for discussion, co-operation and contact between IT researchers, IT students, and IT employees in the Danish region of Northern Jutland. The point of departure is a wish formulated by Department of Computer Science at Aalborg University to go into a closer clinch with the IT industry in Northern Jutland. The goal is for the industry to be an incentive to research, just as research and co-operation with students can stimulate business.

This wish has been tried put into practice through Innovation Centre Nouhauz. The idea is that Nouhauz's activities already exist. In this way, Nouhauz is NOT a construction the contents of which must first be created, but a frame around an already existing and well-tested content including research projects, student projects, educational possibilities within industry, applied courses, Nouhauz research, and networks.

Several businesses do not know about this palette of activities emanating from the Department. Nouhauz opens the possibility of gaining such knowledge, partly through the homepage, www.nouhauz.dk, partly through a number of IT activities, and finally through specific possibilities of co-operation with researchers and students.

The strategy is based on: networks across walls; shared ownership (participants decide on content and direction); minimalist and non-profit organization (the fewest possible rules and criteria); oriented towards activities; The target group is IT researchers, IT students, and IT employees.

2.2 The BIT Network

We now describe our particular network on *Business Intelligence Technology* (BIT) [8].

The idea of a knowledge exchange network is to bring a small (less than 15 participants) and focused group of people together at regular intervals to discuss topics that are relevant for them. The small size and relative "closedness" of the network means that the participants get to know each other quickly. Also, the participants are chosen so that there are no immediate competitor or vendor-customer relationships among them. This has ensured that the participants speak very freely about their problems and solutions, and ensures a good discussion in the network.

The network meets around six times a year, mostly in Spring and Fall. A small fee of a few hundred euros per year is charged for each individual person participating in the network. This is done to pay for the associated costs and, more importantly, to ensure that only persons with a serious interest join the network.

The network was kicked off with a startup meeting where a wide range of potential participants had been invited. The kick-off meeting contained presentations on BI research at the Department of Computer Science at Aalborg University, as well as short presentations from the potential participants about their use of BI. The most important decision made was *not* to allow vendors of BI tools (of which several had shown interest) to participate, as several BI user companies felt that this would destroy the desired atmosphere of mutual trust.

2.3 Network Members

We now proceed with a short description of each company/department participating in the network. The aim of this section is to give an impression of the diversity of the participants, and thus, in turn, of how representative they are for the general use of BI in industry. The individual network members are described in alphabetical order below.

BusinessMinds. BusinessMinds [1] is a Danish consultancy company specialized in data warehousing and business intelligence. They focus on delivering solutions based on an enterprise data warehousing architecture to ensure future growth possibilities. With around 10 employees, BusinessMinds are (size-wise) the number two specialized BI consulting company in Denmark after Platon, another network participant.

Database Group, Department of Computer Science, Aalborg University. The main academic partner in the network is the Department of Computer Science at Aalborg University which has more than 100 employees and is one of the major CS departments in Denmark. More specifically, the network is run by the author who is a member of the Database research group which has BI as one of its major research topics. The author always participates in the meetings, but often 1–3 other researchers will also participate, depending on the topic.

Department of Production, AAU. The Department of Production is a department under the Faculty of Engineering and Science at Aalborg University. The Department's teaching and research activities are mainly related to the establishment, development, and running of service and industrial companies both in Denmark and in other countries. The participants in the network mainly work with how to use IT to optimize the production process. Thus, from a BI perspective, they are BI *users* rather than BI researchers, but their academic background gives them a different angle than the industrial participants.

KMD. With approximately 2,500 employees, KMD is the largest "pure" software company in Denmark, and one of Denmark's major information technology providers for the public sector. KMD is the market leader in digital administration and has the local municipalities as its primary customers. Founded in 1974, KMD has a large portfolio of legacy systems, mainly based on mainframe technology, but has more recently expanded to offer systems on a range of platforms.

Platon. Consultancy Founded in 1999 and with 60 employees and at least twice as many customers, offices in Copenhagen, Århus and Oslo, Platon is the largest independent consulting firm in the BI field in Scandinavia, and thus in Denmark. The main focus areas are data warehousing, master data management, financial management, customer relationship management and knowledge management. Platon does not sell software and hardware. The services are focused on consulting, project management, solution implementation, support and education.

Nykredit. Financial Founded in 1851, Nykredit is one of Denmark's leading financial services providers and the largest mortgage provider in Denmark. Nykredit has a combined total outstanding amount of mortgage bonds of EUR 93.3bn at end-2003. Based on the financing of property, Nykredit offers mortgage finance, banking, insurance as well as estate agency services. Nykredit has around 3,200 employees and more than 540,000 customers. The IT-related functions in Nykredit employ more than 300 people. As most financial institutions in Denmark, the system portfolio is mainly based on mainframe technology.

SONOFON. SONOFON was founded in 1991, with a focus on mobile telecommunications, and is today the second largest telco in Denmark with more than a million own mobile subscribers and turnover of more than 600 million euros. SONOFON has around 1500 employees of which several hundred are in IT-related functions. The IT systems are mainly based on Unix and Oracle.

Member Summary. To summarize, the network has participations of the two largest BI consulting companies in Denmark, three heavy IT/BI users in the public, financial, and telecom sectors, as well as two academic departments. Thus, the author believes that the network members are indeed representative for the state-of-the-art of industrial BI use in Denmark, and probably also (at least in) Europe, as Denmark is traditionally among the technologically most advanced countries in Europe. This means that the experiences gained here will most likely be of broad interest.

3 Network Topics

Now, a description of the meeting topics so far, more or less in chronological order, is given. The idea is to give information on which topics industry finds interesting. The topics of the meetings were only chosen 1-2 meetings ahead, to ensure that new interests can be covered. The reason for presenting the topics in chronological order is to give the reader a feeling for the alternation between "soft" organizational/methodological issues and "hard" technical issues that were discussed in the network. This shows the broadness of issues that industrial BI users need to address at the same time, in contrast to BI researchers that typically concentrate on a single technical issue at a time.

Multidimensional Modeling. The first real meeting concerned multidimensional data modeling. A presentation on modeling complex multidimensional data was given by the author. However, only a few participants showed up. Later, multidimensional modeling has not been discussed at all. Thus, it would seem that multidimensional modeling is not a big issue for the industrial users, which is in contrast to the huge amount of work on multidimensional data models done in academia.

BI Workshop. In September 2002, a two-day workshop about BI was organized by the author. Here, a number of leading BI tool vendors was invited to present their products, including Business Objects, Brio, and Oracle, as well as the Danish vendors Acinta and TARGIT. A number of presentations on BI research were also given. The workshop was finished off with a round-table discussion about the most important BI issues. Among the conclusions were that all the market-leading BI client tools can do more or less all that a BI user would ever require them to do. Thus, BI vendor efforts should be focused on other areas, of which data integration was the one mentioned by most.

Using BI Tools. A follow-up meeting to the workshop concerned the use of BI client tools and featured a presentation on the use of Business Objects at KMD. This was interesting to most of the participants, as several were either already using or considering using this tool. An observation which can be made here is that the network participants value such opportunities to exchange practical experiences with other people using the same tools quite highly.

ETL. The next two meetings concerned Extract-Transform-Load (ETL) tools and methods. The ETL method developed and used by Platon was presented, and demos of two ETL tools, Microsoft Data Transformation Services and Informatica Powercenter, were given. As several of the participants were evaluating ETL tools as part of a purchase decision process, the network tried to coordinate the development of an "ETL tool requirements list" which should capture all important aspects. Apart from the expected technical requirements such as transformation functionality and possible data sources, it was surprising to see how much "soft issues" related to the vendors (reference customers in Denmark, Danish support, perceived vendor "stability") meant.

Another important lesson was the prime importance of having tools that had full support for mainframes, in order to ensure legacy compatibility. In fact, mainframe compatibility was the prime factor in deciding which ETL tool to buy for at least one participant. Another conclusion was that (bad) data quality was always an issue, something which was becoming even more evident with the trend towards "near real-time" data warehousing.

BI Documentation and Metadata. The next meeting was about BI metadata in a broad sense, ranging from a presentation on the BI documentation standard at Platon, i.e., unstructured, textual metadata, to a presentation on the OMG Common Warehouse Metamodel (CWM) standard. None of the participants were using structured metadata in a significant way, but there was great interest in the CWM model and people agreed that this was the way to go to ensure compatibility among the many tools used for BI. In fact, most agreed that CWM compatibility should be standard requirement in future BI purchasing scenarios.

BI Requirements: Management and Prioritization. The topic of managing BI requirements was considered to be one of the more problematic issues in the daily life of the participants. Most of them were bogged down by too many requests for new data, functionality, and reports in their BI environment, and several had no standard way of prioritizing requests, e.g., based on business value. Thus, a standard methodology for doing this was highly desired.

BI User Acceptance: Success Criteria and Follow-Up. Another "soft issue" concerned how to measure the success of a BI project, and how to use these measurements to follow up in the organization. Some standard metrics such as the number of distinct users and consumed CPU time were discussed, but the ultimate success criteria was perceived to be what was formulated by one of the BI consultants as "BI success is when the users allow you to turn off their old reporting system and use only the new BI system for all reporting tasks."

Real-Time Data Warehousing and Data Streams. The next two meetings concerned two closely related topics, namely real-time data warehousing and data streams. There was a presentation by Platon on the issues related to the "real-time enterprise", most notably the process known as Business Activity Monitoring (BAM) [2], which concerns monitoring and reacting to business events in near real-time. In industry, the term "real-time" is used a little differently than in the research community, as delays of, e.g., a few minutes, in data propagation to the DW are typical. Other presentations concerned the real-time DW activities of the author [7] and a survey of current research in data streams [3]. These topics were very popular as most of the participants either already had, or were planning, projects about real-time DW, mostly based on Operational Data Store-like technology. However, the attitude towards data streams was that is was highly interesting, but a little to early as no major vendor currently support streams.

BI Challenges in 2003–04. In January 2004, a meeting was held about what the major BI challenges had been in 2003 and would be in 2004. On the technical side, real-time data warehousing and so-called "second generation data warehouses" were mentioned along with implementing procedures for ensuring data quality. However, most of the challenges were related to "soft issues". Optimizing BI development processes, especially the maintenance of BI systems and the set up of testing procedures, was a big issue. Another challenge mentioned by several participants was the integration of new BI staff members, most BI groups were growing rapidly. Finally, a number of major challenges were related to changes in the market for the participants, e.g., the proposed reform of local government in Denmark, and the introduction of the Basel 2 standard in the financial sector.

Topics Summary. As can be seen below, the topics vary from some that are quite research oriented to some that are practice oriented. The whole idea of the network is to have a good mix of research and practice topics to ensure that the researchers and the industry people can mutually inspire and inform each other. Note that the topics alternate between being of a technical nature and a more methodogical and organizational nature. This is a desire of the participants as they often find the technical problems to be easier to handle than the "softer" ones. Another common experience was that using the network as a forum for exchanging practical experiences with concrete tools was highly valued by the industrial participants.

4 Lessons Learned

We will now try to summarize the lessons learned from running the network over the last two years, some of which more take the form of open-ended questions.

Standard Reports Are Enough. A common lesson is that standard reports, containing pre-defined text or graphics layouts and produced at regular intervals, is by far the most used way of interacting with the BI system. Even relatively simple analytical functionality like the one offered by OLAP systems seem to be used at lot less. Even though this is probably related to the "BI maturity" of the organization, the trend will likely remain.

BI Client Tools Can Do Everything. Another common, and quite positive, lesson is that the current BI client tools from the most popular vendors really have all the analysis and reporting functionality that will ever be needed by 90% of the users. However, it is often wise to purchase at choose at least two BI tool sets, one for the average (casual) user, and one or more tool sets aimed at power users.

Converting Old Reports? A common problem seems to be the question of whether existing reports should be converted 1-1 into a new BI system and whether this should be done as the first step when developing a new BI system. There is often a Catch-22 like situation, where a lot of resources will be used on re-creating existing reports before the new BI system can provide any advantages over the old one. However, given the two alternative reports, users tends to always trust the old report more than the new one. Thus, no real success is achieved for the new BI system until the new system can do significantly more than the old one. However, as noted above, the ultimate stamp of approval of the new system is received when the users agree to "turn off" the old reporting system. As it is often seen that the existing reports are not really optimal, it is advisable to assess the value of old reports carefully before converting them.

No Data Mining? When looking at the topics discussed in the network, the topic of data mining is missing completely, which is in sharp contrast to the huge research activity in the field. Among the participants, only very few people do "real" data mining, i.e., something more than just standard aggregate reports. For example, SONOFON has just one employee doing data mining, while Nykredit has a few statisticians that really do traditional statistics rather than data mining. The consulting companies trained quite a lot of consultants in data mining a couple of years ago due to an expected future demand, but so far only very few clients have requested data mining expertise. Again, this issue is probably related to the maturity of BI use in the companies, as a complete DW and basic reporting and analysis must be in place before data mining can be meaningfully used.

Legacy Compatibility Matters. Another lesson that is probably surprising to many researchers is the emphasis most companies place on maintaining compatibility with legacy systems, most notably mainframe-based systems. Both for ETL and BI client tools, issues such as mainframe connectivity and integration were among the primary decision points when purchasing software.

ETL (Still) Takes All the Time. It has long been common wisdom in data warehousing projects that ETL issues take up to 80% of the time spent on developing a DW system. Even though most of the participant companies were using state-of-the-art ETL tools from the leading vendors, it seems that ETL still accounts for by far most of the development time. The value of ETL tools thus lie just as much in making ETL programs more similar, understandable, and better documented, as in increasing ETL programmer productivity.

The Real-Time Enterprise Is Here. An exiting lesson is that the BI topics surrounding the "real-time enterprise," i.e., an enterprise that can both capture relevant data and react to it in (almost) real-time seem to be relevant for most of the participating companies, either now or in the near future. The analysis firm Gartner Group uses the term Business Activity Monitoring (BAM) for this [2].

BI Systems Never Go into Maintenance Mode. For operational (OLTP) systems, most software companies have well established procedures for when and how systems should be transferred from development mode into (long-term) maintenance mode to ensure that developers can be freed for new tasks. The same cycle seem beneficial for BI systems. However, there seems to be a tendency that BI systems are considered as something "new" and "special," and thus stay in development for years without going into maintenance mode, something which is bogging down BI developers and preventing them from dedicating their resources to new projects. This seems to be yet another issue of achieving "maturity" in BI use in companies.

Soft Issues Matter. A more general, but related, lesson is that "soft issues," e.g., organizational ones, seem to be just as important as technical ones, or perhaps even more important. Examples of such issues include how to integrate new BI people and how to optimize BI development processes, such as maintenance or documentation.

5 What Should BI Researchers Be Working on?

Based on the sections above, it seems that three general research areas stand out as being particularly interesting to industry, namely *data integration*, *real-time DW and data streams*, and *"soft BI issues"*.

Data integration covers issues such as ETL, data cleansing, and data quality. Researchers should try to integrate these areas into a common, practically applicable framework, e.g., along the lines of the DWQ project [4], and to have their results transferred into industrial tools, as this is probably the only way to have the techniques applied by companies.

The topics of data streams and real-time data warehousing are already very hot database research topics, so here is an example of a close synergy between industry and research. Here, an especially challenging issue is the integration of data streams and data integration (ETL, cleansing, quality) in one unified framework.

The final area, "soft issues" is really not so much a computer science area as an information systems one, but it is nonetheless very important to industry. Here, research should focus on BI process improvement, management of BI requirements, and BI project management, as well as people issues.

On the negative side, it seems that the great research effort going into, e.g., multidimensional modeling and data mining, is not really very relevant to industry. However, the author believes that this is to quite a large extent due to the current level of "BI maturity" in most companies.

6 Conclusion and Future Work

Motivated by the author's experiences from running BI knowledge exchange activities and the relative lack of information about BI use in industry, this industrial paper presented the experiences that the author has obtained from running knowledge exchange activities such as network meetings and workshops, with partners in industry. The paper presented the network participants, the topics of the knowledge exchange activities, and the lessons learned over the last couple of years. Lessons relate to the (surprisingly low) popularity of OLAP and data mining tools, the time spent on different tasks in BI projects, legacy compatibility, and the importance of "soft" people-related issues. Finally, the paper made some recommendations on which research topics would be useful in industry.

In the future, the aim is to bring in researchers with other academic backgrounds than computer science into the network. The Department of Production is already in, focusing on BI use rather than BI technology, but researchers with an economic/financial background, e.g., in Cost-Based Accounting, could also be of great value. Additionally, it is the aim to bring in (a few) more participants to increase the impact of the network, and perhaps even try to invite BI tool vendors for some of the meetings.

References

1. BusinessMinds (in Danish). http://www.businessminds.dk . Current as of March 19, 2004.
2. Gartner Group. Business Activity Monitoring Gartner Group, 2002.

3. J. Gehrke (ed). Special Issue on Data Stream Systems. *IEEE Data Engineering Bulletin* 26(1), 2003.
4. M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis (eds.), Fundamentals of Data Warehousing, 2nd. Ed., Springer-Verlag, 2003.
5. KMD home page (in Danish). http://www.kmd.dk . Current as of March 19, 2004.
6. Nouhauz home page (mostly in Danish). http://www.nouhauz.dk. Current as of March 18, 2004.
7. D. Pedersen, K. Riis, and T. B. Pedersen. XML-Extended OLAP Querying. In *Proceedings of the Fourteenth International Conference on Scientific and Statistical Database Management*, pp. 195–206, 2002.
8. T. B. Pedersen. BIT home page (in Danish). http://www.cs.aau.dk/ tbp/BIT . Current as of March 19, 2004.
9. Platon home page (in Danish). http://www.platon.dk . Current as of March 19, 2004.
10. Nykredit home page (in Danish). http://www.nykredit.dk . Current as of March 19, 2004.
11. Sonofon home page (in Danish). http://www.sonofon.dk . Current as of March 19, 2004.
12. P. Vassiliadis. Gulliver in the land of data warehousing: practical experiences and observations of a researcher. In *Proceedings of the Second International Workshop on Design and Management of Data Warehouses*, paper 12, 2000.

Towards an Adaptive Approach for Mining Data Streams in Resource Constrained Environments

Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy

School of Computer Science and Software Engineering, Monash University
900 Dandenong Rd, Caulfield East, VIC3145, Australia
{Mohamed.Medhat.Gaber,Arkady.Zaslavsky,
Shonali.Krishnaswamy}@infotech.monash.edu.au

Abstract. Mining data streams in resource constrained environments has emerged as a challenging research issue for the data mining community in the past two years. Several approaches have been proposed to tackle the challenges of limited capabilities for small devices that generate or receive data streams. These approaches try to approximate the mining results with acceptable accuracy and efficiency in space and time complexity. However these approaches are not resource-aware. In this paper, a thorough discussion about the state of the art of mining data streams is presented followed by a formalization of our Algorithm Output Granularity (AOG) approach in mining data streams. The incorporation of AOG within a generic ubiquitous data mining system architecture is shown and discussed. The industrial applications of AOG-based mining techniques are given and discussed.

1 Introduction

A data stream is a sequence of unbounded, very high data rate incoming data elements that can be read only once by an algorithm. [26] [19] [18][2]. For example in NASA Earth Observation System (EOS), a pair of Landsat 7 and Terra spacecraft generates 350 GB of data per day [29] and an oil drill can transmit data about its current drilling conditions at 1 Mb/Second [27].

There are two possibilities for the data stream generators with respect to the mobile device. The mobile device could be the data source for example analyzing data generated by sensor nodes, oil drills and seismic readings rather than transferring these huge amounts of data streams to a central site to be analyzed. The mobile device receives data streams from other data stream generators for example analyzing stock market data that is relevant to the user, and analyzing data generated from hospitals around the world that could be used by a specialized medical doctor to keep up to date anytime anywhere by the analyzed information about the new cases that could be used as ad hoc information during the investigations or even the medical operations.

Mining data streams on a mobile device raises a number of issues and challenges for the data mining research community. The following list presents some of these issues and challenges [12][23][2]:

- Handling the continuous flow of data streams.
- Minimizing energy consumption of the mobile device.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 189–198, 2004.

- Unbounded memory requirements due to the continuous flow of data streams.
- Required result accuracy.
- Transferring data mining results over a wireless network with a limited bandwidth.
- Data mining results' visualization on the small screen of the mobile device.
- Modeling mining results' changes over time.
- Developing algorithms for mining results' changes.
- Interactive mining environment to satisfy user requirements.

The techniques proposed so far in the literature try to address the above challenges using different approaches that will be discussed later in the next section. We have proposed Algorithm Output Granularity (AOG) in mining data streams. The AOG is distinguished from the other approaches by being adaptive and resource-aware.

AOG is a resource aware adaptable approach in mining data streams. It is concerned with the amount of knowledge stored in main memory before any incremental integration of the mining results for a time interval. The approach uses an algorithm threshold that controls the algorithm output rate according to the available memory and time of result residency in main memory. The time interval is the acceptable time for a mining technique to have the results resident in memory before incrementing the knowledge in order to continue processing the non-ended stream. We call the algorithms that apply the AOG approach light-weight mining algorithms: LWC for clustering, LWClass for classification and LWF for frequent items or patterns [12][13].

The paper is organized as follows. Section 2 shows related techniques and projects in mining data streams in resource constrained environments. The AOG approach is discussed in details in section 3. Section 4 shows the incorporation of AOG within our ubiquitous data mining system architecture. The potential applications of AOG-based mining techniques are presented in section 5. Section 6 concludes the paper and discusses the future directions.

2 Related Work

There are different algorithms proposed to deal with the high speed nature for mining data streams using different techniques. Clustering data streams has been studied in [1], [4], [6], [7], [9], [10], [17], [24], [28]. Data stream classification has been studied in [11], [14], [20], [30], [36]. Extracting frequent items and frequent itemsets have been studied in [8], [16], [25]. Data stream mining projects reported in [21], [22], [23], [31], [32]. Querying data streams have been studied in [2], [33], [34], [35]. Table 1. summarizes the most cited data stream mining techniques according to the mining task, the used approach and the status of implementation.

The above algorithms tackle the problem of mining data streams using different methodologies. Our proposed solution for output rate adaptation has the advantage of the generality to any algorithm [12],[13] as well as the simplicity of implementation.

3 Algorithm Output Granularity

The process of data mining could be viewed as the process of building a model that represents the input data as accurate as possible to be used in data description or in a

Table 1. Mining Data Stream Algorithms

Algorithm	**Mining Task**	**Approach**	**Implementation**
VFKM	K-Means	Sampling and reducing the number of passes at each step of the algorithm	Implemented and tested.
VFDT	Decision Trees	Sampling and reducing the number of passes at each step of the algorithm	Implemented and tested.
Approximate Frequent Counts	Frequent itemsets	Incremental Pruning and update of itemsets with each block of transactions	Implemented and tested.
FP- Stream	Frequent itemsets	Incremental Pruning and update of itemsets with each block of transactions and time-sensitive patterns extension	Implemented and tested.
Concept-Drifting Classification	Classification	Ensemble classifiers	Implemented and tested.
AWSOM	Prediction	Incremental Wavelets	Implemented and tested (This algorithm is designed to run on a sensor). The implementation is not on a sensor.
Approximate K-median	K-Median	Sampling and reducing the number of passes at each step of the algorithm	Analytical Study
GEMM	General Applied to decision tress and frequent itemsets	Sampling	Analytical study
CDM	Decision Trees, Bayesian Nets and clustering	Fourier spectrum representation of the results to save the limited bandwidth	Implemented and tested.
ClusStream	Clustering	Online summarization and offline clustering	Implemented and tested
STREAM-LOCALSEARCH	Clustering	Sampling and incremental learning	Implemented and tested against other techniques

prediction process. According to this view, there are two main factors that should be taken into consideration in the mining process: the tradeoff between accuracy of the model and size of the output and the tradeoff between the accuracy and running time of the mining algorithm.

The requirements of mining data streams motivate the need for one-pass model builder. The AOG is a similarity based approach. The similarity based model is a very fast one-pass multipurpose model builder. The similarity based model is the process of storing a summary of the input data that could be used in clustering, classification and frequent patterns mining processes. This summary contains the means of data points' clusters and the number of point in each group. A variation from that could be used for classification that in addition to the means and number of points, we store also the dominant class. Another variation could be used in the frequent patterns process by picking up the clusters' means that contain the maximum number of points.

The AOG approach is based on the following axioms:

a) The algorithm output rate (AR) is function in a data rate (DR), i.e., *AR = f(DR).*
b) The time needed to fill the available memory of the mobile device by the algorithm results (TM) is function in (AR), i.e., *TM = f(AR).*
c) The algorithm accuracy (AC) is function in (TM), i.e., *AC = f(TM).*

The following are the concepts used in AOG based approach:

Algorithm Threshold: is a controlling parameter built in the algorithm logic that encourages or discourages the creation of new outputs according to three factors that vary over temporal scale:

a) Available memory.
b) Remaining time to fill the available memory.
c) Data stream rate.

The algorithm threshold is the maximum acceptable distance between the group means and the data element of the stream. The higher the threshold, the lower output size would be produced. The algorithm threshold can use Euclidean or Manhattan distance functions and a normalization process would be done online in case of multidimensional data stream.

Threshold Lower Bound: is the minimum acceptable distance (similarity measure) that could be used. As a matter of fact is the less the threshold the higher the algorithm accuracy. If the distance measure is very small, it has two major drawbacks. It is meaningless in some applications to set the distance measure in a very small value such as astronomical applications. The distance between some of data elements in such applications is relatively high. And the smaller the threshold, the higher the running time for the model use.

Threshold Upper Bound: is the maximum acceptable similarity measure that could be accepted to produce meaningful results. If the distance measure is high, the model building would be faster; however it has the limitation of producing meaningful results; that is not to group data elements that are totally different in the same class or cluster.

Output Granularity: is the amount of generated results that are acceptable according to a pre- specified accuracy measure. This amount should be resident in memory before doing any incremental integration.

Time Threshold: is the required time to generate the results before any incremental integration according to some accuracy measure. This time might be specified by the user or calculated adaptively based on the history of running the algorithm.

Time Frame: is the time between each two consecutive data rate measurements. This time varies from an application to another and from one mining technique to another.

The Main Steps for Mining Data Streams Using Our Proposed Approach:

1) Determine the frequency of adaptation and mining.
2) According to the data rate, calculate the algorithm output rate and the algorithm threshold.
3) Mine the incoming stream using the calculated algorithm threshold.
4) Adjust the threshold after a time frame to adapt with the change in the data rate using linear regression.
5) Repeat the last two steps till the algorithm lasts the time interval threshold.
6) Perform knowledge integration of the results.

The algorithm output granularity in mining data streams has primitive parameters and operations that operate on these parameters. AOG algebra is concerned with defining these parameters and operations. The development of AOG-based mining techniques should be guided by these primitives depending on empirical studies. That means defining the timing settings of these parameters to get the required results.

AOG Parameters

TFi The time frame i
Di: Input data stream during the time frame i.
I(Di): Average data rate of the input stream Di.
O(Di): Average output rate resulting from mining the stream Di

AOG Operations

(Di) Mining process of the Di stream.
([I(D1), O(D1)],…,[I(Di), O(Di)]) Adaptation process of the algorithm threshold at the end of the time frame i.
Ω **(Oi, ...,Ox)** Knowledge integration process done on the output I to the output x.

AOG Settings

D(TF) Time duration of each time frame.
D(Ω) Time duration between each two consecutive knowledge integration process.

The AOG have been used to implement a number of data mining techniques with encouraging results reported in [12],[13]. The incorporation of AOG within a resource-aware ubiquitous data mining is discussed in the following section.

4 RA-UDM System Architecture

The research so far in the ubiquitous data mining field does not pay much attention of how to exploit the increasing computation power in mobile devices in the data mining task preserving the limited bandwidth of data transfer. Motivated by this fact and the increasing need for resource-aware data analysis systems in commercial and scientific applications for the huge data streams generated continuously, we propose a new resource aware UDM system that incorporates our approach in mining data streams using algorithm output granularity. Figure 1 shows our resource aware ubiquitous data mining (RA-UDM) system architecture.

Resource-Aware Component

Local Resource Information: This module has the ability to inform the system about the mobile device resources' measurement such as the available memory, CPU utilization, battery consumption… etc.

Context-Aware Middleware: This component can inform the system by the environmental information such as the available communication channels and the effective bandwidth.

Resource Measurements: This module acts as a resource measurement receiver from both local and environment.

Solution Optimizer: This module determines the data mining task scenario according to the available information about the local and environmental resources. This module is responsible for the initiation of the data mining task and the calculation of the initial parameters for the data mining technique.

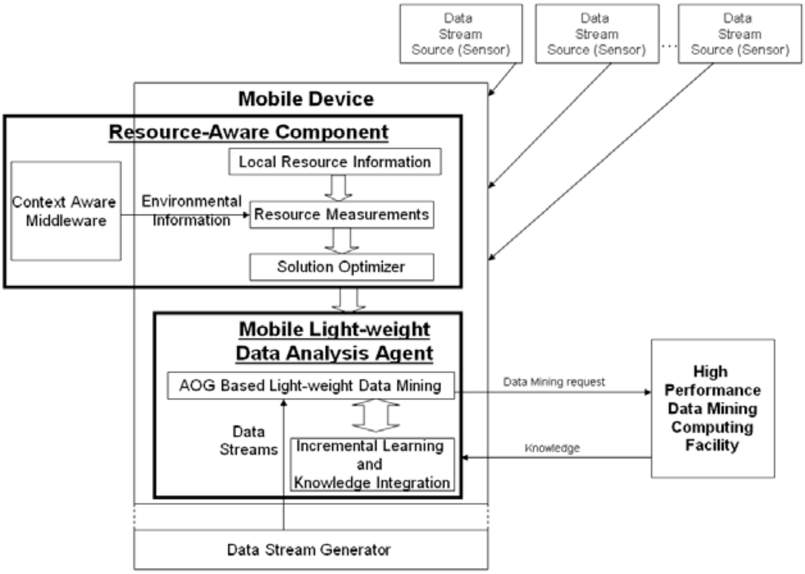

Data Stream Source (Sensor)
Data Stream Source (Sensor)
...
Data Stream Source (Sensor)
Mobile Device
Resource-Aware Component
Local Resource Information
Context Aware Middleware
Environmental Information
Resource Measurements
Solution Optimizer
Mobile Light-weight Data Analysis Agent
AOG Based Light-weight Data Mining
Data Streams
Incremental Learning and Knowledge Integration
Data Mining request
Knowledge
High Performance Data Mining Computing Facility
Data Stream Generator

Fig. 1. RA-UDM System Architecture

Mobile Light-Weight Data Analysis Agent

Light-Weight Data Mining Agent: This module is the core of our system. It has the ability to perform the data mining task faster than the data stream rate. This agent uses the AOG based approach. The mobility factor gives this module the ability to be transferred to another node to continue processing the stream in case of sudden lack of computing resources in the current hosting node.

Incremental Learning and Knowledge Integration: This module has the ability to update the current stored results with the incoming new data from data sources or knowledge from the server by interacting with the data mining module.

Data Stream Generator: If the mobile device generates data streams such as context information for a mobile device, or on-board sensor readings in astronomical applications [32].

High Performance Data Mining Computing Facility: This facility can run the data mining applications upon the mobile device requests. It has the capability to integrate the knowledge produced from different nodes. It could be a grid computing facility with a considerable high performance.

5 Potential Applications

The AOG approach incorporated in RA-UDM system architecture could be used in different application domains. This section discusses some of the potential applications of the proposed approach.

- **Astronomical**

There are two main motivations for using a light weight approach in mining astronomical data generated on-board a satellite; the infeasibility of transferring huge amounts of data streams to ground stations due to the limited bandwidth, and the limited computational power on-board the satellite. Figure 2. shows a typical scenario for using AOG-based approach in analyzing astronomical data streams. The projects described in [31][32] show the importance of such light-weight analysis algorithms on-board a satellite.

- **Business**

The business applications for the proposed approach can vary. Analyzing stock market data streams received at the user's PDA is one of the important applications [23]. Mining network traffic data for the purpose of monitoring and analyzing these data at real-time, then taking a decision about intrusion detection is another potential application. Mining web logs and web clickstreams are other examples. The adoption of AOG in such applications can overcome the problem of the high fluctuations of the incoming data.

- **Scientific**

One of the ambitious applications is to use a light-weight analysis for a real-time feedback in a physics or chemistry laboratory. The mining results could be used to change the experimental settings. Such an application has its importance in the acceleration of achieving the experiment objectives. Figure 3. shows a general framework for such an application.

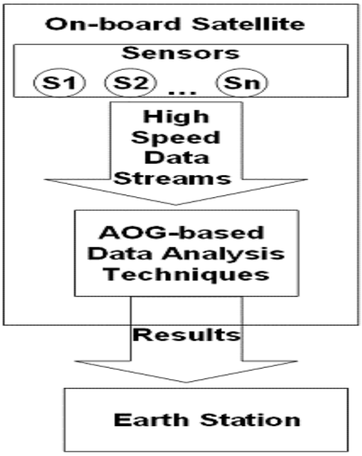


Fig. 2. A Typical astronomical Scenario for AOG approach

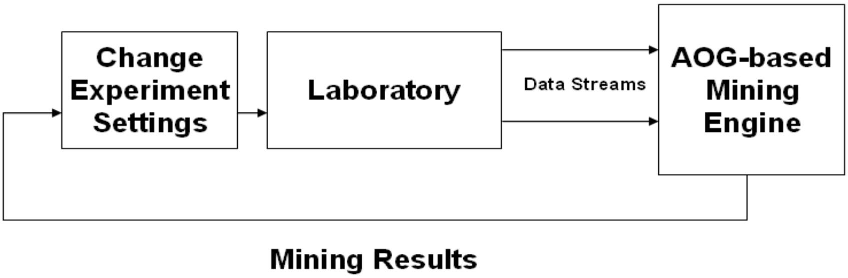


Fig. 3. Real-time laboratory data analysis

6 Conclusions and Future Work

The AOG based approach in mining data stream is a generic resource-aware approach that is efficient to resource constrained environments with limited processing capabilities. The incorporation of AOG in the generic RA-UDM system would realize a robust data analysis system. This system could be used in a number of potential business and scientific applications that generate huge numbers of data streams. These applications include but are not limited to: stock market data analysis, sensor readings in different scientific applications and web clickstream.

There are different directions to our future work. The implementation of RA-UDM within a sensor network environment is the ultimate goal of the project. The adoption of AOG approach to other data mining techniques is another direction of our future work. The realization of the AOG approach for on-board data analysis in astronomical applications is the direction that we try to realize. For example, rather than sending the data from Mars planet using the sensors on the robot, the robot can use AOG based analysis techniques to analyze the data stream and sending the results back to the earth preserving the limited bandwidth.

References

1. Aggarwal C., Han J., Wang J., Yu P. S.: A Framework for Clustering Evolving Data Streams. Proc. 2003 Int. Conf. on Very Large Data Bases (VLDB'03), Berlin, Germany (2003).
2. Babcock B., Babu S., Datar M., Motwani R., and Widom J.: Models and issues in data stream systems. In Proceedings of PODS (2002).
3. Babcock B., Datar M., and Motwani R.: Load Shedding Techniques for Data Stream Systems (short paper). In Proc. of the 2003 Workshop on Management and Processing of Data Streams (MPDS 2003) (2003).
4. Babcock B., Datar M., Motwani R., O'Callaghan L.: Maintaining Variance and k-Medians over Data Stream Windows. To appear in Proceedings of the 22nd Symposium on Principles of Database Systems (PODS 2003) (2003).
5. Burl M., Fowlkes C., Roden J., Stechert A., and Mukhtar S. Diamond Eye: A distributed architecture for image data mining. In SPIE DMKD, Orlando, April (1999).
6. Charikar M., O'Callaghan L., and Panigrahy R.: Better streaming algorithms for clustering problems. In Proc. of 35th ACM Symposium on Theory of Computing (STOC) (2003).
7. O'Callaghan L., Mishra N., Meyerson A., Guha S., and Motwani R.: Streaming-data algorithms for high-quality clustering. Proceedings of IEEE International Conference on Data Engineering, March (2002).
8. Cormode G., Muthukrishnan S.: What's hot and what's not: tracking most frequent items dynamically. PODS 2003. (2003) 296-306
9. Datar M., Gionis A., Indyk P., Motwani R.: Maintaining Stream Statistics over Sliding Windows (Extended Abstract). In Proceedings of 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002) (2002).
10. Domingos P. and Hulten G., A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. Proceedings of the Eighteenth International Conference on Machine Learning, 106–113, Williamstown, MA, Morgan Kaufmann. (2001)
11. Domingos P. and Hulten G. Mining High-Speed Data Streams. In Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining, (2000) 71–80.
12. Gaber, M.M., Krishnaswamy, S. and Zaslavsky, A. (2004). Cost-Efficient Mining Techniques for Data Streams. In Proc. Australasian Workshop on Data Mining and Web Intelligence (DMWI2004), Dunedin, New Zealand. CRPIT, 32. Purvis, M., Ed. ACS.
13. Gaber, M, M., Krishnaswamy, S., and Zaslavsky, A., Adaptive Mining Techniques for Data Streams Using Algorithm Output Granularity, The Australasian Data Mining Workshop (AusDM 2003), Held in conjunction with the 2003 Congress on Evolutionary Computation (CEC 2003), December, Canberra, Australia, Springer Verlag, Lecture Notes in Computer Science (LNCS).
14. Ganti V., Gehrke J., Ramakrishnan R.: Mining Data Streams under Block Evolution. SIGKDD Explorations 3(2): (2002) 1-10.
15. Garofalakis M., Gehrke J., Rastogi R.: Querying and mining data streams: you only get one look a tutorial. SIGMOD Conference 2002: 635. (2002).
16. Giannella C., Han J., Pei J., Yan X., and Yu P.S.: Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In Kargupta H., Joshi A., Sivakumar K., and Yesha Y. (eds.), Next Generation Data Mining, AAAI/MIT (2003).
17. Guha S., Mishra N., Motwani R., and O'Callaghan L.: Clustering data streams. In Proceedings of the Annual Symposium on Foundations of Computer Science. IEEE, November (2000).
18. Golab L. and Ozsu M. T. : Issues in Data Stream Management. In SIGMOD Record, Volume 32, Number 2, June (2003) 5-14.
19. Henzinger M., Raghavan P, and Rajagopalan S.: Computing on data streams. Technical Note 1998-011, Digital Systems Research Center, Palo Alto, CA, May (1998).

20. Hulten G., Spencer L., and Domingos P.: Mining Time-Changing Data Streams. ACM SIGKDD (2001).
21. Kargupta H.: CAREER: Ubiquitous Distributed Knowledge Discovery from Heterogeneous Data. NSF Information and Data Management (IDM) Workshop (2001).
22. Kargupta. H.: VEhicle DAta Stream Mining (VEDAS) Project. http://www.cs.umbc.edu/%7Ehillol/vedas.html. (2003).
23. Kargupta, H., Park, B., Pittie, S., Liu, L., Kushraj, D. and Sarkar, K. (2002). MobiMine: Monitoring the Stock Market from a PDA. ACM SIGKDD Explorations. January 2002. Volume 3, Issue 2. Pages 37–46. ACM Press.
24. Keogh E., Lin J., and Truppel W.: Clustering of Time Series Subsequences is Meaningless: Implications for Past and Future Research. In proceedings of the 3rd IEEE International Conference on Data Mining. Melbourne, FL. November (2003) 19-22.
25. Manku G. S. and Motwani R.: Approximate frequency counts over data streams. In Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China, August (2002).
26. Muthukrishnan S.: Data streams: algorithms and applications. Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms (2003).
27. Muthukrishnan S.: Seminar on Processing Massive Data Sets. Available Online: http://athos.rutgers.edu/%7Emuthu/stream-seminar.html (2003).
28. Ordonez C.: Clustering Binary Data Streams with K-means .ACM DMKD (2003).
29. Park B. and Kargupta H.. Distributed Data Mining: Algorithms, Systems, and Applications. Data Mining Handbook. Editor: Nong Ye (2002).
30. Papadimitriou S., Faloutsos C., and Brockwell A.: Adaptive, Hands-Off Stream Mining. 29th International Conference on Very Large Data Bases VLDB (2003).
31. Srivastava A. and Stroeve J.: Onboard Detection of Snow, Ice, Clouds and Other Geophysical Processes Using Kernel Methods. Proceedings of the ICML'03 workshop on Machine Learning Technologies for Autonomous Space Applications (2003).
32. Tanner S., Alshayeb M., Criswell E., Iyer M., McDowell A., McEniry M., Regner K., EVE: On-Board Process Planning and Execution, Earth Science Technology Conference, Pasadena, CA, Jun. 11 - 14, (2002).
33. Tatbul N., Cetintemel U., Zdonik S., Cherniack M. and Stonebraker M.: Load Shedding in a Data Stream Manager. Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), September (2003).
34. Tatbul N., Cetintemel U., Zdonik S., Cherniack M. and Stonebraker M.: Load Shedding on Data Streams. In Proceedings of the Workshop on Management and Processing of Data Streams (MPDS 03), San Diego, CA, USA, June (2003).
35. Viglas S. D. and Naughton J.: Rate based query optimization for streaming information sources. In Proc. of SIGMOD (2002).
36. Wang H., Fan W., Yu P. and Han J.: Mining Concept-Drifting Data Streams using Ensemble Classifiers. In the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Aug., Washington DC, USA (2003).

Exploring Possible Adverse Drug Reactions by Clustering Event Sequences*

Hongxing He[1], Graham Williams[1], Jie Chen[1],
Simon Hawkins[1], and Chris Kelman[2]

[1] Data Mining Research, CSIRO Math and Information Sciences
GPO Box 664, Canberra, ACT 2601, Australia
Firstname.Lastname@csiro.au
[2] Commonwealth Department of Health and Ageing, Australia
Christopher.Kelman@health.gov.au

Abstract. Historically the identification of adverse drug reactions relies on manual processes whereby doctors and hospitals report incidences to a central agency. In this paper we suggest a data mining approach using administrative pharmaceutical usage data linked with hospital admissions data. Patients, represented by temporal sequences of drug usage, are clustered using unsupervised learning techniques. Such techniques rely on a distance measure, and we propose in this paper such a distance measure for comparing drug usage sequences based on an event-type hierarchy, based around the hierarchical drug classification system. Although developed for a specific domain, we indicate that it is applicable in other applications involving data where event types form a hierarchical structure, such as is found in telecommunications applications. The approach modifies the Uniform Kernel K-Nearest Neighbour Clustering algorithm to constrain the merging of clusters to those clusters within a specified distance. The approach avoids losing clusters that are less dense yet far apart, as would occur without such a modification, but is typical of the types of applications we are interested in (where outliers are important). We demonstrate the algorithm through a successful application exploring for possible adverse drug events, in particular exploring hospital admissions for severe angioedema resulting from the usage of certain drugs and drug combinations. The interesting clusters thus identified have given clues to medical researchers for further investigations.

1 Introduction

Systematic monitoring of adverse drug reactions is important for both financial and social reasons. In general, the early detection of unexpected adverse reactions relies on a local voluntary reporting system and collated statistics from overseas agencies. The availability of a population-based prescribing data set, such as the Pharmaceutical Benefits Scheme (PBS) data in Australia, when linked to

* The authors acknowledge the Commonwealth Department of Health and Ageing, and the Queensland Department of Health for providing data for this research.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 199–208, 2004.

hospital admissions data, provides a unique opportunity to detect common and rare adverse reactions as the data becomes available and at a much earlier stage before too many patients are affected. In studying possible adverse drug reactions from administrative data, like PBS, the entities of interest are patients contain of all of a patients interactions with the health care system, as temporal event sequences. Each event, for our purposes, records the dispensing of a prescribed drug to a patient at a particular time. We can formally represent a temporal event sequence as $s =< (e_1, t_1), (e_2, t_2), \ldots, (e_i, t_i), \ldots, (e_m, t_m) >$. For each (e_i, t_i), e_i indicates an event type and t_i is a timestamp for the event. Traditionally, temporal event sequences have been transformed, as part of the *feature extraction* stage, into a few static features describing some of the characteristics of the sequences [1, 3]. Inevitably, information is lost in converting the original temporal sequences into a static feature set.

When clustering patients represented by event sequences, we need to determine a "distance" between sequences. It is well known that distance determination between sequences is very complex, usually involving complicated procedures such as dynamic programming [7]. In this paper, we present a method for determining the distance between temporal event sequences as might be found in many application areas, including health. The method takes into account similarity of event-types, dealing with similarity at a number of levels within an event-type hierarchy using an international drug classification system to identify similarity between drug types. We introduce an *event-type hierarchy* to support the distance measure (Section 3.2). Patients, represented by their drug usage event sequences, are clustered using the Uniform Kernel KNN Clustering algorithm [11]. We refine the algorithm with a user specified distance constraint to avoid distant clusters being otherwise merged (Section 4.2).

Section 2 reviews related work. Section 3 introduces our distance measure, with our clustering method presented in Section 4. Experimental data and results are given in Section 5. Conclusions and discussion follow in Section 6.

2 Related Work

Similarity of temporal sequences is a key problem in clustering temporal sequence data. A number of different approaches have been developed for computing the similarity between two temporal sequences and searching similar patterns using Edit Distances [4, 8, 7] or conditional probability [13]. Edit Distances use dynamic programming to identify the minimum editing operations needed to transform one sequence into another, and thereby measure distance based on the number of operations required. In contrast to the common distance measure, we introduce a computationally inexpensive method using an *event-type hierarchy* in this study.

Similar to clustering event sequences, but given in attribute-value representation, COBWEB [2] can produce a classification scheme on observed objects. In [9] the authors implement the idea of agglomerative hierarchical clustering and use frequently occurring subsequences as features describing data sequences.

However the similarity measure defined in this paper takes into account not only general similarity but also the similarity based on an event-type hierarchy. Traditional *K-Means* clustering, and related methods [5], can not be applied directly to cluster temporal event sequences since they rely on identifying a mean as the centroid of the cluster. For temporal event sequences, there is no concept of a centroid. An alternative clustering method which requires only a distance measure is *K-Medoids* [5]. Uniform Kernel KNN is another clustering algorithm that only requires knowledge of the distances between all pairs of data points.

3 Distance Measure of Event Sequences

The calculation of a distance between two event sequences of different length is rather difficult if detailed differences are all taken into account. In considering the drug usage sequences from administrative data, the detailed differences are not important. The PBS dataset records the date of purchase of the prescribed drugs. For the purpose of identifying possible adverse drug reactions, the date of usage is more relevant. Therefore with the dataset available, the best we can do is to use the date of purchase as the approximation to the date of usage. This causes significant inaccuracy in the time of occurrence of an event. Consequently we can only identify what kind of drugs or drugs combinations are used during certain periods of time with any certainty. In considering the distance between such temporal event sequences, we might consider only two scenarios: the temporal event sequences are *similar* or they are *different*.

Temporal event sequences might be found to be *similar* when we ignore time and frequency, as in the following event sequences:$< abc >, < bca >, < aabbc >, < acbbba >$. The sequences here have the same combination of events but have different order and different frequency. The temporal event sequences $< ab >, < bcad >, < aabef >, < acmnnb >$ *different*, even though they have some common events. In computing the distance between such event sequences, we take into account both their similarities and their differences. We develop a similarity measure base on *event-type hierarchy* to quantify the differences.

3.1 Dissimilarity of Event Sequences

We do not need to consider the difference between event sequences in detail due to the limitations of the drug usage event sequences data. We only consider the following three levels of dissimilarity of event sequences:

Identical Sequences. Sometime we choose not to consider the time of occurrences of events within a fixed time window. In this case, two sequences s and s' are considered identical if they differ only in the time of occurrences. The distance between them is zero.

Same Combination of Event Types. If two sequences consist of same event types, then the distance between two sequences is β. The β is an adjustable parameter.

Different Combination of Event Types. If two sequences are neither identical nor composed of same event type combination, then we consider if they have common event types or similar event types defined by event-type hierarchy in Section 3.2. The distance is therefore defined as follows:

$$distance(s, s') = 1.0 - sim(s, s'), \tag{1}$$

where $sim(s, s')$ is defined in Section 3.2

3.2 Event-Type Hierarchy

Consider the example of drug prescriptions where two drugs are essentially the same but have different manufacturer and/or labels. At one level, they are clearly identical. We introduce an *event-type hierarchy* to assist in this situation. The event-type hierarchy identifies a number of levels of aggregation whereby events might be identified as being the same at some level.

Consider drug prescriptions where drugs are identified using their $ATC\ code$[1]. Figure 1 illustrates a small portion of the defined hierarchy. There are, for example, 14 alternatives at level 1 in the hierarchy (from drugs associated with *Alimentary tract and metabolism* to drugs associated with *Sensory organs*).

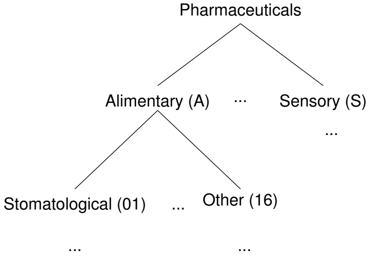


Fig. 1. WHO Code Hierarchy

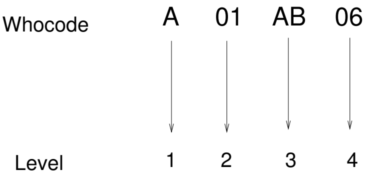


Fig. 2. Sample WHO Code

The codes used to identify drugs within the ATC system encapsulate the hierarchy in seven characters. Drugs can be identified as $A03AA01$, $A02BA06$, $A03FA01$ and $C09AA01$. The first character denotes the level 1 ATC code and can be one of: A,B,C,D,G,H,J,L,M,N,P,R,S and V. The next two characters are digits and form the level 2 ATC code. The fourth and fifth letters define the level 3 ATC code and the final two digits identify the level 4 ATC code. Figure 2 illustrates the *ATC code* for a specific drug (Domiphen). This drug belongs to *alimentary tract and metabolism* (A) at level 1, *stomatological preparations* (A01) at level 2, and *anti-infectives for local/oral treatment* (A01AB) at level 3.

We identify events e_1 and e_2 as being the *same at level* i with respect to a user supplied hierarchy, such as above, if they fall within the same node at level i. For convenience, we identify the maximum level at which they share a

[1] Anatomical Therapeutic Chemical (ATC) classification system: http://www.whocc.no/atcddd/atcsystem.html

node as the level at which they are the same. Thus we identify that the two drugs $A03AA07$ and $A03AA01$ as being the same at level 3 and the two drugs $A13BD08$ and $A03AA07$ the same at level 1.

We introduce a similarity measure with each level $i = 1, 2, \ldots, n_l$, where n_l is the number of levels in the hierarchy. We count the number of events (e.g., drugs) which are the *same at level* i and denote it as c_i.

For each level we then identify the proportion of "same events" in each sequence ($c_i/|s|$ and $c_i/|s'|$) and use the average as the contribution toward similarity at that level. We also introduce a *boost factor* as the parameter θ ($0 \leq \theta \leq 1$) to boost the similarity measure from 0. This is particularly useful for long sequences.

At a specific level i we thus measure the similarity between two sequences as:

$$sim_i(s, s') = \begin{cases} 0 & \text{if } c_i = 0 \\ \frac{\theta + (1-\theta)\frac{1}{2}(\frac{c_i}{|s|} + \frac{c_i}{|s'|})}{n_l} & \text{otherwise.} \end{cases} \tag{2}$$

We now sum the similarities at each level, recognising that comparisons at each level have different contributions (weighted by the adjustable parameters γ_i). The total similarity between two sequences can then be expressed as:

$$sim(s, s') = \sum_{i=1}^{n_l} \gamma_i sim_i(s, s'). \tag{3}$$

3.3 Distance Measure

For a pair of sequences, s and s', we now have a distance measure based on either their dissimilarity or, if the event sequences do not consist of the same event type combination, a similarity measure based on the *event-type hierarchy*. The final distance between the two sequences, can be expressed as Equation 4

$$distance(s, s') = \begin{cases} 0.0 & identical\ sequences \\ \beta & same\ combination\ of\ event\ types \\ 1.0 - sim(s, s') & otherwise. \end{cases} \tag{4}$$

The distance so defined will take values $0 \leq distance(s, s') \leq 1$ if the following constraints are satisfied: $0 \leq \beta \leq 1$ and $0 \leq \gamma_i \leq 1$ for $i = 1, 2, \ldots, n_l$.

4 UNKNN and Its Modifications

A distance measure can be used to cluster entities. Representing entities (e.g., patients) by their temporal event sequences we now describe an algorithm, and our modification to the algorithm, to cluster the entities. After presenting the clustering algorithm we discuss how to measure the quality of the resulting clusters.

4.1 Uniform Kernel KNN Clustering

Uniform Kernel KNN Clustering [11] has been used by [6] for clustering sequences. Uniform Kernel KNN Clustering begins by considering each sequence as a cluster and then merging clusters based on the notion of density. We measure the density around a temporal event sequence s_i ($s_i \in \mathcal{S}$, $\mathcal{S}$ is the set of all sequences) as the number of sequences, n, within a specific region determined by some distance d of s_i. d is the distance to the furthest sequence among those sequences which are within the set of the n-nearest neighbours. The k shortest distinct distances from s_i to other sequences corresponds to n sequences. Note that $n \geq k$ since a number of sequences may be the same distance from s_i. Figure 3 illustrates this for s_1 with $k = 3$. Here, s_2 and s_3 are the same distance from s_1, hence $n = 4$ and $d = 5$.

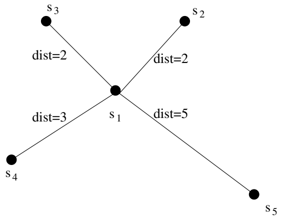


Fig. 3. k-nearest distances

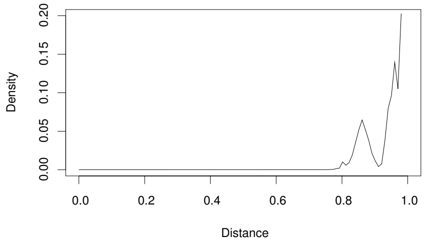


Fig. 4. Distance distribution

The density of a sequence is then calculated as:

$$density(s_i) = \frac{n}{|\mathcal{S}|d}. \tag{5}$$

The algorithm considers the n nearest neighbours as candidates to merge into the cluster containing s_i. A cluster is merged if it has higher density than s_i and there is no other closer cluster in the n neighbours having higher density than s_i. The density of the new combined cluster is the maximum of the densities of the two constituent clusters.

For all clusters that have no nearest neighbour with density greater than itself, we merge into the cluster any other cluster containing nearest neighbours and having equal density. This step merges "plateau neighbour regions".

4.2 Distance Constraint

Only the density of the nearest neighbours is used in the algorithm to decide whether clusters should be merged. For outliers, the nearest neighbours can be sequences that are actually quite far away, leading to outliers being merged, inappropriately, into its "nearest neighbour", potentially losing what otherwise may be important, distinct clusters. We introduce a new constraint to limit the merging of clusters to just those clusters within a specified distance δ, so that the cluster containing s_j is merged into the cluster containing s_i if $distance(s_i, s_j) < \delta$.

The value of δ should reflect the data being clustered. We can use a simple formula to calculate δ as the global mean distance between all sequences, multiplied by an adjustable parameter α. The effect of introducing this distance constraint is demonstrated in Section 4.4.

4.3 Quality of Clusters

The aim of clustering is to partition objects into a number of good, non-overlapping clusters. A good cluster has objects in the same cluster being close to each other and objects in different clusters being further apart. We can measure the quality of clustering by introducing the following concepts based on [5].

- We identify the *medoid* of a cluster of temporal event sequences as that sequence in the cluster having the minimum mean distance to all other sequences in the cluster, which is defined as the *mean inner medoid distance.* The mean inner medoid distance over all clusters is defined as *overall mean inner medoid distance.*
- The *mean inner cluster distance* of a cluster is the overall mean distance between each pair of sequences in the cluster. The *overall mean inner cluster distance* is the mean of the inner cluster distance means over all clusters.
- The *mean inter cluster distance* between two clusters is the mean pairwise distance between sequences in each cluster. The *overall mean inter cluster distance* is the mean of the inter cluster distance means over all clusters pairs.
- The *overall mean inter medoid distance* is the average medoid to medoid distance over all possible pairs of clusters.

We employ the following two ratios to evaluate the quality of our clusters:

- The *Inner-Inter Cluster Ratio* is the ratio of the overall mean inner cluster distance to the overall mean inter cluster distance.
- The *Inner-Inter Medoid Ratio* is the ratio of the overall mean inner medoid distance to the overall mean inter medoid distance.

For a given set of clusters, smaller Inner-Inter ratios are certainly desirable. The Inner-Inter Cluster Ratio is more accurate as a quality measure, since it considers distance between all pairs, but is computationally more expensive.

4.4 Effect of Distance Constraint

The constrained Uniform Kernel KNN Clustering algorithm can be compared to the unconstrained algorithm using the quality of cluster measures outlined above. We use the sequence of drugs used by a patient over a six month period prior to a hospitalisation for angioedema as our temporal event sequences. Table 1 compares the results with and without the distance constraint.

The additional distance constraint successfully separates a number of outliers. The resulting Inner-Inter Cluster and Medoid Ratios are almost halved. The minimum distance δ is calculated from the global mean distance between sequences, as discussed in Section 4.2. Here we use $\alpha = 0.9$.

Table 1. Comparison of clustering with and without constraints

Additional distance constraint:	Without	With
Number of clusters	10	42
Inner-Inter Cluster Ratio	0.64	0.34
Inner-Inter Medoid Ratio	0.48	0.25
Outliers	3	26

Table 2. Distance parameters

Parameter	Description	Value
β	Same combination	0.1
γ_1	Hierarchy level 1	0.1
γ_2	Hierarchy level 2	0.2
γ_3	Hierarchy level 3	0.3
γ_4	Hierarchy level 4	0.4
θ	Boost factor	0.3

5 Experimental Data and Results

5.1 Data Description

The Queensland Linked Data Set [12] links hospital admissions data from Queensland Health with the Commonwealth Department of Health and Ageing's pharmaceutical prescription data. In the context of discovering relationships between drug prescriptions leading to hospitalisations, we construct temporal event sequences based on six months of prescriptions prior to a hospitalisation for a particular disease (angioedema). Angioedema was chosen on the basis of professional advice from Australia's regulating agency (the Therapeutic Goods Administration), because it represented a small population of just 222 patients, useful for experimentation.

Angioedema is a swelling (large welts or weals), where the swelling is beneath the skin rather than on the surface. It is associated with the release of histamine and other chemicals into the bloodstream, and is part of the allergic response. The swelling may occur in the face, neck, and in severe cases may compromise breathing.

5.2 Distance Measure

The dataset was clustered using the distance measure for temporal event sequences described in Section 3. The parameter values used for the distance measure are listed in Table 2.

The distribution of pairwise distances over the 222 patients is shown in Figure 4 where it is clear that the distance measure is primarily above 0.8.

5.3 Clustering

We apply the Constrained Uniform Kernel KNN Clustering with $k = 3$ (the number of nearest neighbours) and $\alpha = 0.9$ (distance constraint parameter).

The algorithm identified 42 clusters (see Table 3). Twenty six of these were outliers containing just one member and are not included in the Table. The overall inner cluster distance is small and the overall inter cluster distance is large, giving a good Inner-Inter Cluster Ratio of 0.34. Similarly, the Inner-Inter Medoid Ratio is 0.25.

Below we identify some of the interesting clusters and their characteristics. We only list a few interesting clusters, where the patients in the same cluster

Table 3. Clusters details(outliers not included)

Cluster Index	Cluster Size	Mean Inner Medoid Dist	Mean Inner Cluster Dist
0	4	0.58	0.78
5	71	0.83	0.90
8	8	0.72	0.84
11	31	0.83	0.88
12	2	0.40	0.81
14	2	0.41	0.82
16	5	0.66	0.84
18	6	0.70	0.86
19	2	0.41	0.82
23	4	0.61	0.84
25	9	0.73	0.85
26	23	0.71	0.89
35	2	0.39	0.78
36	3	0.55	0.83
38	19	0.80	0.88
41	5	0.65	0.89
Overall		0.24	0.32

share many common characteristics and the characteristics may be interesting to the medical researchers.

- Cluster 11 has 31 patients of which 94% have used cardiovascular system drugs. 84% of patients have used ACE inhibitors. Subsequent investigation has identified that ACE inhibitor usage is associated with an increased risk of hospitalisation for angioedema, a clear adverse reaction. Most patients in this cluster have used ACE inhibitors during the six month prior to their angioedema hospitalisation. There are a number of case series in the literature demonstrating that ACE inhibitor-related angioedema is responsible for as many as 40% of angioedema episodes [10].
- Cluster 26 has 23 patients of which 82% has taken antibacterials for systemic use. In particular, 78% of the patients have used the amoxicillin, and it may be of value to further investigate the relationship between this class of drugs and angioedema.
- Cluster 16 is a small cluster with only 5 patients, all having used a combination of the following four drug groups: cardiovascular system, musculoskeletal system, genito urinary and sex hormones, and general anti-infectives. Since adverse drug reactions are usually rare events, a small group of patients sharing such a pattern may be worthy of further investigation.

6 Conclusion and Discussion

This paper develops an effective method for comparing drug usage temporal event sequences, through general dissimilarity and a similarity measure based on event-type hierarchy. The Uniform Kernel KNN Clustering algorithm is modified to cluster the temporal event sequences based on the proposed distance measure. The modified algorithm ensures that outliers are retained.

The method is applied to real world drug usage sequence. The clusters produced were shown to identify significant areas for further clinical study in the area of hospitalisation resulting from adverse drug reactions. The results revealed well known adverse drug reaction of angioedema caused by a popular blood pressure lowering drug i.e. ACE inhibitor. Some possible unknown adverse reactions of severe angioedema caused by usage of drugs such as anti-bacterials, amoxicillin or usage of combinations of wide range of drugs may be of some values for further investigations. Although the method has been applied to only adverse drug reactions of severe angioedema, it can be easily applied to a wide range of adverse effects of various medical conditions.

The initial methodology, as presented, provides many opportunities for extension and further development. For example, the current experimentation has not included patient demographics such as gender and age, or other static features like co-morbidity and levels of sickness. The further study may take these static features into considerations in clustering patients in exploring possible adverse drug reactions. Moreover, it is definitely interesting to study the problem of adjusting the parameters used in the method automatically.

References

1. R. A. Baxter, G. J. Williams, and H. He. Feature selection for temporal health records. *Lecture Notes in Computer Science*, 2035:198–209, 2001.
2. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, (2):139–172, 1987.
3. V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM)*, pages 179–186, 2001.
4. D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.
5. L. Kaufman and P. Pousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, New York, 1990.
6. H. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP: Approximate mining of consensus sequential patterns. In *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM)*, pages 311–315, 2003.
7. H. Mannila and P. Ronkainen. Similarity of event sequences. In *Proceedings of 4th International Workshop on Temporal Representation and Reasoning (TIME 1997)*, pages 136–139, 1997.
8. P. Moen. *Attribute,Event Sequence,and Event Type Similarity Notions for Data Mining.* PhD thesis, University of Helsinki Finland, Jan. 2000.
9. T. Morzy, M. Wojciechowski, and M. Zakrzewicz. Scalable hierarchical clustering method for sequences of categorical values. In *Proceedings of 5th Pacific-Asia International Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 282–293, 2001.
10. M. Reid, B. Euerle, and M. Bollinger. Angioedema. In *URL: http://www.emedicine.com/med/topic135.htm*, 2002.
11. SAS. Proc Modeclust. In *SAS online documentation*, 2000.
12. G. Williams, D. Vickers, R. Baxter, S. Hawkins, C. Kelman, R. Solon, H. He, and L. Gu. The Queensland Linked Data Set. Technical Report CMIS 02/21, CSIRO Mathematical and Information Sciences, Canberra, 2002.
13. J. Yang and W. Wang. CLUSEQ: Efficient and effective sequence clustering. In *Proceedings of the International Conference on Data Engineering (ICDE03)*, pages 101–112, 2003.

SCLOPE: An Algorithm for Clustering Data Streams of Categorical Attributes

Kok-Leong Ong[1], Wenyuan Li[2], Wee-Keong Ng[2], and Ee-Peng Lim[2]

[1] School of Information Technology, Deakin University
Waurn Ponds, Victoria 3217, Australia
leong@deakin.edu.au

[2] Nanyang Technological University, Centre for Advanced Information Systems
Nanyang Avenue, N4-B3C-14, Singapore 639798
liwy@pmail.ntu.edu.sg, {awkng,aseplimg}@ntu.edu.sg

Abstract. Clustering is a difficult problem especially when we consider the task in the context of a data stream of categorical attributes. In this paper, we propose `SCLOPE`, a novel algorithm based on `CLOPE`'s intuitive observation about cluster histograms. Unlike `CLOPE` however, our algorithm is very fast and operates within the constraints of a data stream environment. In particular, we designed `SCLOPE` according to the recent `CluStream` framework. Our evaluation of `SCLOPE` shows very promising results. It consistently outperforms `CLOPE` in speed and scalability tests on our data sets while maintaining high cluster purity; it also supports cluster analysis that other algorithms in its class do not.

1 Introduction

In recent years, the data in many organizations take the form of continuous streams, rather than finite stored data sets. This possesses a challenge for data mining, and motivates a new class of problem called *data streams* [1, 2]. Designing algorithms for data streams is a challenging task: (a) there is a sequential one-pass constraint on the access of the data; (b) and it must work under bounded (i.e., fixed) memory with respect to the data stream.

Also, the continuity of data streams motivates time-sensitive data mining queries that many existing algorithms do not adequately support. For example, an analyst may want to compare the clusters, found in one window of the stream, with clusters found in another window of the same stream. Or, an analyst may be interested in finding out how a particular cluster evolves over the lifetime of the stream. Hence, there is an increasing interest to revisit data mining problems in the context of this new model and application.

In this paper, we study the problem of clustering a data stream of categorical attributes. Data streams of such nature, e.g., transactions, database records, Web logs, etc., are becoming common in many organizations [3]. Yet, clustering a categorical data stream remains a difficult problem. Besides the dimensionality and sparsity issue inherent in categorical data sets, there are now additional

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 209–218, 2004.

stream-related constraints. Our contribution towards this problem is the SCLOPE algorithm inspired by two recent works: the CluStream [4] framework, and the CLOPE [3] algorithm.

We adopted two aspects of the CluStream framework. The first is the *pyramidal* timeframe, which stores summary statistics at different time periods at different levels of granularity. Therefore, as data in the stream becomes outdated, its summary statistics looses details. This method of organization provides an efficient trade-off between the storage requirements and the quality of clusters from different time horizons. At the same time, it also facilities the answering of time-sensitive queries posed by the analyst.

The other concept we borrowed from CluStream, is to separate the process of clustering into an *online* micro-clustering component and an *offline* macro-clustering component. While the online component is responsible for efficient gathering of summary statistics (a.k.a *cluster features* [4]), the offline component is responsible for using them (with the user inputs) to produce the different clustering results. Since the offline component does not require access to the stream, this process is very efficient.

Set in the above framework, we report the design of the online and offline components for clustering categorical data organized within a pyramidal timeframe. We begin with the online component, where we propose an algorithm to gather the required statistics in one sequential scan of the data. Using an observation in the FP-Tree [5], we eliminated the need to evaluate the clustering criterion. This dramatically drops the cost of processing each record, and allows it to keep up with the high data arrival rate.

We then discuss the offline component, where we based its algorithmic design on CLOPE. We were attracted to CLOPE because of its good performance and accuracy in clustering large categorical data sets, i.e., when compared to k-means, CLARANS [6], ROCK [7], and LargeItem [8]. More importantly, its clustering criterion is based on *cluster histograms*, which can be constructed quickly and accurately (directly from the FP-Tree) within the constraints of a data stream environment.

2 Maintenance of Summary Statistics

For ease of discussion, we assume that the reader are familiar with the CluStream framework, the CLOPE algorithm, and the FP-Tree [5] structure. Also, without loss of generality, we define our clustering problem as follows. A data stream $\mathcal{D}$ is a set of records $\mathcal{R}_1, \ldots, \mathcal{R}_i, \ldots$ arriving at time periods $t_1, \ldots, t_i, \ldots$, such that each record $\mathcal{R} \in \mathcal{D}$ is a vector containing attributes drawn from $\mathcal{A} = \{a_1, \ldots, a_j\}$. A clustering $\mathcal{C}_1, \ldots, \mathcal{C}_k$ on $\mathcal{D}_{(t_p,t_q)}$ is therefore a partition of records $\mathcal{R}_x, \mathcal{R}_y, \ldots$ seen between t_p and t_q (inclusive), such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_k = \mathcal{D}_{(t_p,t_q)}$ and $\mathcal{C}_\alpha \neq \varnothing$ and $\forall \alpha, \beta \in [1, k]$, and $\mathcal{C}_\alpha \cap \mathcal{C}_\beta = \varnothing$.

From the above, we note that clustering is performed on all records seen in a given time window specified by t_p and t_q. To achieve this without accessing the stream (i.e., during offline analysis), the online micro-clustering component has to maintain sufficient statistics about the data stream. Summary statistics, in this case, is an attractive solution because they have a much lower space requirement than the stream itself. In SCLOPE, they come in the form of micro-clusters and cluster histograms. We define them as follows.

Definition 1 (Micro-Clusters). *A micro-cluster $\mu^{\mathcal{C}}$ for a set of records $\mathcal{R}_x$, $\mathcal{R}_y, \ldots$ with time stamps $t_x, t_y, \ldots$ is a tuple $\langle \overline{L}, \overline{H} \rangle$, where $\overline{L}$ is a vector of record identifiers, and $\overline{H}$ is its cluster histogram.*

Definition 2 (Cluster Histogram). *The cluster histogram $\overline{H}$ of a micro-cluster $\mu^{\mathcal{C}}$ is a vector containing the frequency distributions $\texttt{freq}(a_1, \mu^{\mathcal{C}}), \ldots$, $\texttt{freq}(a_{|\mathcal{A}|}, \mu^{\mathcal{C}})$ of all attributes $a_1, \ldots, a_{|\mathcal{A}|}$ in $\mu^{\mathcal{C}}$, In addition, we define the following derivable properties of $\overline{H}$:*

- *the **width**, defined as $|\{a : \texttt{freq}(a, \mu^{\mathcal{C}}) > 0\}|$, is the number of distinct attributes, whose frequency in $\mu^{\mathcal{C}}$ is not zero.*
- *the **size**, defined as $\sum_{i=1}^{|\mathcal{A}|} \texttt{freq}(a_i, \mu^{\mathcal{C}})$, is the sum of the frequency of every attribute in $\mu^{\mathcal{C}}$.*
- *the **height**, defined as $\sum_{i=1}^{|\mathcal{A}|} \texttt{freq}(a_i, \mu^{\mathcal{C}}) \times |\{a : \texttt{freq}(a, \mu^{\mathcal{C}}) > 0\}|^{-1}$, is the ratio between the size and width of $\overline{H}$.*

2.1 Algorithm Design

We begin by introducing a simple example. Consider a data stream $\mathcal{D}$ with 4 records: $\{\langle a_1, a_2, a_3\rangle, \langle a_1, a_2, a_5\rangle, \langle a_4, a_5, a_6\rangle, \langle a_4, a_6, a_7\rangle\}$. By inspection, an intuitive partition would reveal two clusters: $\mathcal{C}_1 = \{\langle a_1, a_2, a_3\rangle, \langle a_1, a_2, a_5\rangle\}$ and $\mathcal{C}_2 = \{\langle a_4, a_5, a_6\rangle, \langle a_4, a_6, a_7\rangle\}$, with their corresponding histograms: $\overline{H_{\mathcal{C}_1}} = \{\langle a_1, 2\rangle, \langle a_2, 2\rangle, \langle a_3, 1\rangle, \langle a_5, 1\rangle\}$ and $\overline{H_{\mathcal{C}_2}} = \{\langle a_4, 2\rangle, \langle a_5, 1\rangle, \langle a_6, 2\rangle, \langle a_7, 1\rangle\}$. Suppose now we have a different clustering, $\mathcal{C}_1' = \{\langle a_1, a_2, a_3\rangle, \langle a_4, a_5, a_6\rangle\}$ and $\mathcal{C}_2' = \{\langle a_1, a_2, a_5\rangle, \langle a_4, a_6, a_7\rangle\}$. We then observe the following, which explains the intuition behind CLOPE's algorithm:

- clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ have better intra-cluster similarity then $\mathcal{C}_1'$ and $\mathcal{C}_2'$; in fact, records in $\mathcal{C}_1'$ and $\mathcal{C}_2'$ are totally different!
- the cluster histograms of $\mathcal{C}_1'$ and $\mathcal{C}_2'$ have a lower size-to-width ratio than $\overline{H_{\mathcal{C}_1}}$ and $\overline{H_{\mathcal{C}_2}}$, which suggests clusters with higher intra-cluster similarity have higher size-to-width ratio in their cluster histograms.

Ideally, a straightforward application of CLOPE should provide us with the summary statistics we need. Unfortunately, CLOPE requires multiple scans of the data, where the number of iteration depends on the desired level of intra-cluster similarity. This violates the one-pass requirement. Furthermore, CLOPE requires multiple evaluation of the clustering criterion for each record, an expensive operation when the size of the stream is massive.

Our solution in SCLOPE is based on the following observation: the optimal height of individual cluster histograms (for each micro-cluster) can be obtained

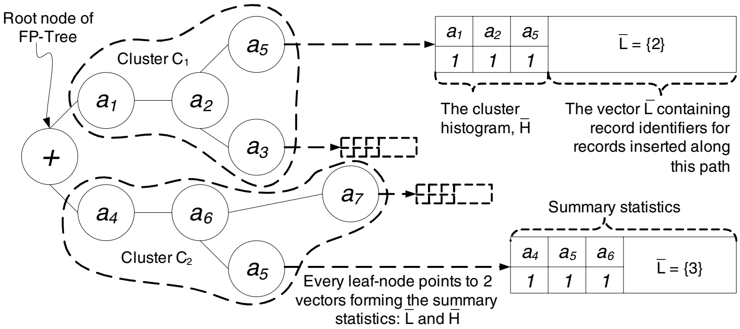

Root node of FP-Tree
Cluster C₁
Cluster C₂
+
a₁
a₂
a₃
a₄
a₅
a₆
a₇
a₅
a₁ a₂ a₅
1 1 1
L = {2}
The cluster histogram, H
The vector L containing record identifiers for records inserted along this path
Summary statistics
a₄ a₅ a₆
1 1 1
L = {3}
Every leaf-node points to 2 vectors forming the summary statistics: L and H

Fig. 1. Each path in the FP-Tree leads to a cluster histogram $\overline{H}$ and a vector $\overline{L}$ containing the record identifiers. Notice that records with common attributes share common prefixes, leading to a natural way of identifying clusters.

from a FP-Tree–like [5] structure. And this can be done in one sequential scan without the need to compute the clustering criterion. To understand this observation, we first revisit our example but this time with the FP-Tree. The formalism and algorithm follows after that.

Figure 1 shows the FP-Tree constructed for our example. We have omitted the *count*, *node links* and *header table* in the original FP-Tree definition as they are not needed in SCLOPE. We also ignore the one-pass constraint for the time-being, and note that the FP-Tree in the figure requires two scans – the first to determine the singleton frequency, i.e., $\texttt{freq}(a, \mathcal{D}_{(t_p,t_q)})$, and the second to insert each $\mathcal{R} \in \mathcal{D}_{(t_p,t_q)}$ into the FP-Tree after arranging all attributes $a \in \mathcal{R}$ according to their descending singleton frequency.

The implication above is that similar records are inherently "clustered" together through the sharing of a common prefix in the FP-Tree. In our example, we can visually confirm two natural clusters from the common prefixes a_1, a_2 and a_4, a_5, which suggests that $\mathcal{C}_1$ and $\mathcal{C}_2$ would be a better clustering than $\mathcal{C}'_1$ and $\mathcal{C}'_2$. In other words, we can actually consider each path (from the root to a leaf node) to be a micro-cluster, where the common prefixes *suggest* the micro-clusters to merge. This leads us to the following.

Observation 1. *An* FP-Tree *construction on* $\mathcal{D}_{(t_p,t_q)}$ *produces a set of micro-clusters (not necessary the optimal)* $\mu_1^{\mathcal{C}}, \ldots, \mu_k^{\mathcal{C}}$*, where k is determined by the number of unique paths* $\mathcal{P}_1, \ldots, \mathcal{P}_k$ *in the* FP-Tree*.*

Due to space, we shall skip the rationale of *all* our observations, and refer the reader to our technical report [9]. Nevertheless, we want to point out the fact that Observation 1 does not guarantee an optimal result. While this may not sound ideal, it is often sufficient for most stream applications. In fact, a near optimal solution that can be quickly obtained (without the need to evaluate the clustering criterion) is preferred in the design of the online component.

Not obvious is that CLOPE's clustering technique, which is to maximize the height of its cluster histograms, is closely related to the properties of FP-Tree's

Algorithm 1 Online Micro-clustering Component of SCLOPE

```
on begining of (window $w_i$) do
if ($i = 0$) then $\mathcal{Q}' \leftarrow \{$a random order of $v_1, \ldots, v_{|\mathcal{A}|}\}$
$\mathcal{T} \leftarrow$ new FP-Tree and $\mathcal{Q} \leftarrow \mathcal{Q}'$
for all (incoming record $\mathcal{R} \in \mathcal{D}_{(t_p,t_q)}$) do
  order $\mathcal{R}$ according to $\mathcal{Q}$ and $\forall a \in \mathcal{R}$, freq($a, \mathcal{Q}'$)++
  if ($\mathcal{R}$ can be inserted completely along an existing path $\mathcal{P}_i$ in $\mathcal{T}$) then
    $\forall a \in \mathcal{R}$, $\overline{L_i} \leftarrow \overline{L_i} \cup$ rid($\mathcal{R}_i$) $\wedge$ freq($a, \overline{H_i}$)++
  else
    $\mathcal{P}_j \leftarrow$ new path in $\mathcal{T}$ and $\overline{H_j} \leftarrow$ new cluster histogram for $\mathcal{P}_j$
    $\forall a \in \mathcal{R}$, freq($a, \overline{H_j}$) $\leftarrow 1$ and $\forall a \notin \mathcal{R}$, freq($a, \overline{H_j}$) $\leftarrow 0$
  end if
end for
on end of (window $w_i$) do
$\mathcal{L} \leftarrow \{\langle n,$ height($n$)$\rangle : n$ is node in $\mathcal{T}$ with $> 2$ children$\}$
order $\mathcal{L}$ according to height($n$)
while ($|\overline{H}_1, \ldots| > \varphi$) do
  select $\langle n,$ height($n$)$\rangle \in \mathcal{L}$ where $\forall n \neq m$, height($n$) $\geqslant$ height($m$)
  select paths $\mathcal{P}_i, \mathcal{P}_j$ where $n \in \mathcal{P}_i, \mathcal{P}_j$
  $\overline{H_{new}} \leftarrow \overline{H_i} \cup \overline{H_j}$
  delete $\overline{H_i}, \overline{H_j}$
end while
output micro-clusters $\mu_1^{\mathcal{C}}, \ldots, \mu_\varphi^{\mathcal{C}}$ and cluster histograms $\overline{H_1}, \ldots, \overline{H_\varphi}$ for $w_i$
```

construction. Recall that each path in the FP-Tree can contain multiple records, and that the construction is to maximize the overlapping (or sharing of common prefixes) of nodes, we actually have a natural process of obtaining a good cluster histogram for each micro-cluster. Observation 2 states this property.

Observation 2. *Given a micro-cluster $\mu_i^{\mathcal{C}}$ from a path $\mathcal{P}_i$, its cluster histogram $\overline{H_i}$ has a height that is naturally optimized (again, not necessary optimal) by the* FP-Tree *construction process.*

In the simplest case, once the FP-Tree is obtained, we can output the micro-clusters as the summary statistics for offline analysis. Unfortunately, these micro-clusters are often too fine in granularity and thus, continue to consume a lot of disk space. One solution is to agglomeratively merge the micro-clusters until they are sufficiently lightweight. However, doing so by evaluating the clustering criterion will prevent the algorithm from keeping up with the data rate of the stream. A strategy to do this efficiently is required.

Observation 3. *Given any micro-cluster $\mathcal{C}_i$ in the* FP-Tree *and its corresponding unique path $\mathcal{P}_i$, the micro-cluster(s) that give a good intra-cluster similarity (when merged with $\mathcal{C}_i$) are those whose paths overlap most with $\mathcal{P}_i$.*

Essentially, the above observation answers the question: How can we quickly determine, without the need to evaluate the clustering criterion, the micro-cluster to merge with t+he one under consideration? Since stream applications require only approximate results, we can conveniently exploit the property of common prefixes (i.e., Observation 3) to select the pair of micro-clusters to be merged. This is realized in Algorithm 1, lines $12 - 19$, where the key operation is to merge the cluster histograms and the record identifiers.

The idea is to start at the node having more than one child (i.e., more than one path/micro-cluster). This node would be the furthest from the root (therefore, lines $12 - 13$) and thus, contains the set of paths with the longest common prefix. By Observation 3, any two paths passing through this node would have a good intra-cluster similarity. Thus, we select any two paths passing through the node, and merge its corresponding cluster histograms and record identifiers (i.e., $\overline{H_i}$ and $\overline{H_j}$, lines $17 - 18$). This process repeats until the set of micro-clusters are sufficiently reduced to fit in the given space.

2.2 Working in Bounded Memory

Up to this point, we have shown how the FP-Tree is used to produce the summary statistics we need. In this sub-section and the next, we discuss how we made adjustments to satisfy the data stream constraints.

We begin with the issue of bounded memory. Without doubt, any attempt to process an unbounded data stream is likely to exhaust the limited computing resources before producing any results. To overcome this, we process the stream in a sliding window fashion. We assume δ to be the space allocated for storing summary statistics in the pyramidal timeframe. In the beginning, δ will be uniformly shared with each window having w_s space. For easy discussion, this space can be expressed in terms of the maximum number of micro-clusters (and histograms) allowed in a given window.

At the start of each window, we begin with an empty FP-Tree and insert each record into the data structure according to the rules given in Algorithm 1, lines $4 - 9$. This continues until we reach the end of the window, where we begin FP-Tree minimization to produce the summary statistics of size w_s. Clearly, by this process, there will be a time when the δ space is filled by the first δ/w_s windows. Therefore, space must be created for the subsequent windows, i.e., $(\delta/w_s) + 1, \ldots, (\delta/w_s) + j, \ldots$, and so on. In the pyramidal timeframe, there are two strategies to do so: *compress* and *delete*.

Intuitively, we first make room by compressing the statistics that became old, and then deleting them as they become outdated. Our strategy to create space for the subsequent $(\delta/w_s) + 1$ windows is to redistribute the δ spaces among all the $(\delta/w_s) + p$ windows created so far. In other words, rather then to have w_s amount of space for each window, we reduce w_s by a fraction using a "decay" function: $(1 - e^{-\mu}) \times w_s$ that is dependent on the window's age. Thus, if we have seen $(\delta/w_s) + p$ windows, then the size of the $(\delta/w_s) + j^{th}$ window, would be $(1 - e^{-j/p}) \times w_s$ where $1 \leqslant j \leqslant p$ (see [9]).

The final step is to resize the summary statistics in each window. We first reconstruct the FP-Tree from the summary statistics. This procedure is similar to the FP-Tree construction, where we simply insert a path (i.e., a group of records) instead of a record at a time. We then perform minimization until the

Algorithm 2 Offline Macro-clustering Component of SCLOPE

```
let $\mathbb{C} = \{\langle \overline{H}_{i+1}, \mu^{\mathcal{C}}_{i+1} \rangle, \ldots, \langle \overline{H}_{i+\varphi}, \mu^{\mathcal{C}}_{i+\varphi} \rangle, \ldots, \langle \overline{H}_{j+1}, \mu^{\mathcal{C}}_{j+1} \rangle, \ldots, \langle \overline{H}_{j+\varphi}, \mu^{\mathcal{C}}_{j+\varphi} \rangle\}$
repeat
  for all ($\mathcal{C}_{\mathcal{F}} \in \mathbb{C}$) do
    move $\mathcal{C}_{\mathcal{F}}$ to an existing cluster or new cluster $\mathcal{C}_j$ that maximizes profit
    if ($\mathcal{C}_{\mathcal{F}}$ has been moved to some cluster $\mathcal{C}_k$) then
      update cluster label of $\mathcal{C}_{\mathcal{F}}$ to $k$
    end if
  end for
until no further cluster is moved or processing time is exceeded
```

set of micro-clusters fit in the smaller space. Thus, older windows will have less space allocated and depending on the domain requirements, they may be deleted if they become obsolete.

2.3 Accessing Data in One-Sequential Pass

Our solution to the sequential one-pass access of data streams is to use an incremental update strategy to compute the ordering of attributes based on their descending singleton frequencies. The idea is simple: we begin by assuming a default order (Algorithm 1, line 1), e.g., attributes are seen in each incoming record. As we process each of them, we update the singleton frequency (line 4) of each attribute before inserting the record into the FP-Tree.

Upon reaching the end of window, we update the ordering of attributes (i.e., line 1), and use this new ordering in the next window. As a result, a record can have its attributes ordered differently in each window. Thus, it is possible to obtain a sub-optimal FP-Tree (initially) depending on the initial assumed order. Fortunately, this isn't an issue as the FP-Tree improves on optimality as the stream progresses. In our empirical results, this proved to be effective and reduces the construction to a single pass.

More importantly, this strategy is crucial to the success of exploiting Observation 3 for accurate clustering. Recall that a stream's characteristics actually changes over time, it will not be appropriate to use an assumed or pre-computed ordering. If it's used, a change in the stream's characteristics will caused all subsequent clustering to be sub-optimal, and there will not be any mechanism to recover from that. In that sense, our proposal is more robust because any sub-optimality in the FP-Tree (due to changing data characteristics) will be corrected on the next window cycle.

3 Cluster Discovery

Once summary statistics are generated, the analyst performs clustering over different time-horizons using the offline macro-clustering component. Since the offline component does not require access to the data, its design is not constrained by the one-pass requirement. Hence, we have Algorithm 2.

A typical time-sensitive cluster discovery begins with the analyst entering the time-horizon h, and the repulsion r. The time-horizon of interest usually spans one or more windows, and determines the micro-clusters involved in the analysis. On the other hand, the repulsion controls the intra-cluster similarity required, and is part of the clustering criterion defined as [3]:

$$\texttt{profit}(\{\mathcal{C}_1,\ldots,\mathcal{C}_k\}) = \left[\sum_{i=1}^{k}\left(\frac{\texttt{size}(\mathcal{C}_i)}{\texttt{width}(\mathcal{C}_i)^r}\times|\mathcal{C}_i|\right)\right]\times\left(\sum_{i=1}^{k}|\mathcal{C}_i|\right)^{-1}$$

The most interesting aspect of Algorithm 2 is its design for time-sensitive data mining queries. When used together with the pyramidal timeframe, we can analyze different parts of the data stream, by retrieving statistics of different granularity to produce the clustering we need. And since this is the offline component of SCLOPE (which can run independent of the data stream), our design favors accuracy over efficiency, i.e., it makes multiple iterations through the statistics, and cluster using the profit criterion.

Nevertheless, our offline component is still fast despite the fact that it is based on the design of CLOPE. The rationale behind this speed is that CLOPE works with one record at a time while SCLOPE works with a group of records at a time. In our algorithm, each micro-cluster is treated as a pseudo-record, and are clustered accordingly to the given r value that in turn, determines the number of clusters k. Since the number of pseudo-records are very much lower than the physical records, it takes less time to converge on the clustering criterion.

4 Empirical Results

The objective of our empirical tests is to evaluate SCLOPE in 3 aspects: *performance*, *scalability*, and *cluster accuracy*. Due to space constraints, we only report the overall results of our experiments. The interested can refer to our technical report [9] for all the test details.

Performance For an accurate comparison, we tested the performance of SCLOPE using only real-life data sets from the FIMI repository (*fimi.cs.helsinki.fi*). When we compared our results against CLOPE, the best algorithm for clustering categorical data sets, our proposal outperforms CLOPE by a large margin. On cases where the required number of micro-clusters is large, e.g., 200, CLOPE takes more than 10 hours to complete while SCLOPE comes in under 900 seconds. In all 4 real-life data sets tested, our results revealed that SCLOPE is very suitable for processing data streams given its low runtime and insensitivity to the number of clusters. This is crucial since the number of micro-clusters need to be inherently large to facilitate different analysis tasks.

Scalability To test scalability, we used the IBM synthetic data generator. We tested two aspects of scalability: the number of attributes, and the number of records. In the first test, we injected a data set of 50K records with different number of attributes from 467 to 4194. We then recorded the runtime of SCLOPE and CLOPE in creating 50, 100 and 500 clusters. In all cases, SCLOPE's runtime remains stable while CLOPE's runtime rises sharply as the attributes goes beyond 800. On the second part of the test, we vary the number of records from 10K to 500K. Again, SCLOPE is faster (or on par) with CLOPE in terms of their runtime for different number of clusters.

Accuracy In our final test, we used the mushroom data set (also from the FIMI repository) which contains two predefined classes: 4208 *edible* mushrooms and 3916 *poisonous* mushroom. To measure the accuracy, we used the purity metric (see [3]). In our experiment, we tried different combinations of w_s and φ which are the two parameters affecting the cluster quality in SCLOPE. From the results, SCLOPE consistently attains a higher purity than CLOPE in all situations. Together with SCLOPE's performance, we are convinced of its potential.

5 Related Work

Much of the early works in clustering were focused on numerical data, where most are efficient in situations where the data is of low-dimensionality. Representative of these include k-means, BIRCH [10], CLARANS [6], and CLIQUE [11].

In recent years, there has been a large amount of categorical data accumulated. Their dimensionality and size are often very much larger than numerical data, and exhibit unique characteristics that make numerical-based techniques awkward. This motivated the design of new algorithms leading to works such as CACTUS [12], ROCK [7], STIRR [13], and CLOPE.

While these algorithms are an advancement over numerical solutions, they are not designed with the constraints of data streams in mind. As a result, they are often resource intensive. For example, ROCK has a high computational cost, and require sampling in order to scale to large data sets. Closest to our work are therefore CluStream, STREAM [14], FC [15], and binary k-means.

In comparing the CluStream framework, our work differs by the virtue of the data type we investigate, i.e., we focus on categorical data. Likewise, STREAM and FC are numerical-based techniques, and is thus different from SCLOPE. In the case of binary k-means, a different clustering criterion is used, and its design does not support time-sensitive cluster analysis.

6 Conclusions

In this paper, we propose a fast and effective algorithm, called SCLOPE, that clusters an evolving categorical data stream. We chose to design our algorithm within the framework of CluStream so that it not only outperforms algorithms in its class, but also provide support for time-sensitive cluster analysis not found in most preceding works.

Our empirical tests, using real-world and synthetic data sets, proved that SCLOPE has very good performance and scalability. It also demonstrates good cluster accuracy despite the data stream constraints imposed on the algorithm.

More importantly, the accuracy of clusters generated by SCLOPE can be improved by varying the resource parameters: γ and δ, or allowing an extra scan of the data. This makes SCLOPE an attractive solution for clustering categorical data, either in the context of streams or conventional snapshots.

The drawback with the current design of SCLOPE is the lack of an error quantification on its approximated results. In some data stream applications, it is desirable for the analyst to specify the allowable error in the results, rather then to specify the amount of space. Therefore, an immediate future work would be to extend SCLOPE to handle both situations.

References

[1] Bradley, P.S., Gehrke, J., Ramakrishnan, R., Srikant, R.: Philosophies and Advances in Scaling Mining Algorithms to Large Databases. Communications of the ACM (2002)

[2] Hulten, G., Domingos, P.: Catching Up with the Data: Research Issues in Mining Data Streams. In: Workshop on Research Issues in Data Mining and Knowledge Discovery, Santa Barbara, CA (2001)

[3] Yang, Y., Guan, X., You, J.: CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data. In: Proc. SIGKDD, Edmonton, Canada (2002)

[4] Aggarwal, C., Han, J., Wang, J., Yu, P.S.: A Framework for Clustering Evolving Data Streams. In: Proc. VLDB, Berlin, Germany (2003)

[5] J. Han and J. Pei and Y. Yin: Mining Frequent Patterns without Candidate Generation. In: Proc. SIGMOD, Dallas, Texas, USA (2000)

[6] Ng, R., Han, J.: Efficient and Effective Clustering Methods for Spatial Data Mining. In: Proc. VLDB, Santiago de Chile, Chile (1994)

[7] Guha, S., Rastogi, R., Shim, K.: ROCK: A Robust Clustering Algorithm for Categorical Attributes. In: Proc. ICDE, Sydney, Austrialia (1999)

[8] Wang, K., Xu, C., Liu, B.: Clustering Transactions Using Large Items. In: Proc. CIKM, Kansas City, Missouri, USA (1999)

[9] Ong, K.L., Li, W., Ng, W.K., Lim, E.P.: SCLOPE: An Algorithm for Clustering Data Streams of Categorical Attributes. Technical Report (C04/05), Deakin University (2004)

[10] Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: AN Efficient Data Clustering Method for Very Large Databases. In: Proc. SIGMOD, Canada (1996)

[11] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: Proc. SIGMOD, Seattle, Washington, USA (1998)

[12] Ganti, V., Gehrke, J., Ramakrishnan, R.: CACTUS: Clustering Categorical Data Using Summaries. In: Proc. SIGKDD, San Diego, California, USA (1999)

[13] Gibson, D., Kleinberg, J.M., Raghavan, P.: Clustering Categorical Data: An Approach Based on Dynamical Systems. In: Proc. VLDB, New York, USA (1998)

[14] O'Callaghan, L., Meyerson, A., Motwani, R., Mishra, N., Guha, S.: Streaming Data Algorithms for High Quality Clustering. In: Proc. ICDE, USA (2002)

[15] Barbara, D.: Requirements for Clustering Data Streams. ACM SIGKDD Explorations **2** (2002)

Novel Clustering Approach that Employs Genetic Algorithm with New Representation Scheme and Multiple Objectives

Jun Du, Erkan Korkmaz*, Reda Alhajj**, and Ken Barker

Department of Computer Science, University of Calgary, Calgary, Alberta, Canada
{jundu,korkmaze,alhajj,barker}@cpsc.ucalgary.ca

Abstract. In this paper, we propose a new encoding scheme for GA and employ multiple objectives in handling the clustering problem. The proposed encoding scheme uses links so that objects to be clustered form a linear pseudo-graph. As multiple objectives are concerned, we used two objectives: 1) to minimize the *Total Within Cluster Variation (TWCV)*; and 2) minimizing the number of clusters in a partition. Our approach obtains the optimal partitions for all the possible numbers of clusters in the *Pareto Optimal* set returned by a single GA run. The performance of the proposed approach has been tested using two well-known data sets: *Iris* and *Ruspini*. The obtained results demonstrate improvement over classical approaches.

1 Introduction

Clustering approaches may be classified into: hierarchical, grid-based, partitioning and those based on co-occurrence of categorical data [1]. Some of the existing clustering approaches employ Genetic Algorithms (GA) in the process. However, such clustering techniques have some drawbacks, including: 1) redundancy seems to be a problem for the representations used [5]; 2) ensuring the validity of the chromosomes that appear throughout the search [10]; and 3) the number of clusters has to be specified beforehand in many of the proposed methods [10].

In this paper, we proposed a new GA-based clustering technique characterized mainly by the usage of multiple objectives and a new encoding scheme based on forming a linked-list structure from objects in the same cluster. So, the genetic operators have the effect of combining and dividing different linked lists that correspond to different clusters. The number of clusters does not need to be specified beforehand. A multi-objective GA [8] has been used with two conflicting objectives: 1) minimize *Total Within Cluster Variation (TWCV)*; and 2) minimize the number of clusters. TWCV [10] is a measure which denotes the sum of average distances of cluster elements to cluster center. If this measure is used as the single objective in the search, GA tends to reduce the sizes of the

* Erkan Korkmaz is partially supported by TUBITAK.
** Reda Alhajj completed part of this work during his visit to Global University, Beirut, Lebanon.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 219–228, 2004.

clusters and eventually clusters with almost single elements are formed. Hence, it is possible to use TWCV as the single objective function only with k-clustering. However, the new representation proposed in this paper is able to encode different number of clusters in a fixed-length chromosome. Therefore, the search is not limited with k-clustering and the general clustering problem is attacked by using multi-objective GA. The *Pareto Optimal* set [8] obtained at the end of the run provides different choices for the potential number of clusters and the best corresponding TWCV. The performance of the proposed approach has been tested using two well-known data sets: *Iris* [2] and *Ruspini* [13], which have been widely used as benchmarks for testing different techniques; the optimal clustering in each of the two domains is known. The results obtained are compared with the classical *Group Number Encoding* (GNE) [9] and it has been observed that a clear improvement has been achieved.

The rest of the paper is organized as follows. Previous work is presented in Section 2. The application of multi-objective GA to clustering and the objective functions used are presented in Section 3. The design of the new method is introduced in Section 4. The experimental results are discussed in Section 5. Section 6 includes summary and conclusions.

2 Related Work

In general, two different schemes are used for representing clustering in the field of GA. The first allocates each object to a different gene of a chromosome and the value of the gene indicates the cluster of the object; e.g., GNE presented in [9]. The number of clusters should be provided beforehand, and redundancy is quite high in the GNE. An attempt to overcome these drawbacks can be found in [4]. However, the representation used cannot encode different numbers of clusters in a fixed length chromosome. Thus a variable length GA [3] is used to encode the different numbers of clusters that might appear throughout the search.

GA is also used to increase the search power of other clustering methods. The classical *K-means* [7] algorithm has been widely used in different clustering applications. The initialization process in the algorithm might lead to a local optimum [12]. To overcome this drawback, a genetic based approach has been proposed in [11]. The method is based on encoding the cluster centers of the classic *K-means* algorithm as the genes of the chromosome. Hence, the task of the GA is to find out the appropriate cluster centers that minimize the clustering metric. A different approach can be found in [14] where the GA is used to search clustering rules. The aim is to eliminate the drawbacks of the classical minimization approaches and be able to obtain clusters which can have large differences in their sizes, densities or geometries.

The new approach proposed in this paper, shares some common properties with the work in [10], which has a single objective, i.e., to minimize *TWCV*; and the classic GNE is used as the representation of the clusters. Hence, it can only be used with k-clustering.

3 Multi-objective GA for Clustering

The *Niched Pareto Genetic Algorithm* presented in [8] is multi-objective GA based on the notion of *Pareto Domination* used during the selection operation. An element is considered to be *Pareto-dominant* over another one only if it is superior in terms of all the objectives used. Hence, a set of *Pareto-optimal* solutions is obtained at the end of the search, such that none of the elements is *Pareto-dominant* over another. Also, niching is an effective operation that applies a pressure on the search to spread genetic population along the pareto optimal surface.

In our clustering approach, we try to minimize the following two objectives: 1) TWCV [10], and 2) Number of clusters. It is apparent, given two different partition sizes, the smaller the partition size is, the smaller (or equal when duplicated data exists in dataset) the minimum TWCV for that partition size is. Therefore, targeting these two objectives in GA could possibly include solutions for all different partition sizes in the Pareto optimal set.

4 Linked-List Representation for Genetic Clustering

GNE [9] is the most straightforward encoding scheme and thus used most widely for clustering. The drawbacks of this traditional encoding for clustering are presented in [5]. The remedy proposed in [5] is to use a length variable encoding scheme. It reduces redundancy of chromosome population but adds redundancy inside a chromosome; it needs more genes to encode a solution than traditional encoding. Another deficiency of the length variable encoding is that it cannot take advantage of conventional simple crossover and mutation operators. The linear linkage (LL) encoding method we present in this section is a fixed length encoding scheme without any type of redundancy.

4.1 Linkage Encoding (LE) Scheme

Under LE scheme, although each gene stores an integer number, it no longer directly tells the membership of an object but its fellowship - this is the fundamental difference between GNE and LE. Each gene is a link from an object to another object of the same cluster. Given n objects, any partition on them can be described as a chromosome of length n. Two objects are in the same group if either object can be directed to the other object through the links. Without any constraint, the state of redundancy is just as bad as that of GNE because the number of feasible chromosomes is still n^n.

4.2 LL Encoding Scheme

LL encoding is a restricted LE. Let the n genes inside a chromosome be indexed from 1 to n. Each gene in a LL chromosome allows one integer value to store the index of a fellow gene in the same cluster. We can also treat the stored index as an out-link from a node. If a gene stores its own index, it depicts an ending node. To qualify an unrestricted linkage as a valid LE chromosome, the chromosome must comply for these constraints:

1. The integer value stored in each gene is greater than or equal to its index but less than or equal to n.
2. No two genes in a chromosome can have the same value; the only exception is index of an ending node.

LL encoding gets its name because the objects in a cluster construct a pseudo linear path with only a self reference on the last node allowed. It can be represented by the *labeled oriented pseudo* (LOP) graph.

Definition 1 (LOP Graph). *A LOP graph is a labeled directed graph $G(V, E)$, where V contains n vertices. A composition of G is a partition of $V(G)$ into disjointed oriented pseudo path graphs $G_1, G_2, ..., G_m$ with the following properties:*

1. *Disjoint paths: $\bigcup_{i=1}^{m} V(G_i) = V(G)$ and for $i \neq j, V(G_i) \bigcap V(G_j) = \emptyset$*
2. *Non-backward oriented edges: If there is an edge e directed from vertex v_l to v_k, then $l \leq k$.*
3. *Balanced connectivity:*
   (a) *$|E(G)| = 2 \times |V(G)|$;*
   (b) *each G_i must have only one (self referencing) ending node, which has an in-degree of 2 and an out-degree of 1;*
   (c) *each G_i must have only one starting node whose in-degree=0 and out-degree=1.*
4. *All other $|V(G_i)| - 2$ vertices in G_i have in-degree=out-degree=1.*

Some straightforward observations regarding LOP graphs: 1) Given a set of objects S, there exists one and only one composition of LOP graph $G(V, E)$ for each partition scheme of S, where $|V| = |S|$. 2) The number of LOP graphs is given by the n^{th} *Bell number*. 3) Linear LE is an implementation of the LOP graph.

Based on these observations, it is clear that LL encoding makes a one-to-one mapping between chromosomes and clustering solutions. Compared to LL encoding scheme, traditional GNE demands the GA to work in a solution space of $\frac{n^n}{B(n)}$ times larger.

Although LL encoding keeps only fellowship in gene, it also implies the membership of each object. As each cluster must have one starting node and one ending node, both nodes can be used to identify clusters. In practice, ending node is treated as the membership identifier for clusters because it is easier to be detected. Apparently, finding the membership of an object in LL encoding requires only linear time.

4.3 Initialization

The initial population should include diverse chromosomes. It is intuitive to achieve this goal by generating random chromosomes. However, such chromosomes may violate the restrictions of LL encoding. Table 1 illustrates what would happen if we use this approach to initialize the population. Suppose we are to

Table 1. Chromosome and partition mapping

Non-LL chromosomes (27)	Non-LL chromosomes without backward out-links (6)	LL chromosomes (5)	Partitions (5)
$(1,2,3)$	$(1,2,3)$	$(1,2,3)$	$(a)(b)(c)$
$(1,2,2)$ $(1,3,2)$ $(1,3,3)$	$(1,3,3)$	$(1,3,3)$	$(a)(bc)$
$(1,2,1)$ $(3,1,3)$ $(3,2,1)$ $(3,2,3)$	$(3,2,3)$	$(3,2,3)$	$(ac)(b)$
$(1,1,1)$ $(1,1,2)$ $(1,3,1)$ $(2,1,1)$ $(2,1,2)$ $(2,2,1)$ $(2,2,2)$ $(2,3,1)$ $(2,3,2)$ $(2,3,3)$ $(3,1,1)$ $(3,1,2)$ $(3,2,2)$ $(3,3,1)$ $(3,3,2)$ $(3,3,3)$	$(2,3,3)$ $(3,3,5)$	$(2,3,3)$	(abc)

find an optimal for three objects a, b, and c. Column one in Table 1includes all 27 different chromosomes. If LL encoding restrictions are applied, only five of them survive in column three. And there is a one-to-one mapping between column three and column four, which is the actual partition the chromosomes represent. From Table 1, it can be easily seen that the partition scheme $\{(abc)\}$ will dominate the initial population produced. Based on LL encoding constraint one, each integer should be equal to its index or the maximum integer index number. Therefore, the chromosome generator must create each gene based on this constraint. However, we should be aware that the chromosomes produced by such chromosome generator are not fully complied with the constraints laid for LL encoding. Obviously, those chromosomes do not have backward links but allow multiple nodes link to the same node. The last chromosome in column 2 of Table 1 is such an example. Generally, this chromosome generator will produce chromosomes that tend to have small number of clusters.

Although constructors ensure that there are no backward links in each chromosome, multiple links directed to the same node are still possible. Therefore, a recovery process is needed after the constructors to rectify a chromosome into its legitimate format. This process is also required after crossover and mutation operations where the structures of the population are modified. The rectify algorithm used for the recovery process employs two correction steps. First, backward links are eliminated from a chromosome. Then, multiple links to a node (except for the ending nodes) are replaced with one link in and one link out.

4.4 Selection, Crossover and Mutation

The selection process is very similar to that of Niched Pareto GA described in [8]. One chromosome is said to be fitter than other when all objective values of the former beat those of the latter. If only parts of the objective values of one chromosome are better than the other's, the two chromosomes are deemed equal. A chromosome can be compared with a set of chromosomes. If any individual in the set is fitter than the chromosome, the chromosome is dominated by the set. Otherwise, the chromosome is not dominated by the set.

When two randomly selected chromosomes compete for a spot in the parent pool, they are not directly compared with each other. Rather, each is compared with a comparison set of chromosomes sampled from the current generation. If one of the competing chromosomes, say A, is dominated by the comparison set and the other, say B is not dominated, then B advances to the parent pool. However, when both A and B are either dominated or not by the set, the niche count of each chromosome is compared. The chromosome with smaller niche count gets advantage. Niche count is an indicator of the solution density around a chromosome in a certain solution population. This approach encourages even distribution of solutions in the GA population.

As cross-over is concerned, we adapted one point crossover. It allows different clusters to exchange partial contents and also may split a cluster into two.

Classical mutation operation would modify the value of a gene, one at a time. When the classical GNE is applied to clustering, such mutation causes a reassigning of at most one object only. However, if this approach is to apply to LL encoding, more than just one object could be moved to a different cluster. This classical mutation was tested for LL-encoding and the results show that it does not work well. The solution is to make sure that every time a gene gains link to a different cluster instead of just a different node. In our new mutation operator we also accommodate splitting by allowing the new out-link point to its own cluster, and the cluster splits. The following is the description of our grafting mutation operator:

Grafting Mutation:

1. Select a gene (a node) G_1 at random. Find the ending node E_1 in the same cluster C_1 of G_1.
2. Pick a random number m from 1 to n, where n is the number of clusters.
3. If m = n, make G_1 an ending node. C_1 is thus split into two clusters with exception that the splitting will not occur when G_1 is E_1.
4. If $m < n$, make G_1 ending node. Thus, C_1 is divided into two clusters, ending with G_1 and E_1, respectively. Either link G_1 or E_1 to its m^{th} ending node to its right (loop), and one of the half clusters of C_1 is graft to another cluster.

5 Experimental Results

The Ruspini and Iris data sets have been chosen as test-beds in order to analyze *LE*. The former is an artificial dataset of two attributes; its 75 data points naturally form 4 clusters. The latter is a real dataset recording 150 observations of the three species (setosa, Versicolor, and Virginica) of Iris flowers, all described in four features.

Originally, data points in both datasets are sorted so that the clusters are easily perceived. To obtain unbiased result, we reorganize the datasets and all the data points are randomly placed. In addition, data standardization process is applied to the dataset to neutralize the effects brought by data attributes of disparage magnitude. We divide each data element by the maximum value of its dimension to scale all data elements ranging from 1 to 0.

The proposed new representation is compared with the classic GNE scheme. In our experiments, two GAs are developed. Apart from the encoding schemes, all GA operators are kept the same as described in Section 4. The following genetic parameters are fixed for all GA runs: Number of Experiments is 10, Number of Generations is 2000, Population Size for Iris is 800, Population Size for Ruspini is 500, Niche Comparison Size (selection) is 5, Nitch Radius is 5, Nitch Count Size is 25, Crossover Rate is 0.9, and Mutation Rate is 0.2.

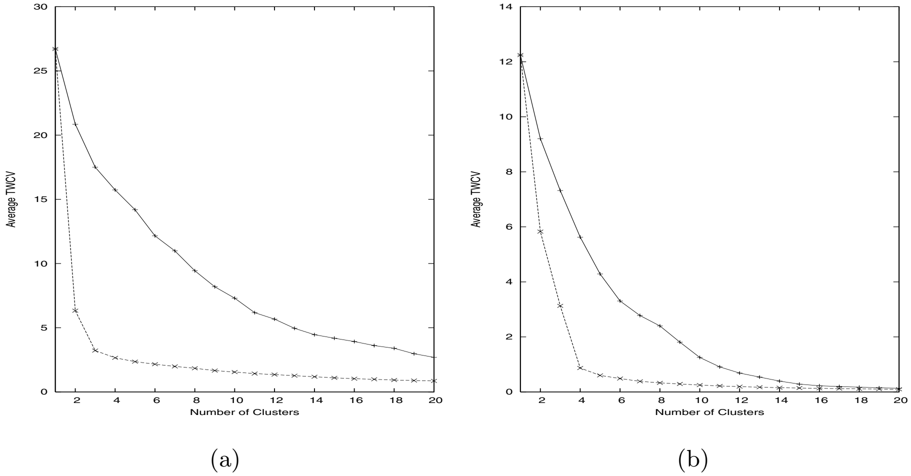
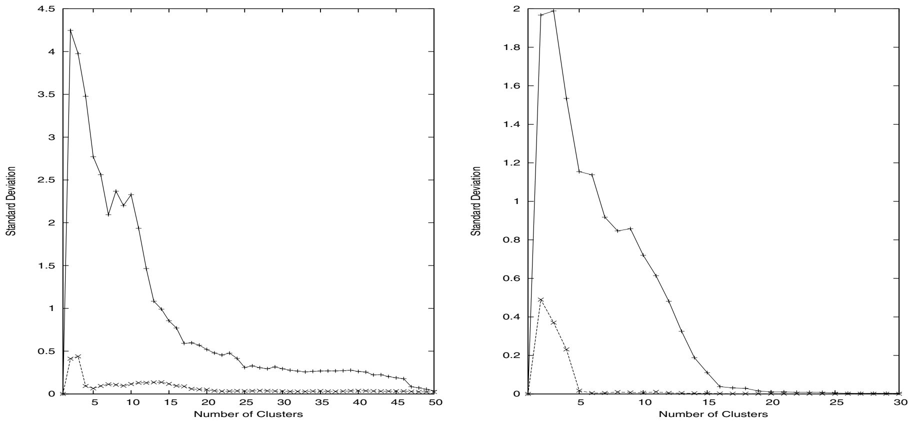


Fig. 1. *LE* versus GNE on a) Iris-data; b) Ruspini-data. The broken lines denote the performance of LE

The comparison of the two methods for the two data sets is presented in Figure 1. The multi-objective GA tries to minimize TWCV for all the possible number of clusters, and reported in Figure 1 are TWCVs obtained by the two methods for up to 20 clusters. Note that the optimal number of clusters is 3 and 4, respectively, for the two data sets. Hence, TWCVs obtained for smaller number of clusters is of more interest. As seen in the two graphs, *LE* clearly dominates the classical approach. More interestingly, TWCVs found by LE rapidly increase for cluster numbers smaller than the optimal clustering. On the other side, it is not possible to observe such a clear distinction for GNE approach presented in the same graphs. Hence, it is possible to derive conclusions about the optimum number of clusters by considering the pareto optimal set obtained at the end of GA search. However, this is possible due to the characteristics of the data sets used. In both data sets the optimum clusters are well separated from each other. In a domain where the cluster borders are not very clear, the leap after



(a) (b)

Fig. 2. Standard deviation of LE and GNE on a) Iris-data; b) Ruspini-data. The broken lines denote the performance of LE

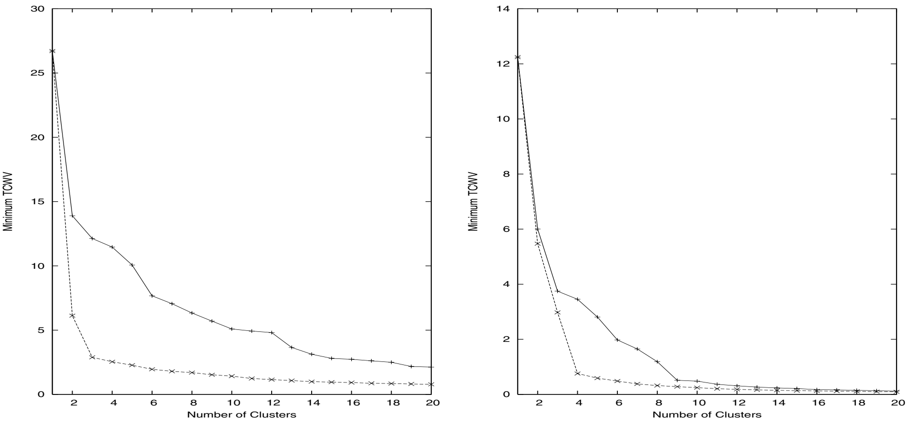
Table 2. CPU Time Statistics for execution of GA using LE on Iris. The other processes item denotes the cost of reading and printing files, statistical computing etc.

	Iris Total CPU Time	Iris %	Ruspini Total CPU Time	Ruspini %
Total Runtime	1h 44m 28.7s	100	24m 35.0m	100
Initialization	5.8s	0.09	1.0s	0.07
Crossover	13.7s	0.22	6.3s	0.43
Mutation	15.9s	0.25	6.3s	0.43
Rectification	3m 25.0s	3.27	44.1s	3.00
Selection	17m 3.5s	16.33	2m 14.7s	9.13
Evaluation	1h 2m 45.7s	60.07	13m 12.7s	53.70
Other processes	20m 23.8s	19.52	7m 59.5s	32.50

the optimum number of clusters may not be as clear as the result obtained in these two domains.

The deviation that appears throughout different runs is another important indicator about GA performance. In Figure 2, the standard deviation of LE and GNE are compared. As seen in the graphs, the deviation with the classical encoding is quite high compared to the new scheme. This is due to the high redundancy in the classical encoding. However, a more consistent output set is obtained with the newly proposed scheme.

Clustering is an off-line process. Therefore, even though an approach is superior to another one in terms of the average output it produces, the success of the method could still be questioned if the two approaches do not differ in terms of the best solution they can find. It is essential to compare the minimum TWCVs obtained by the two schemes. This comparison is presented in Figure 3 for both



(a) (b)

Fig. 3. Minimum TWCVs obtained by LE and GNE on a) Iris-data; b) Ruspini-data. The broken lines denote the performance of LE

data sets and it has been observed that the new encoding is superior in terms of the best solution it can find throughout different experiments.

As described in Section 4, GA operators are updated according to the new encoding scheme. What is more, a rectification process is used in order to decrease the redundancy of the representation. These processes bring in some extra computation cost to the classical execution of GA. If this extra cost increases the initial running cost of GA considerably, then the success of the method would be questionable. Therefore, we have analyzed the average real time CPU statistics of different processes in GA runs using the new scheme. The results for the two data sets are presented in Table 2. Note that the rectification process requires only 3 percent of the total execution time. There are extra computational costs in mutation and crossover operations. However, as seen in the tables their share in the total execution cost is quite small, too. The fitness evaluation is still the most dominant process in terms of computational time. Hence, it can be concluded that the extra execution time required by the new scheme is within acceptable limits.

6 Summary and Conclusions

In this paper, a new encoding scheme is proposed for GA based clustering. This new scheme is successfully used with multi-objective GA. The results obtained on two well-known data sets provide a good insight about the contribution of the new scheme. These results are compared with the output of the classical GNE. The analysis carried out clearly notifies that the new scheme is more advantageous. Although some extra processes are needed in order to keep the redundancy low, it has been observed that the computational cost of these processes is not significant. The leap in TWCV after the optimum number of clusters seems to be an important issue about the technique proposed. It is expected to observe such a leap for datasets with well separated clusters. It would be interesting to observe the change in TWCV, in domains where cluster borders are not clear. In such domains, probably it would not be possible to directly observe the optimum number of clusters. However, an automatic analysis of the change in TWCV might be helpful to determine the optimum point.

Reference

1. P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.
2. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 2000.
3. D.S. Burke and et al. Putting more genetics into genetic algorithms. *Evolutionary Computation*, 6(4):387–410, 1998.
4. E. Falkenauer. A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation*, 2(2):123–144, 1994.
5. E. Falkenauer. *Genetic Algorithms and Grouping Problems.* John Wiley&Sons, 1998.
6. S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *Proc. of ACM SIGMOD*, pages 73–84, New York, 1998.
7. J. Han and M. Kamer. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2001.
8. J. Horn, N. Nafpliotis, and D.E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proc. of IEEE CEC*, volume 1, pages 82–87, 1994.
9. D.A. Jones and M.A. Beltramo. Solving partitioning problems with genetic algorithms. In *Proc. of the International Conference on Genetic Algorithms*, pages 442–449, San Diego, CA, July 1991.
10. K. Krishna and M. Murty. Genetic k-means algorithm. *IEEE TSMC-B*, 29(3):433–439, 1999.
11. U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33:1455–1465, 2000.
12. A. Pen, J. Lozano, and J. Larran. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.
13. E.H. Ruspini. Numerical methods for fuzzy clustering. *Inform. Sci.*, 2(3):19–150, 1970.
14. I. Sarafis, A.M.S. Zalzala, and P. Trinder. A genetic rule-based data clustering toolkit. In *Proc. of the Evolutionary Computation Congress*, pages 1238–1243, 2002.

Categorical Data Visualization and Clustering Using Subjective Factors

Chia-Hui Chang and Zhi-Kai Ding

Department of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320
chia@csie.ncu.edu.tw, sting@tpe.hyweb.com.tw

Abstract. A common issue in cluster analysis is that there is no single correct answer to the number of clusters, since cluster analysis involves human subjective judgement. Interactive visualization is one of the methods where users can decide a proper clustering parameters. In this paper, a new clustering approach called CDCS (Categorical Data Clustering with Subjective factors) is introduced, where a visualization tool for clustered categorical data is developed such that the result of adjusting parameters is instantly reflected. The experiment shows that CDCS generates high quality clusters compared to other typical algorithms.

1 Introduction

Clustering is one of the most useful tasks in data mining process for discovering groups and identifying interesting distributions and patterns in the underlying data. The clustering problem is about partitioning a given data set into groups (clusters) such that the data points in a cluster is more similar to each other than points in different clusters. The clusters thus discovered are then used for describing characteristics of the data set. Cluster analysis has been widely used in numerous applications, including pattern recognition, image processing, land planning, text query interface, market research, etc.

Many clustering methods have been proposed in the literature, and most of these handle data sets with numeric attributes where proximity measure can be defined by geometrical distance. For categorical data which has no order relationship, a general method is to transform it into binary data. However, such binary mapping may lose the meaning of original data set and result in incorrect clustering as reported in [3]. Furthermore, high dimensions will require more space and time if the similarity function is involved with matrix computation such as the Mahalanobis measure.

Another problem we face in clustering is how to validate the clustering results and decide the optimal number of clusters that fits a data set. For a specific application, it may be important to have well separated clusters, while for another it may be more important to consider the compactness of the clusters. Hence, there is no correct answer for the optimal number of clustering since cluster

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 229–238, 2004.

analysis may involve human subjective judgement and Visualization is one of the most intuitive ways for users to decide a proper clustering.

In this paper, we present a new approach, CDCS (Categorical Data Clustering Using Subjective Factors) for clustering categorical data. The central idea in CDCS is to provide a visualization interface to extract users' subjective factors, and therefore to increase the clustering result reliability. CDCS can be divided into three steps. The first step incorporates a single-pass clustering method to group objects with high similarity. Then, small clusters are merged and displayed for visualization. Through the proposed interactive visualization tool, users can observe the data set and determine appropriate parameters for clustering.

The rest of the paper is organized as follows. Section 2 introduces the architecture of CDCS and the clustering algorithm utilized. Section 3 discusses the visualization method of CDCS in detail. Section 4 presents an experimental evaluation of CDCS using popular data sets and comparisons with two famous algorithms AutoClass [2] and k-mode [4]. Section 5 presents the conclusions and suggests future work.

2 The CDCS Algorithm

In this section, we introduce our clustering algorithm which utilizes the concept of Bayesian classifiers as a proximity measure for categorical data and involves a visualization tool of categorical clusters. The process of CDCS can be divided into three steps. In the first step, it applies "simple clustering seeking" [6] to group objects with high similarity. Then, small clusters are merged and displayed by categorical cluster visualization. Users can adjust the merging parameters and view the result through the interactive visualization tool. The process continues until users are satisfied with the result.

Simple cluster seeking, sometimes called dynamic clustering, is a one pass clustering algorithm which does not require the specification of cluster number. Instead, a similarity threshold is used to decide if a data object should be grouped into an existing cluster or form a new cluster. More specifically, the data objects are processed individually and sequentially. The first data object forms a single cluster by itself. Next, each data object is compared to existing clusters. If its similarity with the most similar cluster is greater than a given threshold, this data object is assigned to that cluster and the representation of that cluster is updated. Otherwise, a new cluster is formed. The advantage of dynamic clustering is that it provides simple and incremental clustering where each data sample contributes to changes in the clusters.

However, there are two inherent problems for this dynamic clustering: 1) a clustering result can be affected by the input order of data objects, 2) a similarity threshold should be given as input parameter. For the first problem, higher similarity thresholds can decrease the influence of the data order and ensure that only highly similar data objects are grouped together. As a large number of small clusters (called s-clusters) can be produced, the cluster merging step is required to group small clusters into larger clusters. Another similarity threshold is designed to be adjusted for interactive visualization. Thus, a user's views about the clustering result can be extracted when he/she decides a proper threshold.

2.1 Proximity Measure for Categorical Data

Clustering problem, in some sense, can be viewed as a classification problem for it predicts whether a data object belongs to an existing cluster or class. In other words, data in the same cluster can be considered as having the same class label. Therefore, the similarity function of a data object to a cluster can be represented by the probability that the data object belongs to that cluster. Here, we adopt a similarity function based on the naive Bayesian classifier [5]. The idea of naive Bayes is to computed the largest posteriori probability $\max_j P(C_j|X)$ for a data object X to a cluster C_j. Using the Bayes' theorem, $P(C_j|X)$ can be computed by

$$P(C_j|X) \propto P(X|C_j)P(C_j) \tag{1}$$

Assuming attributes are conditionally independent, we can replace $P(X|C_j)$ by $\prod_{i=1}^{d} P(v_i|C_j)$, where v_i is X's attribute value for the i-th attribute ($X = (v_1, v_2, ..., v_d)$). $P(v_i|C_j)$, a simpler form for $P(A_i = v_i|C_j)$, is the probability of v_i for the i-th attribute in cluster C_j, and $P(C_j)$ is the priori probability defined as the number of objects in C_j to the total number of objects observed.

Applying this idea in dynamic clustering, the proximity measure of a incoming object X_i to an existing cluster C_j can be computed as described above, where the prior objects $X_1, \ldots, X_{i-1}$ before X_i are considered as the training set and objects in the same cluster are regarded as having the same class label. For the cluster C_k with the largest posteriori probability, if the similarity is greater than a threshold g defined as

$$g = p^{d-e} \times \epsilon^e \times P(C_k) \tag{2}$$

then X_i is assigned to cluster C_k and $P(v_i|C_k), i = 1, \ldots, d$, are updated accordingly. For each cluster, a table is maintained to record the pairs of attribute value and their frequency for each attribute. Therefore, to update $P(v_i|C_k)$ is simply an increase of the frequency count. Note that to avoid zero similarity, if a product is zero, we replace it by a small value.

The equation for the similarity threshold is similar to the posteriori probability $P(C_j|X) = \prod_{i=1}^{d} P(v_i|C_j)P(C_j)$, where the symbol p denotes the average proportion of the highest attribute value for each attribute, and e denotes the number of attributes that can be tolerated for various values. For such attributes, the highest proportion of different attribute values is given a small value ϵ. This is based on the idea that the objects in the same cluster should possess the same attribute values for most attributes, even though some attributes may be dissimilar. For large p and small e, we will have many compact s-clusters. In the most extreme situation, where $p = 1$ and $e = 0$, each distinct object is classified to a cluster. CDCS adopts a default value 0.9 and 1 for p and e, respectively. The resulting clusters are usually small, highly condensed and applicable for most data sets.

2.2 Group Merging

In the second step, we group the resulting s-clusters from dynamic clustering into larger clusters ready for display with our visualization tool. To merge s-clusters, we first compute the similarity scores for each cluster pair. The similarity score between two s-clusters C_x and C_y is defined as follows:

$$sim(C_x, C_y) = \prod_{i=1}^{d} [\sum_{j}^{|A_i|} \min\{P(v_{ij}|C_x), P(v_{ij}|C_y)\} + \epsilon] \tag{3}$$

where $P(v_{ij}|C_x)$ denotes the probability of the j-th attribute value for the i-the attribute in cluster C_x, and $|A_i|$ denotes the number of attribute values for the i attribute. The idea behind this definition is that the more the clusters intersect, the more similar they are. If the distribution of attribute values for two clusters is similar, they will have a higher similarity score. There is also a merge threshold g', which is defined as below:

$$g' = (p')^{d-e'} \times \epsilon^{e'} \tag{4}$$

Similar to the last section, the similarity threshold g' is defined by p', the average percentage of common attribute values for an attribute; and e', the number of attributes that can be neglected.

For each cluster pair C_x and C_y, the similarity score is computed and recorded in a $n \times n$ matrix SM, where n is the number of s-clusters. Given the matrix SM and a similarity threshold g', we compute a binary matrix BM (of size $n \times n$) as follows. If $SM[x, y]$ is greater than the similarity threshold g', cluster C_x and C_y are considered similar and $BM[x, y] = 1$. Otherwise, they are dissimilar and $BM[x, y] = 0$. When parameters p' and e' are adjusted, BM is updated accordingly.

With the binary matrix BM, we then apply a transitive concept to group s-clusters. To illustrate this, in Figure 1, clusters 1, 5, and 6 can be grouped in one cluster since clusters 1 and 5 are similar, and clusters 5 and 6 are also similar (The other two clusters are {2} and {3,4}). This merging step requires $O(n^2)$ computation, which is similar to hierarchical clustering. However, the computation is conducted for n s-clusters instead of data objects. In addition, this transitive concept allows the arbitrary shape of clusters to be discovered.

3 Visualization with CDCS

Simply speaking, visualization in CDCS is implemented by transforming a cluster into a graphic line connected by 3D points. These three dimensions represent the attributes, attribute values and the percentages of an attribute value in the cluster. These lines can then be observed in 3D space through rotations. In the following, we first introduce the principle behind our visualization method; and then describe how it can help determine a proper clustering.

	1	2	3	4	5	6
1	1	0	0	0	1	0
2	0	1	0	0	0	0
3	0	0	1	1	0	0
4	0	0	1	1	0	0
5	1	0	0	0	1	1
6	0	0	0	0	1	1

Resulting groups:
{1,5,6},
{2},
{3,4}

Fig. 1. Binary similarity matrix (BM)

3.1 Principle of Visualization

Ideally, each attribute A_i of a cluster C_x has an obvious attribute value $v_{i,k}$ such that the probability of the attribute value in the cluster, $P(A_i = v_{i,k}|C_x)$, is maximum and close to 100%. Therefore, a cluster can be represented by these attribute values. Consider the following coordinate system where the X coordinate axis represents the attributes, the Y-axis represents attribute values corresponding to respective attributes, and the Z-axis represents the probability that an attribute value is in a cluster. Note that for different attributes, the Y-axis represents different attribute value sets. In this coordinate system, we can denote a cluster by a list of d 3D coordinates, $(i, v_{i,k}, P(v_{i,k}|C_x)), i = 1, \ldots, d$, where d denotes the number of attributes in the data set. Connecting these d points, we get a graphic line in 3D. Different clusters can then be displayed in 3D space to observe their closeness.

This method, which presents only attribute values with the highest proportions, simplifies the visualization of a cluster. Through operations like rotation or up/down movement, users can then observe the closeness of s-clusters from various angles and decide whether or not they should be grouped in one cluster. Graphic presentation can convey more information than words can describe. Users can obtain reliable thresholds for clustering since the effects of various thresholds can be directly observed in the interface.

3.2 Building a Coordinate System

To display a set of s-clusters in a space, we need to construct a coordinate system such that interference among lines (different s-clusters) can be minimized in order to observe closeness. The procedure is as follows. First, we examine the attribute value with the highest proportion for each cluster. Then, summarize the number of distinct attribute values for each attribute, and then sort them in increasing order. Attributes with the same number of distinct attribute values are further ordered by the lowest value of their proportions. The attribute with the least number of attribute values are arranged in the middle of the X-axis and others are put at two ends according to the order described above. In other words, if the attribute values with the highest proportion for all s-clusters are the same for some attribute A_k, this attribute will be arranged in the middle of the X-axis. The next two attributes are then arranged at the left and right of A_k.

After the locations of attributes on the X-axis are decided, the locations of the corresponding attribute values on the Y-axis are arranged accordingly. For each s-cluster, we examine the attribute value with the highest proportion for each attribute. If the attribute value has not been seen before, it is added to the "presenting list" (initially empty) for that attribute. Each attribute value in the presenting list has a location as its order in the list. That is, not every attribute value has a location on the Y-axis. Only attribute values with the highest proportion for some clusters have corresponding locations on the Y-axis. Finally, we represent a s-cluster C_k by its d coordinates $(L_x(i), L_y(v_{i,j}), P(v_{i,j}|C_k))$ for $i = 1, \ldots, d$, where the function $L_x(i)$ returns the X-coordinate for attribute A_i, and $L_y(v)$ returns the Y-coordinate for attribute value v.

	A1	A2	A3	A4	A5	A6	A7	A8
s1	a 90%	P 100%	S 40%	x 60%	M 100%	α 100%	G 100%	B 99%
	b 10%		F 30%	q 40%				C 1%
			D 30%					
s2	a 100%	O 80%	S 100%	x 40%	M 100%	α 100%	H 100%	B 100%
		P 20%		q 30%				
				z 30%				

(a) Two s-clusters and their distribution table

A2	A4	A1	A6	A5	A8	A3	A7

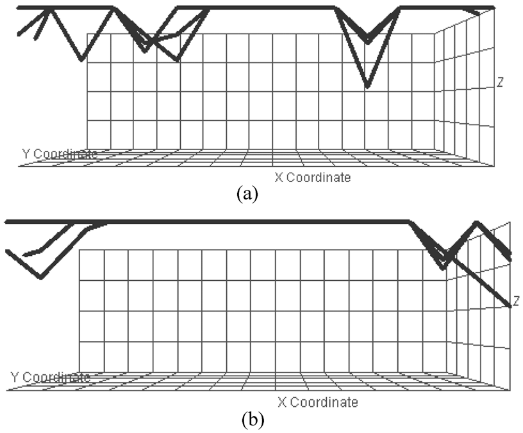
(b) Rearranged X coordinate

s1	(1, 1, 1)	(2, 1, 0.6)	(3, 1, 0.9)	(4, 1, 1.0)	(5, 1, 1.0)	(6, 1,0.9)	(7, 1, 0.4)	(8, 1, 1.0)
s2	(1, 2, 0.8)	(2, 1, 0.4)	(3, 1, 1.0)	(4, 1, 1.0)	(5, 1, 1.0)	(6, 1, 1.0)	(7, 1, 1.0)	(8, 2, 1.0)

(c) 3-D coordinates for s1 and s2

Fig. 2. Example of constructing a coordinate system

In Figure 2 for example, two s-clusters and their attribute distributions are shown in (a). Here, the number of distinct attribute values with the highest proportion is 1 for all attributes except for A_2 and A_7. For these attributes, they are further ordered by the lowest proportion. Therefore, the order for these 8 attributes are $A_5, A_6, A_8, A_1, A_3, A_4, A_7, A_2$. With A_5 as center, A_6 and A_8 are arranged to the left and right, respectively. The rearranged order of attributes is shown in Figure 2(b). Finally, we transform cluster s_1, and then s_2 into the coordinate system we build, as shown in Figure 2(c). Taking A_2 for example, the presenting list includes P and O. Therefore, P gets a location 1 and O a location 2 at Y-axis. Similarly, G gets a location 1 and H a location 2 at Y-axis for A_7.



(a)

(b)

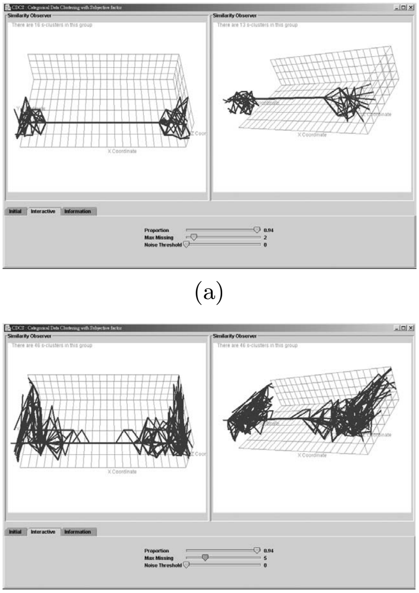
Fig. 3. Three s-clusters (a) before and (b) after attribute rearrangement

Figure 3 shows an example of three s-clusters displayed in one window before (a) and after (b) the attribute rearrangement. The thickness of lines reflects the size of the s-clusters. Comparing to the coordinate system without rearranging attributes, s-clusters are easier to observe in the new coordinate system since common points are located at the center along the X-axis presenting a trunk for the displayed s-clusters. For dissimilar s-clusters, there will be a small number of common points, leading to a short trunk. This is an indicator whether the displayed clusters are similar and this concept will be used in the interactive analysis described next.

3.3 Interactive Visualization and Analysis

The CDCS's interface, as described above, is designed to display the merging result of s-clusters such that users know the effects of adjusting parameters. Instead of showing all s-clusters, our visualization tool displays only two groups from the merging result. More specifically, our visualization tool presents two groups in two windows for observing. The first window displays the group with the most number of s-clusters since this group is usually the most complicated case. The second window displays the group which contains the cluster pair with the lowest similarity. The coordinate systems for the two groups are conducted respectively.

Figure 4 shows an example of the CDCS's interface. The dataset used is the Mushroom database taken from UCI [1]. The number of s-clusters obtained from the first step is 106. The left window shows the group with the largest number of s-clusters, while the right window shows the group with the least similar s-cluster pair. The number of s-clusters for these groups are 16 and 13, respectively, as shown at the top of the windows. Below these two windows, three sliders are used to control the parameters for group merging and visualization. The first two sliders denote the parameters p' and e' used to control similarity



There are 16 s-clusters in this group

There are 13 s-clusters in this group

Proportion 0.94
Max Missing 2
Noise Threshold 0

(a)

There are 46 s-clusters in this group

There are 46 s-clusters in this group

Proportion 0.94
Max Missing 5
Noise Threshold 0

(b)

Fig. 4. Visualization of the mushroom dataset (a) a strict threshold (b) a mild threshold

threshold g'. The third slider is used for noise control in the visualization so that small s-clusters can be omitted to highlight the visualization of larger s-clusters. Each time the slider is moved, the merging result is computed and updated in the windows. Users can also lock one of the windows for comparison with different threshold.

A typical process for interactive visualization analysis with CDCS is as follows. We start from a strict threshold g' such that the displayed groups are compact; and then relax the similarity threshold until the displayed groups are too complex. A compact group usually contains a long trunk such that all s-clusters in the group have same values and high proportions for these attributes. A complex group, on the other hand, presents short trunk and contains different values for many attributes. For example, both groups displayed in Figure 4(a) have obvious trunks which are composed of sixteen common points (or attribute values). For a total of 22 attributes, 70% of the attributes have the common values and proportions for all s-clusters in the group. Furthermore, the proportions of these attribute values are very high. Through rotation, we also find that the highest proportion of the attributes outside the trunk is similarly low for all s-clusters. This implies that these attributes are not common features for these s-clusters. Therefore, we could say both these groups are very compact since these groups are composed of s-clusters that are very similar.

If we relax the parameter e' from 2 to 5, the largest group and the group with least similar s-clusters are the same group which contains 46 s-clusters, as shown in Figure 4(b). For this merging threshold, there is no obvious trunk for this group and some of the highest proportions outside the trunk are relatively high while some are relatively low. In other words, there are no common features for these s-clusters, and this merge threshold is too relaxed since different s-clusters are put in the same group. Therefore, the merging threshold in Figure 4(a) is better than the one in Figure 4(b).

In summary, whether the s-clusters in a group are similar is based on users' viewpoints on the obvious trunk. As the merging threshold is relaxed, more s-clusters are grouped together and the trunks of both windows get shorter. Sometimes, we may reach a stage where the merge result is the same no matter how the parameters are adjusted. This may be an indicator of a suitable clustering result. However, it depends on how we view these clusters since there may be several such stages.

4 Experiments

We presents an experimental evaluation of CDCS on five real-life data sets from UCI machine learning repository [1] and compares its result with AutoClass [2] and k-mode [4]. The number of clusters required for k-mode is obtained from the clustering result of CDCS. To study the effect due to the order of input data, each dataset is randomly ordered to create four test data sets for CDCS. Four users are involved in the visualization analysis to decide a proper grouping criteria.

The five data sets used are Mushroom, Soybean-small, Soybean-large, Zoo dataset and Congress Voting which have been used for other clustering algorithms. Table 1 records the number of clusters and the clustering accuracy for the 5 data sets. As shown in the last row, CDCS has better clustering accuracy than the other two algorithms. CDCS is better than K-mode in each experiment given the same number of clusters. However, the number of clusters found by CDCS is more than that found by AutoClass, especially for the last two data sets. The main reason for this phenomenon is that CDCS reflects users' view on the degree of intracluster cohesion. Various clustering results, say 9 clusters and 10 clusters, can not be observed in this visualization method. In general, CDCS has better intracluster cohesion for all data sets, while AutoClass has better cluster separation (smaller intercluster similarity) on the whole.

5 Conclusion

In this paper, we introduced a novel approach for clustering categorical data based on two ideas. First, a classification-based concept is incorporated in the computation of object similarity to clusters; and second, a visualization method is devised for presenting categorical data in a 3-D space. Through interactive visualization interface, users can easily decide a proper parameter setting. From the experiments, we conclude that CDCS performs quite well compared to state-of-the-art clustering algorithms. Meanwhile, CDCS handles successfully data sets with significant differences in the sizes of clusters. In addition, the adoption of

Table 1. Number of clusters and clustering accuracy for three algorithms

	# of clusters		Accuracy		
	AutoClass	CDCS	AutoClass	K-mode	CDCS
Mushroom	22	21	0.9990	0.9326	0.996
22 attributes	18	23	0.9931	0.9475	1.0
8124 data	17	23	0.9763	0.9429	0.996
2 labels	19	22	0.9901	0.9468	0.996
Zoo	7	7	0.9306	0.8634	0.9306
16 attributes	7	8	0.9306	0.8614	0.9306
101 data	7	8	0.9306	0.8644	0.9306
7 labels	7	9	0.9207	0.8832	0.9603
Soybean-small	5	6	1.0	0.9659	0.9787
21 attributes	5	5	1.0	0.9361	0.9787
47 data	4	5	1.0	0.9417	0.9574
4 labels	6	7	1.0	0.9851	1.0
Soybean-large	15	24	0.664	0.6351	0.7500
35 attributes	5	28	0.361	0.6983	0.7480
307 data	5	23	0.3224	0.6716	0.7335
19 labels	5	21	0.3876	0.6433	0.7325
Voting	5	24	0.8965	0.9260	0.9858
16 attributes	5	28	0.8942	0.9255	0.9937
435 data	5	26	0.8804	0.9312	0.9860
2 labels	5	26	0.9034	0.9308	0.9364
Average			0.8490	0.8716	0.9260

naive-Bayes classification makes CDCS's clustering results much more easily interpreted for conceptual clustering.

Acknowledgements

This work is sponsored by National Science Council, Taiwan under grant NSC92-2213-E-008-028.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases. http://www.cs.uci.edu/~mlearn/MLRepository.html. Irvine, CA., 1998.
2. P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In *Proceedings of Advances in Knowledge Discovery and Data Mining*, pages 153–180, 1996.
3. S. Guha, R. Rastogi, and K. Shim. ROCK: a robust clustering algorithm for categorical attributes. *Information Systems*, 25:345–366, 2000.
4. Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2:283–304, 1998.
5. T. M. Mitchell. *Machine Learning.* McGraw Hill, 1997.
6. J. T. To and R. C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesley Publishing Company, 1974.

Multidimensional Data Visual Exploration by Interactive Information Segments

Francisco J. Ferrer-Troyano, Jesús S. Aguilar-Ruiz, and José C. Riquelme

Department of Computer Science, University of Seville
Avenida Reina Mercedes s/n, 41012 Seville, Spain
{ferrer,aguilar,riquelme}@lsi.us.es

Abstract. Visualization techniques provide an outstanding role in KDD process for data analysis and mining. However, one image does not always convey successfully the inherent information from high dimensionality, very large databases. In this paper we introduce VSIS (Visual Set of Information Segments), an interactive tool to visually explore multidimensional, very large, numerical data. Within the supervised learning, our proposal approaches the problem of classification by searching of meaningful intervals belonging to the most relevant attributes. These intervals are displayed as multi–colored bars in which the degree of impurity with respect to the class membership can be easily perceived. Such bars can be re–explored interactively with new values of user–defined parameters. A case study of applying VSIS to some UCI repository data sets shows the usefulness of our tool in supporting the exploration of multidimensional and very large data.

1 Introduction

Visualization techniques provide an important support to extract knowledge from huge amounts of data incorporating ingenuity, analytic capability, and experience of the user in order to steer the KDD process [15]. From graphic representations of a query or data set, the user carries out an interactive visual exploration from which interesting subsets and data relationships can be identified, and new hypotheses and conclusions can be drawn. Such hypotheses can be later verified by data mining techniques. Through a visual exploration, the user can intuitively have a good idea of the result interpretation. Different graphic views of the same data set can give the user a better understanding about it, and an easy way to detect patterns, outliers, and noise. In addition, visualization tools can be also used to reduce the search space and therefore, to obtain simpler models for complex sub–domains.

An important concern for multidimensional data visualization techniques is to avoid different entities overlapping on the screen. These individual entities can be data–items or examples, data–values or attribute–values, or data aggregations based on the former ones. If the values are directly displayed, they usually are a significantly small portion of the entire available data. Otherwise, it is likely that the resulting image cannot convey the data properties appropriately and the

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 239–248, 2004.

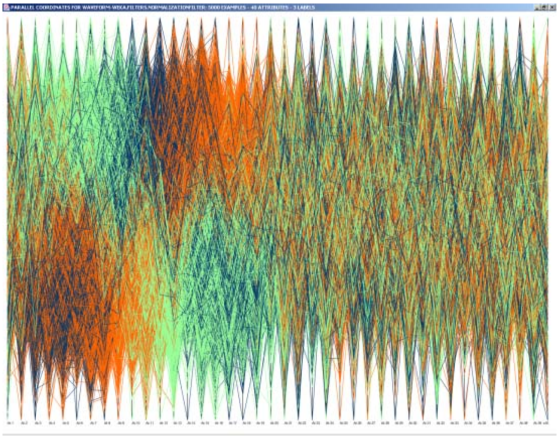

PARALLEL COORDINATES FOR WAVEFORM-WEKA.FILTERS.NORMALIZATIONFILTER: 5000 EXAMPLES - 40 ATTRIBUTES - 3 LABELS

Fig. 1. Wave–Form database (40 attributes, 5000 examples, 3 class labels) in Parallel Coordinates.

exploration becomes a difficult task. As an example, Figure 1 shows the Wave–form data set displayed using the well–known Parallel Coordinates technique [11]. Because of the high width and depth of this data set, individual examples cannot be clearly seen from this display, also preventing the detection of relevant patterns and attributes. We think it is more interesting to display as few graphic entities as possible in order to represent as large amount of data as possible. The smaller number of graphical entities containing higher number of examples, the easier and more meaningful interpretation of results.

In addition, many visualization techniques have restrictions regarding the data size, with respect either to the number of examples or the number of attributes. In this paper we introduce VSIS (Visual Set of Information Segments), an interactive tool to explore multidimensional and very large databases. Handling enormous amount of data might seem risky to graphically represent every different value in only one image, not only because of the screen limitations, but also the human ability to understand a complex image. Therefore, our goal is also to incorporate user's constraints, so the display can become more significant for the expert.

2 VSIS: Visual Set of Information Segments

Within the supervised learning, the problem of classification is generally defined as follows. An input finite data set of training examples is given. Every training example is a pair $e = (x, y)$ where x is a vector of m attribute values (each of which may be numeric or symbolic) and y is a class discrete value named label. The goal is to obtain a model $y = f(x)$ to classify or decide the label for new non–labelled test examples named queries. VSIS supports the problem of classification with numerical attributes by displaying only the most relevant attributes with only the most meaningful intervals. The set of intervals is as small as possible depending on the user demand. For each interval is displayed the distribution of labels within it and the relationship with other intervals. We name these graphic entities information segments.

Henceforth, the next notation is used to describe VSIS. Let m be the number of continuous attributes $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$. Let $Y = \{y_1, \ldots, y_z\}$ be the set of class labels from one nominal attribute previously selected by the user. Let $\mathcal{T}$ be the training set so that: $\mathcal{T} = \{e_1, \ldots, e_n\}$; $e_i = (x_i, y_i)$; $x_i \in \mathcal{R}^m$; $y_i \in Y; i \in \{1, \ldots, n\}$; $m, n \in \mathcal{N}$.

Definition 1 (Empty Segment) *An empty segment $S_{j,k}$ represents an interval I of the j^{th} attribute $\mathcal{A}_j$ for which no training example has a value within I: $\forall\ e_i \in \mathcal{T} \cdot x_{ij} \notin I$.*

Definition 2 (Pure Segment) *A pure segment $S_{j,k}$ represents an interval I of the j^{th} attribute $\mathcal{A}_j$ for which all the training examples are associated with the same class label: $\nexists\ e_i,\ e_{i'} \in \mathcal{T} \cdot x_{ij} \in I \wedge x_{i'j} \in I \wedge y_i \neq y_{i'}$.*

Definition 3 (Impure Segment) *An impure segment $S_{j,k}$ represents an interval I of the j^{th} attribute $\mathcal{A}_j$ for which there are training examples associated with different class labels: $\exists\ e_i,\ e_{i'} \in \mathcal{T} \cdot x_{ij} \in I \wedge x_{i'j} \in I \wedge y_i \neq y_{i'}$.*

The segments are displayed as colored bars. The color represents the class label and it is previously selected by the user. Empty segments are not displayed, pure segments are displayed with one color and impure segments are displayed with a number of colors. Every color takes up an area inside a rectangle which is proportional to the number of examples with the label associated to such a color in the respective information segment.

The process is divided into two steps: first, an initial set of segments is calculated; second, the minimal set of segments meaningful for the user is obtained. Both sets are displayed and can be interactively re–explored.

2.1 ISIS: Initial Sets of Information Segments

This first phase builds m initial sets $ISIS_j$ $(j \in \{1, \ldots, m\})$, one per attribute. Each set $ISIS_j$ is formed by α information segments and provide the user with insight about the label distribution of input data. α is a user parameter (integer) which splits the continuous attributes of $\mathcal{T}$ into α equal–width intervals. The higher value for α, the higher accuracy is obtained.

Each information segment $S_{j,k}$ $(k \in \{1, \ldots, \alpha\})$ is composed by three elements:

- $I_{j,k} = [l_{j,k}, u_{j,k})$ is a left–closed, right–open interval in $\mathcal{R}$, such that $u_{j,k} = l_{j,k+1}$ $(\forall k < \alpha)$.
- $H_{j,k} = \{H_{j,k^1}, \ldots, H_{j,k^z}\}$ is a histogram with the number of examples for each label that are covered by $S_{j,k}$. An example e_i is covered by a segment $S_{j,k}$ if the j^{th} attribute value of the example (x_{ij}) belongs to the interval $I_{j,k}$.

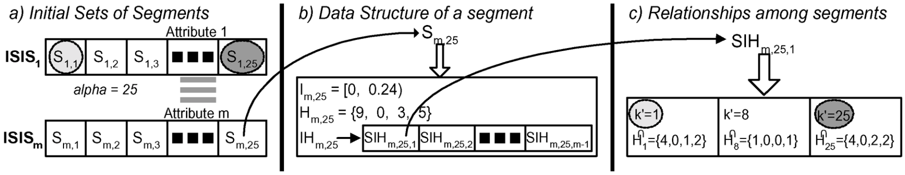


Fig. 2. Diagram of the data structure used to build the initial sets of information segments ($\alpha = 25$).

- $IH_{j,k}$ is a set of $m-1$ elements $SIH_{j,k,j'}$, one per each attribute $\mathcal{A}_{j'}$ different to $\mathcal{A}_j$. Each element $SIH_{j,k,j'}$ is composed by a set of pairs $(k', H^{\cap}_{k'})$, related to segments for other attributes containing examples covered by S_{jk}. The element k' is the index of a segment $S_{j',k'}$, and $H^{\cap}_{k'}$ is the histogram of class labels for examples in the intersection $H_{j,k} \cap H_{j',k'}$. The purpose of this data structure is to compute the minimal set of segments in the next phase.

The initial sets of segments are built by one only scan, previously generating α empty segments $S_{j,k}$ for each attribute with $H_{j,k^p} = 0$ ($p \in \{1, \ldots, z\}$) and $IH_{jk} = \emptyset$. Then every example $e_i = (x_i, y_i)$ updates the class–labels histogram H_{jk} of the segment S_{jk} that covers x_i (increasing by one $H_{jk^{y_i}}$), and the relationships IH_{jk} among such updated segments. The computational cost of the process is not expensive since the index k of the segment S_{jk} associated to a value x_{ij} can be calculated directly:

$$k = \lfloor norm(x_{ij}) \cdot \alpha \rfloor;\ norm(x_{ij}) = \frac{x_{ij} - MIN_j}{MAX_j - MIN_j};\ MIN_j = l_{j1};\ MAX_j = u_{j\alpha}$$

Figure 2 shows a diagram of the data structure used to build the initial set of information segments, using $\alpha = 25$ initial intervals and 4 class labels ($Y = \{A, B, C, D\}$). For each attribute $\mathcal{A}_j$, each $ISIS_j$ has 25 equal–width segments. The last segment of the last attribute ($S_{m,25}$) is associated to the real interval $[0, 0.24]$. This interval covers 17 examples, 9 of them with label A, 3 with label C, and 5 with label D. These 17 examples are covered by three segments in the attribute 1 ($S_{1,1}$,$S_{1,8}$ and $S_{1,25}$). The first segment, $S_{1,1}$, covers 7 examples (4 with class A, 0 with class B, 1 with class C and 2 with class D), 2 the second and 8 the third one.

When all the examples have been processed, all the empty segments are removed. Next, every pair of consecutive segments with equal label distribution is joined ($\frac{|H_{j,k-1^p}|}{\sum_{p=1}^{z} |H_{j,k-1^p}|} = \frac{|H_{j,k^p}|}{\sum_{p=1}^{z} |H_{j,k}|}, \forall p \in \{1, \ldots, z\}$). Given $S_{j,k-1}$ and $S_{j,k}$ as consecutive segments, $H_{j,k-1}$ and $IH_{j,k-1}$ are updated with $H_{j,k}$ and $IH_{j,k}$, respectively, and the right segment $S_{j,k}$ is removed. Finally, when all the attributes have been examined, a ranking of them is obtained as a function of the number of pure segments, the number of impure ones, and the impurity level of them, by means of the next heuristic:

$$Weight(\mathcal{A}_j) = n^{-ns_j} \sum_{k=1}^{ns_j} (\max_{p=1}^{z} |H_{j,k^p}|)$$

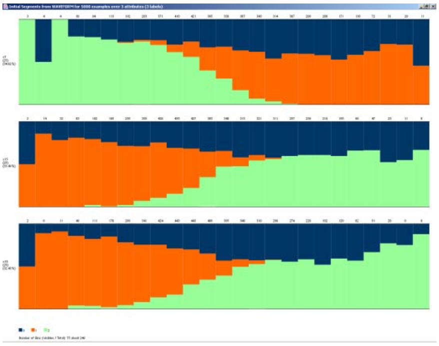


Fig. 3. Wave–Form database. Initial segments for the three most significant attributes (x7, x15, and x16), using $\alpha = 25$. There are not empty segments in any attribute.

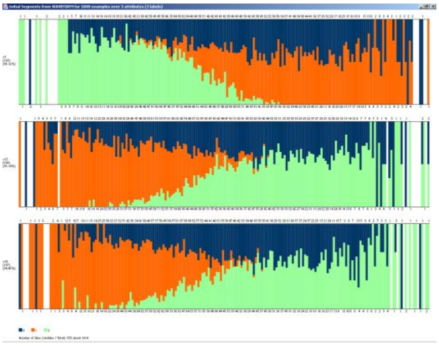


Fig. 4. Wave–Form database. Initial segments using $\alpha = 200$. There are empty segments in every attribute: x7 (32 segments), x15 (25), and x16 (29).

where n is the number of training examples, z is the number of different class labels and ns_j is the number of non–empty initial segments in $\mathcal{A}_j$.

Figures 3 and 4 show the initial segments obtained from Wave-form data set, for the three most significant attributes according to the above heuristic: x7, x15 and x16. Attributes are graphically shown in order of relevance, from top to bottom. The display shows similarities among attributes -when two or more of them have similar shapes- and their relevance -when the distribution is homogeneous, that is, intervals are impure-. The higher value for α, the greater number of empty intervals are displayed. That is the reason why in Figure 4 the number of initial segments are not equal to the number of initial intervals. Having a look to the images, we can know the label distribution and the overlap level inside the attributes. It is interesting the correlation between attributes x15 and x16 (two at the bottom of figure on the left). That correlation is stronger when the class *orange* is present, and we can observe that shapes for class *green* are different for these two attributes from the middle of the attribute until the right bound. In addition, the initial segments provide the user with an insight about the potential complexity of the *Minimal Set of Information Segments.*

2.2 MSIS: Minimal Set of Information Segments

In the second phase, consecutive segments belonging to the attributes selected in the first phase are joined, trying to take advantage of attributes with least number of segments and smaller intersection among them. The goal now is to find the least number of segments from which to describe the label distribution, transforming thousands of examples with dozens of attributes into several colored segments clearly separated in the image.

Definition 4 (Support) *The support of a segment $S_{j,k}$ is the number of examples covered by $S_{j,k}$.*

Definition 5 (Purity) *The purity of a segment $S_{j,k}$ is the percentage of examples covered by $S_{j,k}$ with a majority label with respect to its support.*

Definition 6 (Minimal Support γ) *The minimal support γ is the lowest support that a segment must surpass to belong to the MSIS.*

Definition 7 (Minimal Purity δ) *The minimal purity δ is the lowest percentage of examples with a majority label with respect to the number of covered examples that an impure segment must surpass for to it be a part of the MSIS.*

Therefore, a pure segment $S_{j,k}$ takes $\delta_{j,k} = 100$ whereas an impure segment $S_{j',k'}$ takes $\delta_{j',k'} = 100\frac{\max_{p=1}^{z}(|H_{j',k'^p}|)}{\sum_{p=1}^{z}|H_{j',k'^p}|}$. γ and δ are two user parameters.

The MSIS is built from the $ISIS_j$ sets by two iterative procedures. For each attribute $\mathcal{A}_j$, the algorithm looks for the two consecutive impure segments in $ISIS_j$ whose union is possible and whose resulting support is the highest. Two consecutive impure segments can be joined if the resulting purity is greater than or equal to the minimal purity δ.

Next, another iterative procedure adds joined segments from the $ISIS_j$ sets to MSIS. In each iteration, a new segment is included in the MSIS: the one with the largest number of examples that are not yet covered by another segments already included in the MSIS. Thus, the first segment to be included will be the one with the highest support. The procedure ends when either all the examples have been covered or there is no segment that covers examples uncovered by the MSIS. The number Δ of examples that a segment $S_{j,k}$ in $ISIS_j$ can provide for the MSIS is computed by the intersection among $IH_{j,k}$ and the histograms $H^{\bigcap}_{k'}$ associated with the segments $S_{j',k'}$ already included in the MSIS, according to the equation 1:

$$\Delta = (\sum_{p=1}^{z} |H_{j,k^p}|) - | \bigcap_{\forall\ S_{j'k'} \in\ MSIS} (SIH_{j,k,j'}, H_{j',k'}) | \qquad (1)$$

In each new iteration, the number of examples uncovered by non–included segments may change with respect to the earlier iteration, and every segment is re–visited again. If a segment $S_{j,k} \in ISIS_j$ does not contain examples uncovered by MSIS, then it is removed from $ISIS_j$ so that the next iteration will have lower computational cost.

When the above procedure ends, the MSIS is displayed. For each attribute with at least one information segment, a horizontal attribute–bar shows its segments in increasing order of values, from left to right. Every attribute–bar is equal in size, both in width and height. To the left of each bar, the name of the associated attribute is displayed, along with the total number of examples covered by the segments belonging to such an attribute, and the total number of exclusive examples that all the segments provide. We allow interactive capability to VSIS tool so that the user can keep on exploring the examples belonging to impure segments (both in ISIS and MSIS) until finding a meaningful visual description, through both Parallel Coordinates technique and new segments over different dimensions with higher purity.

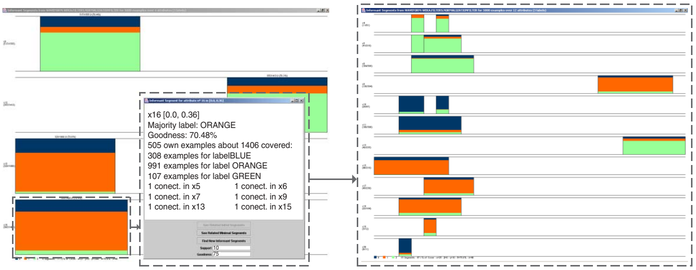

x16 [0.0, 0.36]
Majority label: ORANGE
Goodness: 70.48%
505 own examples about 1406 covered:
308 examples for labelBLUE
991 examples for label ORANGE
107 examples for label GREEN
1 conect. in x5 1 conect. in x6
1 conect. in x7 1 conect. in x9
1 conect. in x13 1 conect. in x15

Fig. 5. Wave–Form data set. To the left, first exploration level using α=25, γ=1000 and δ=70. To the right, second exploration level after selecting one segment from the image to the left, using α=25, γ=10 and δ=75.

An example of this capability of VSIS is shown in Figure 5. To the left, we try to get more insight from Figure 3. To select the most significant segments we set γ=1000, so only segments containing at least 1000 examples will be displayed. In addition, as the label distribution provides many impure segments, we will relax the criterion by setting δ=70, so only segments containing at least 70% of examples with the same label must be displayed. Only four attributes were selected by the algorithm (x6, x13, x15 and x16), and one segment for each one. Attribute x7 appeared in Figures 3 and 4, but not in Figure 5, because the new intervals offer fewer intervals with better label distribution.

Now we are interested in the last attribute x16 (bottom), so we can click on it and set new values for δ and γ. This means we are going to analyze only that segment, and therefore, the examples covered by it (exploration level = 2). The new values for δ and γ are 75 and 10, respectively. The result is shown on the right, where more segments are displayed due to the reduction of the value of γ (only 10 examples). However, the purity has been increased up to 75. New attributes appear in this image (x4, x5, x6, x9, x10, x11, x13, x15, x17, x18, x19 and x38), and also new segments (some of them were already in the exploration level =1), which provide more insight about the data, as this might mean that x6 and x13 are decisive for class *green* and x15 for class *orange*.

Re–exploring impure segments gives a powerful insight of data, as we can achieve a higher accuracy on a specific domain. In this way VSIS cedes the control to the user in order to find, group and validate decision rules with the detail level needed. When a segment is explored, the new segments represent sub–domains satisfying new user specifications. Each new exploration level gives a new description of one attribute condition in a decision rule, since when we decide to explore a segment, the subset defined by that attribute condition is visualized. The process is completely interactive, reducing the support and increasing the accuracy every time.

3 Displaying Very Large Databases

We have selected two databases from the UCI repository [5] to show the usefulness of our tool for visualizing great amount of data: one with large number of

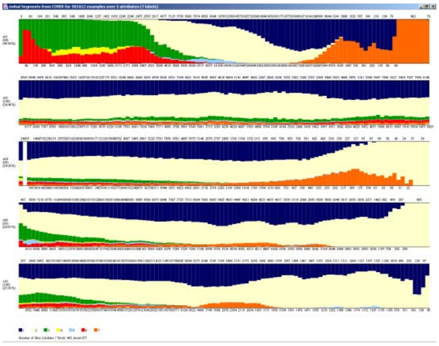

Initial Segments from COVER for 581012 examples over 5 attributes (7 labels)

Fig. 6. Covtype database (581012 examples, 54 attributes, and 7 class labels). Initial segments for the best five attributes according to ranking: a01, a12, a14, a36, a37 (α=50).

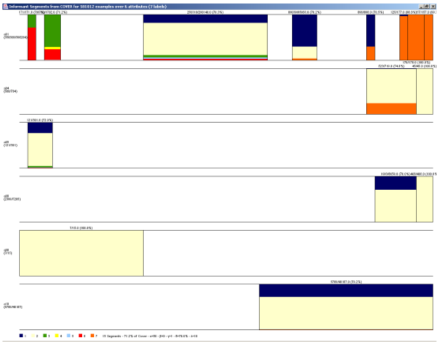

Informant Segments from COVER for 581012 examples over 6 attributes (7 labels)

Fig. 7. Covtype database: MSIS using $\alpha = 50$, $\gamma = 1000$ and $\delta = 70$. Fifteen information segments (5 pure segments and 10 impure segments).

examples (Covtype) and another one with large number of attributes (Isolet). Figures 6 and 7 show two visualization examples of Covtype database. Such displays represent the manner to explore and detect significant sub–domains and irrelevant attributes, and to validate patterns or rules extracted by learning algorithms. These displays also give a good estimate with respect to accuracy and complexity of the model to be extracted by a learning algorithm.

In Isolet database, the high degree of overlapping among different class labels (most of segments are impure with a very low purity) shows the difficulty to obtain both a non–complex and accurate knowledge model for (Figures 8 and 9). To tackle 617 attributes and 26 class labels is computationally expensive for many visualization techniques, however VSIS performance is satisfactory.

4 Related Work

In [15], Information Visualization and Visual Data Mining techniques are classified according to three criteria:

- The data type to be visualized: *one–dimensional data* [19], *two–dimensional data* [20], *multidimensional data* [16], *text & hypertext* [19], *hierarchies & graphs* [6,4], and *algorithms & software* [8].
- The data representation: *standard 2D/3D displays* [20], *geometrically transformed displays* [1,10,11], *icon–based displays* [7], *dense pixel displays* [14], *stacked displays* [12], and hybrid techniques.
- The user interaction way: *projection* [3], *filtering* [20], *zooming* [17], *spherical & hyperbolic distortions*, and *linking & brushing*.

According to the data type, our proposal VSIS can visualize multidimensional data sets with numerical attributes. With respect to the second group, VSIS

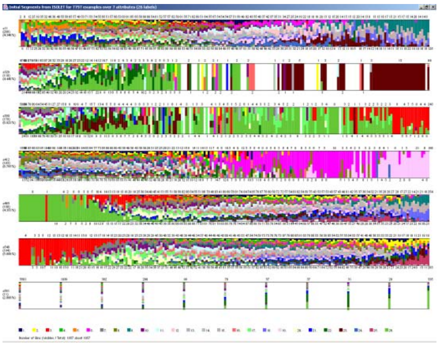

Initial Segments from ISOLET for 7797 examples over 7 attributes (26 labels)

Fig. 8. Isolet database (7797 examples, 617 continuous attributes, and 26 class labels). Initial segments for the best seven attributes using $\alpha = 200$.

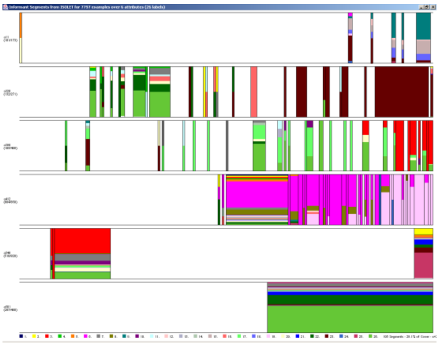

Informant Segments from ISOLET for 7797 examples over 6 attributes (26 labels)

Fig. 9. Isolet database: MSIS using $\alpha = 200$, $\gamma = 10$ and $\delta = 50$.

belongs to standard 2D techniques. Regarding the third category, VSIS provides data projections to the user, zooming and filtering to detect and validate relevant and meaningful attributes and subdomains. Dimensionality reduction has been dealt by three major approaches: Principal Component Analysis (PCA) [13], Multidimensional Scaling (MDS) [18], and Kohonen's Self Organizing Maps (SOM) [9]. Recently, new dimensionality reduction techniques have been proposed to process very large data sets with high dimensionality [2]. VSIS reduces the dimensionality in an interactive manner so as to find meaningful subdomains according to user measures.

5 Conclusions and Future Work

VSIS is a visualization tool to explore multidimensional numerical data. Through visual interaction and feedback, the user decides how many examples and what level of purity a segment must fulfil to be considered representative of a significant subdomain. The information segments can be *seen* as decision rules with a number of disjunctions equals the number of exploration levels, whose support is greater than or equal to the last value of γ, and whose purity is greater than or equal to the last value of δ. This representation helps the user with the identification of the most relevant attributes. Results are very interesting as the tool is very flexible and allows the user to go into the level of exploration needed.

We are currently improving VSIS by using dense pixel approach to display segments with variable color intensity degree in such a way the support of the segments can be easily perceived. Several similarity measures are being addressed to re–order segments on the display regarding the number of shared examples, making the graphical information understandability easier.

References

1. D. F. Andrews. Plots of high–dimensional data. *Biometrics*, 29:125–136, 1972.
2. M. Ankerst, S. Berchtold, and D.A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *InfoVis'98*, pages 52–60, 1998.
3. D. Asimov. Grand tour: A tool for viewing multidimensional data. *SIAM J. Science and Statistical Computing*, 6:128–143, 1985.
4. G.D. Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing.* Prentice Hall, 1999.
5. C. Blake and E. K. Merz. Uci repository of machine learning databases, 1998.
6. C. Chen. *Information Visualization and Virtual Environments.* Springer-Verlag, 1999.
7. H. Chernoff. The use of faces to represent points in k–dimensional space graphically. *Journal of Am. Statistical Assoc.*, 68:361–368, 1973.
8. M. C. Chuah and S. G. Eick. Managing software with new visual representations. In *IEEE Information Visualization*, pages 30–37, 1995.
9. Arthur Flexer. On the use of self-organizing maps for clustering and visualization. In *PKDD'99*, pages 80–88, 1999.
10. P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–474, 1985.
11. A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. In *Visualization'90.*
12. B. Johnson and B. Shneiderman. Treemaps: A space-filling approach to the visualization of hierarchical information. In *Visualization'91*, pages 284–291, 1991.
13. J. Jolliffe. *Principal Component Analysis.* Springer Verlag, 1986.
14. D.A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Trans. Visualization and Computer Graphics*, 6(1):59–78, 2000.
15. D.A. Keim. Information visualization and visual data mining. *IEEE Trans. Visualization and Computer Graphics*, 8(1):1–8, 2002.
16. D.A. Keim and M.C. Hao. Hierarchical pixel bar charts. *IEEE Trans. Visualization and Computer Graphics*, 8(3):255–269, 2002.
17. N. Lopez, M. Kreuseler, and H. Schumann. A scalable framework for information visualization. *IEEE Trans. Visualization and Computer Graphics*, 8(1):39–51, 2002.
18. A. Mead. Review of the development of multidimensional scaling methods. *The Statistician*, 33:27–35, 1992.
19. L. Nowell, S. Havre, B. Hetzler, and P. Whitney. Themeriver: Visualizing thematic changes in large document collections. *IEEE Trans. Visualization and Computer Graphics*, 8(1):9–20, 2002.
20. D. Tang, C. Stolte, and P. Hanrahan. Polaris: A system for query, analysis and visualization of multidimensional relational databases. *IEEE Trans. Visualization and Computer Graphics*, 8(1):52–65, 2002.

Metadata to Support Transformations and Data & Metadata Lineage in a Warehousing Environment

Aurisan Souza de Santana and Ana Maria de Carvalho Moura

Military Institute of Engineering - IME/RJ
Department of Computer Engineering - Rio de Janeiro - Brazil
{aurisan,anamoura}@de9.ime.eb.br

Abstract. Data warehousing is a collection of concepts and tools which aim at providing and maintaining a set of integrated data (the data warehouse – DW) for business decision support within an organization. They extract data from different operational data sources, and after some cleansing and transformation procedures data are integrated and loaded into a central repository to enable analysis and mining. Data and metadata lineage are important processes for data analysis. The first allows users to trace warehouse data items back to the original source item from which they were derived and the latter shows which operations have been performed to achieve that target data. This work proposes integrating metadata captured during transformation processes using the CWM metadata standard in order to enable data and metadata lineage. Additionally it presents a tool specially developed for performing this task.

1 Introduction

Metadata has been identified as a key success factor in DW projects. It represents all knowledge captured from the DW processes, providing accurate mechanisms to generate more precise information, so crucial for decision support. Metadata captures all sort of information necessary to extract, transform and load data from source systems into the DW. This information will be used later to interpret the DW contents.

Data transformation is an important process in DW. However, knowledge transformation is scattered in programs source code, procedures, or stored as proprietary information of exclusive use of the data transformation tool. At this point, we may raise the following question: “How is it possible to follow a data item step by step since its source system, before extraction, until its complete integration to the DW, after being submitted to all kind of transformations?” The ability of visualizing data in a reverse order, following its transformation step by step is called lineage data process [5]. Current commercial ETL (Extraction, Transformation, Load) tools offer neither lineage data mechanisms, nor *back tracing*, i.e., the reverse track submitted by a data item at some granularity level (table, file, register, document or element), which turns the data transformation activity into a complex process. Due to the heterogeneity of transaction and legacy systems as well as the diversity of data transformation solutions, the use of a metadata standard becomes an important requirement. It provides a way to represent all the metadata captured during data transformation process, which is necessary for data and metadata lineage. The Common Warehouse Metamodel

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 249–258, 2004.

(CWM – [12]) is a metadata standard specifically used in DW domain, which offers essential packages and metamodels to represent data transformation, assuring semantic metadata equivalence.

This paper presents two main contributions: the first concerns the use of the CWM standard to support transformations and data and metadata lineage processes in a DW environment. This task will be achieved by integrating some of the CWM metamodels, and additionally including new transformation capabilities. The resulting metamodel can also be represented in XML Metadata Interchange (XMI)[1] to provide metadata interchange between other repositories. In order to validate this proposal, we have developed a tool to perform transformations and data and metadata lineage in a DW environment, which is also considered as a second contribution.

The rest of this paper is organized as follows. Section 2 describes the main concepts of data transformations operators in a DW and an example that will be used as use case throughout the paper. Section 3 presents the main algorithms to perform data and metadata lineage. Section 4 presents the CWM metamodel and how it is has been extended to support data lineage. Section 5 presents the tool developed to support transformation and lineage functionalities. Section 6 reviews some related work, and finally section 7 concludes the paper.

2 Transformations

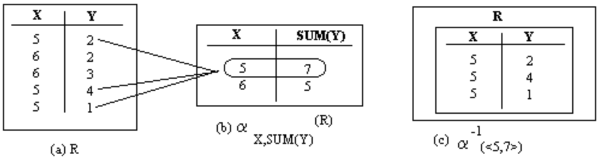
ETL is a well-known process cycle inherent to a *warehousing* environment. Whenever it occurs it generates a new materialized view of the DW, involving data transformations under different levels of granularity (table, column, file, etc.). In order to ensure data quality whenever a new view is produced, it is important to establish a metadata management strategy, whose aim is to integrate all the metadata in the *data warehousing* environment.This process will be achieved by the CWM metamodel, whose main characteristics will be presented in section 4. Therefore, in order to understand how metadata is integrated using this model, it is essential to describe some important lineage and transformation concepts, as these are relevant in the integration process.

Our work was based on a previous study made by Cui and Widom [5][6], where a data set is defined as any set of data items - tuples, values, complex objects - with no duplicates in the set. A transformation T is any procedure that takes data sets as input and produces data sets as output. For any input data set I, the application of T to I resulting in an output set O, denoted $T(\mathrm{I})=O$, is an instance of T. Given transformations T_1 and T_2 (both components of T), $T= T_1 \circ T_2$ is the transformation that first applies T_1 to I to obtain I' and then applies T_2 to obtain O. A transformation that is not defined as a composition of other transformations is atomic. Additionally, a transformation can be of three types: **Dispatcher:** for each input data, a dispatcher transformation produces zero or more output data items, independently; **Aggregator:** an aggregator transformation T for any input data set I generates a resulting output set O and I is composed of disjoint partitions $\mathrm{I}_1,..,\mathrm{I}_n$ of I such that $T(\mathrm{I}_k)=\{o_k\}$ for k=1,…n. Furthermore, an aggregator operator can be **Context-Free** (if any two input data items

[1] www.omg.org/technology/documents/formal/xmi.htm

either always belong or not to the same input partition, regardless of the other items in the input set) and **Key-Preserving** (giving any input set I and its input partition $I_1,..,I_n$ for output $T(I)=O\{o_1,\ldots,o_n\}$, all subsets of I_k produce a single output item with the same key value as o_k, for k=1,…n); **Black Box:** it is an atomic transformation that is neither a dispatcher nor an aggregator, and it is not possible to generate a lineage tracing procedure.

The purpose of data lineage is to implement the reverse operator from the set of canonical operators, **ASPJ** (**A**ggregation, **S**election, **P**rojection and **J**oin), in order to present the subset of original relations that derived a specific tuple in a resulting relation. Intuitively, they define data lineage as following: "given a data item in a materialized warehouse view, determine the set of source data items that produced the view item". Fig. 1 shows an example of tuple derivation that uses the aggregation operator α. Given a relation R and a tuple $t=<5,7> \in \alpha_{X,sum(Y)} R$, a tuple derivation t is expressed by $\alpha_{X,sum(Y)}{}^{-1} R(<5,7>) = \{<5,2>,<5,4>,<5,1>\}$.



X	Y
5	2
6	2
6	3
5	4
5	1

X	SUM(Y)
5	7
6	5

R

X	Y
5	2
5	4
5	1

Fig. 1. The Aggregation Operator.

2.1 A Use Case

This section presents an example of transformation operations based on Cui and Widom classification [6]. In the scope of ASPJ operations, transformations will be integrated into the CWM model, as well as they will be used as basis to implement data lineage process in our tool. To exemplify such a situation, consider the relations Sales and Shops presented in Tables 1 and 2.The column Product of Sales table contains a list of information about name marktrade and category of a product.

Suppose a marketing manager wants to analyze a promotion result implemented in March 2003 for *cleaning* and *drinking* products.

In order to extract the required data it is necessary to perform the following list of transformations, illustrated by the graph of Fig. 2: $\mathbf{T_1}$: selects *cleaning* and *drinking* products from **V1**, producing relation **V2**. This is a Dispatcher operator; $\mathbf{T_2}$: joins **V2** and Shop. where **V2.shopcd = Shop.Id** followed by a projection, producing the relation **VL1** with the following schema: **<shopname, city, state, product, marktrade, category,total,date>.** This is a Dispatcher operator; $\mathbf{T_3}$: selects from **VL1** sales occurred in 'March', producing the relation **VL2;**$\mathbf{T_4}$: aggregates in **VL2** sales by 'shop' and 'category', generating the resulting relation **VL3**, illustrated in Table 3. In these examples it is intuitive verifying that, according to the previous definitions T_1, T_2 and T_3 are transformations of type dispatcher, whereas T_4 is of type aggregator.

Table 1. Sales Table.

Product	Mark Trade	Category	Shop Cd	Customer Cd	Date (m/d/y)	Qty	Price	Total (U$)
Coco Soap	Jax	Cleaning	L3	C3	3/2/03	1	3.00	3.00
Cream Soap	Tox	Cleaning	L3	C1	3/2/03	20	1.00	20.00
cheese	Minas	Frozen	L2	C1	2/2/03	4	3.62	14.48
Soap	Neutral	Cleaning	L1	C1	2/10/03	10	2.50	25.00
Soft drink	Coke	Drinking	L2	C2	2/15/03	5	4.50	22.50
Light soft	Coke	Drinking	L4	C2	3/3/03	1	6.22	6.22
Paper	Soft	Cleaning	L5	C2	3/4/03	20	6.00	126.00
Chicken	Perdigao	Frozen	L6	C3	3/4/03	3	12.00	36.00
Almond	Sadia	Frozen	L6	C2	3/4/03	2	4.35	8.70
Coco soap	Ruth	Cleaning	L5	C3	3/10/03	1	8.30	8.30
Phosphorate	Lux	Cleaning	L2	C1	3/15/03	5	2.00	10.00
English Tea	Leao	Drinking	L4	C2	3/15/03	8	0.80	6.40

Table 2. Shop Table.

Id	ShopName	Address	City	State	Manager
L1	ACME Botafogo	11XXX Street	Rio de Janeiro	RJ	João P. Souza
L2	ACME Flamengo	22 YYY Street	Rio de Janeiro	RJ	Maria Silva
L3	ACME Copacabana	33 ZZZ Street	Rio de Janeiro	RJ	Paulo Pereira
L4	ACME Pituba	44 KKK Street	Salvador	BA	Carlos José
L5	ACME Barra	55 HHH Street	Salvador	BA	Ana Silva
L6	ACME Leblon	66 TTT Street	Rio de Janeiro	RJ	Bianca Souza

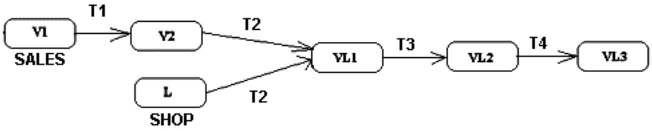


Fig. 2. Graph of Transformations.

Table 3. Resulting Relation VL3.

Name	**Category**	**Total (U$)**
ACME Pituba	Drinking	12.62
ACME Barra	Cleaning	134.30
ACME Flamengo	Cleaning	10.00
ACME Cobacabana	**Cleaning**	**23.00**

Table 4. Original Tuple from Shop Table.

Id	**ShopName**	**Address**	**City**	**State**	**Manager**
L3	ACME Copacabana	33 ZZZ Street	Rio de Janeiro	RJ	Paulo Pereira

Now suppose the marketing manager wants to see details about the tuple **<ACME Copacabana, Cleaning, 23,00 >**. In fact, he wants to visualize the data subsets that produced tuple t from its original relations, such as those shown in Tables 4 and 5.

Table 5. Original Tuples from Sales Table (2.5).

Product	Mark	Category	ShopCd	Customer	Date	Qty	Price	Total
	Trade			Cd				(U$)
Coco Soap	Jax	Cleaning	L3	C3	3/2/03	1	3.00	3.00
Cream Soap	Tox	Cleaning	L3	C1	3/2/03	20	1.00	20.00

3 Data and Metadata Lineage Processes

Lineage tracing solution does not provide the visualization of all transformations this data item has been submitted to. This issue justifies the necessity of a metadata lineage, since data lineage concerns only data, where only content is evaluated. In this work, data lineage is considered as a relational data based approach, as it is applied over a data space considered minimal of ASPJ transformations. However, other types of transformations (out of the relational range) may usually occur in a *warehousing* environment. Consider, for example, transformations that can be represented as a tree expression, such as (ASSIGN(Product.IdProd, SUM(ProductPartition.UPC, 1000000000)) – see fig. 6). In this context, metadata lineage raises as an important issue.

Metadata lineage is an orthogonal approach when compared to data lineage. The combined solution of applying data tracing with intensive use of metadata provides great benefit concerning the quality of data produced in a *warehousing,* as it is necessary to represent the set of metadata related to each transformation. Intuitively the idea is: given a data item, resulting from an ETL process (column, table or materialized view), the metadata lineage procedure will show all transformations applied to that item, whenever navigation is achieved across an integrated metadata repository.

Consider **W** a warehousing. **m** the ETL processes used to produce data in a DW and **M** a metamodel to represent the metadata of **W**. Assume P_k $k<=m$. ($\{k,m,n,i\} \in$ N, N:natural numbers) is an ETL process and w_i a column of a materialized view $\mathbf{V}_i$. Consider also the transformation operator **Op** applied to the column w_i, generating the column w_{i+1} of the materialized view $\mathbf{V}_{i+1}$, such that $\mathbf{Op}(w_i) \rightarrow w_{i+1}$.

If **n** transformation operators have been applied to generate a column in a view, $\mathbf{ML}(w_i, P_k)$ – the metadata lineage of w_i and $\mathbf{P}_k$ – is the list Γ of all transformations that participated in the column w_i generation, such that: $\mathbf{ML}\ (w_i, \mathbf{P}_k) = \Gamma(\mathbf{Op}(w_i) \rightarrow w_{i+1}), i=1..n; k<=m.$ An operator (Op) can also be applied to a materialized view (V_i) generating a view into another level (V_{i+1}), i.e., $\mathbf{Op}(\mathbf{V}_i) \rightarrow \mathbf{V}_{i+1.}$

Thus, the metadata lineage is also applied at a materialized view granularity level, where the list of all transformations Γ applied to a view in a determined ETL process $\mathbf{P}_k$ is defined as: $ML\ (\mathbf{V}_i, \mathbf{P}_k) = \Gamma\ (\mathbf{Op}(\mathbf{V}_i) \rightarrow \mathbf{V}_{i+1})_{i=1.\,....\,n}$; k <= m.

Hence, *total metadata lineage* is defined as $\mathbf{TML\ (W,M)} = \cup\ (\mathbf{ML}(\mathbf{V}_i \mid w_{i,}\mathbf{P}_k))_{,}$ **k=1,...m** i.e., lineage can be applied to both a column or a view. Fig. 3 presents an algorithm to generate metadata lineage. This algorithm receives a target item as parameters of a data transformation at a granularity level (table or column), and for each transformation T ($T(E_s) \rightarrow E_t$) that originated the current target, the adjacent matrix[2] that represents the graph of transformations is updated and the routine MetaLineage containing the data source item E_s is recursively called. The procedure is executed until the whole graph is generated.

Fig. 4 shows an algorithm to visualize metadata or data lineage after being submitted to a set of transformation operations, represented through a graph. The algorithm initially checks the transformation category (represented by the edge in a transformation graph). If the transformation is of type Dispatcher, the tree expression containing the structure and the semantic of the transformation is presented (fig. 6, window 2), as well as the lineage tracing that generated these data items. In case of a Black-Box transformation, its description (that can be a script or a program) is also shown. Otherwise, the algorithm enables data lineage for data cubes. This procedure takes into account that all metadata concerning ETL processes definition are stored in a CWM based repository, and in the case of hypercubes, these can be accessed from metadata interoperability standards such as XMI (XML Metadata Interchange).

```
Procedure MetaLineage
($Target E_t$)
T: Transformation
for each T | $T(E_s) \rightarrow E_t$
do
 UpdateGraphNodes($T, E_s, E_t$);
 MetaLineage($E_s$);
```

Fig. 3. The Metadata Lineage Procedure.

```
Procedure MetaLineageView
($Target E_s , E_t$)
T: Transformation
    for T | $T(E_s) \rightarrow E_t$ do
    if T is Dispatcher
     enableTreeExpression() ;
    else
     if T is BlackBox
      showTransformation () ;
     else
       enableLineageCube() ;
```

Fig. 4. Visualizing Data and Metadata Lineage.

4 Using CWM to Integrate Metadata

Metadata plays an essential role in the process of data integration in a *warehousing* environment. The idea of using a metadata standard to provide a generic model to represent metadata independently of platform, and ensuring metadata semantic equivalence among all the *warehousing* components emerges as an interesting integration solution, and provides interoperability between metadata repository.

The CWM (Common Warehouse Metamodel) [12,14] is a specific standard completely oriented to provide DW main requirements, such as metadata representation. Its main purpose is to provide a metamodel that enables integration and interoperability of metadata between components (tools, software programs, and repositories) in a *warehousing* environment. It provides a framework to represent metadata captured from different data sources, transformations, analysis, processes and operations to create and manage the DW. It is composed of specific *packages*, which are responsible for performing each one of these functionalities: **The Relational Package:** this

[2] An element $A_{i,j}$ of an adjacent matrix A is an edge that connects a node *i* to a node *j* in a graph.

package provides a metamodel to represent metadata from relational interfaces, and it uses standard SQL to access relational DBMS catalogs; **The Transformation Package**: the Transformation Package of CWM implements a hierarchy of data transformation (Activity-Step-Task), grouping a set of transformations into a logical unit represented by Transformation, TransformationTask, TransformationStep and TransformationActivity classes. In this way, an activity contains a set of steps, which have a set of tasks, and each task implements a set of data transformations. The CWM transformation metamodel has been extended in order to provide data and metadata lineage implementation. Each transformation associates a data item source with a target data item in different granularity levels (Column, Table or View). The transformation is then categorized according to one of the types Aggregator, Dispatcher or BlackBox; **The OLAP Package:** the CWM ***OLAP Package*** defines a metamodel to represent multidimensional structures so employed in OLAP applications.

5 The Lineage Tool

The MetaL Tool (**Meta**data **L**ineage **T**ool) developed in the context of this work can be seen as a module embedded in the *warehousing* architecture. It has been developed in a relational framework, where data from flat files or semi-structured documents are not managed in the first version of this tool. Fig. 5 represents the star schema used in the example. It will serve as a case study to build the DW of an organization called ACME.

In order to show the way the tool manages metadata according to the ***Activity-Step-Task-Transformation*** CWM hierarchy, consider the following list of SQL queries, whose purposes are extracting, transforming and updating DW dimensions. In this simple example, dimension Product is generated from two other tables of a database (not shown in the example), named respectively ProductPartition01 and ProductPartition02. These tables contain transactional data that will be used to perform the initial load of the DW product dimension.

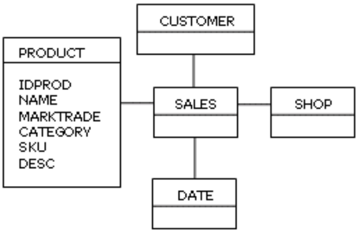


Fig. 5. The Star Schema Example.

Activity: Load_Dim (Load the dimensions Product, Date, Shop and Customer).

- **Step 1**:**Load_Dim _Product** (Load table Product from table ProductPartition01). It is composed of the following task:
  - **Task 1**: **Load_from_Partition01**

```
"INSERT INTO Product (IdProd, Desc, TradeMark, Category,
SKU) SELECT UPC + 1000000000, Description, TradeMark,
Categ, SKU FROM ProductPartition01".
```

This task is composed of five transformations, each one applied to a different column of Product table. For example, it is possible to define the transformation applied to IdProd column as:
ASSIGN(Product.IdProd, SUM(ProductPartition.UPC, 1000000000)). This transformation can be represented as a tree expression (see window 3 of Fig. 6);

5.1 The Data and Metadata Lineage Module

This module allows the user to perform the back tracing of data and metadata transformation processes when lineage mechanisms, such as those described in section 2 are applied. Figure 7 presents the derivation graph for a column. This graph shows that the column **IdProduct** of Product table has been generated from **UPC** and **SKU** columns of **ProductPartition01** and **ProductPartition02** tables.

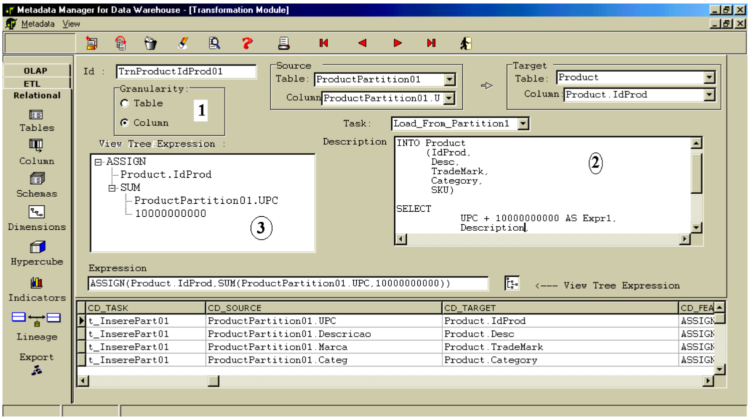
5.2 The XMI Interoperability Module

The set of CWM metadata captured during transformations can be exported into XMI metadata standard, a very appropriate mechanism to provide metadata interoperability with other systems [3].

6 Related Work

Significant work has been developed on data transformation and integration techniques in DW [1,16]. Some focused their efforts on implementing lineage information in different contexts: at schema level [2,8,11]; at instance level [5]; for data cleaning [7]; or embedded in a framework to provide metadata information about data items from multiple data sources [2,11]. Additionally, Hachem et al. in [8] propose a system to manage derived attributes from a scientific database named Gaea, extended to object oriented modeling, where semantic constructors are also supported.

Another set of works have concentrated their efforts on using metadata to improve and standardize most of warehousing processes. In this context Jarke et al.[9] present a framework to improve management quality in Warehousing environment, whereas in [10] they describe a metamodel to represent complex activities within a DW operational process. It is based on the language Telos, and uses a metadata repository. A very good overview on metadata management in DW environment is done in [17], where important aspects about standards, tools and metadata categorization are emphasized. This paper has similar characteristics to the works developed in [2,4,5]. In [2] the OIM (Open Information Model), a repository schema that covers the metadata needed for DW scenarios, is used to support DW transformations. The others are focused on data lineage tracing based on transformation properties, taking into account transformations performed in the ASPJ space at fine-grained level (or instance level).



Id : TrnProductIdProd01

Granularity: ○ Table ● Column

Source Table: ProductPartition01 Column: ProductPartition01.U

Target Table: Product Column: Product.IdProd

Task: Load_From_Partition1

View Tree Expression :

```
⊟ ASSIGN
  ├ Product.IdProd
  ⊟ SUM
    ├ ProductPartition01.UPC
    └ 10000000000
```

Description

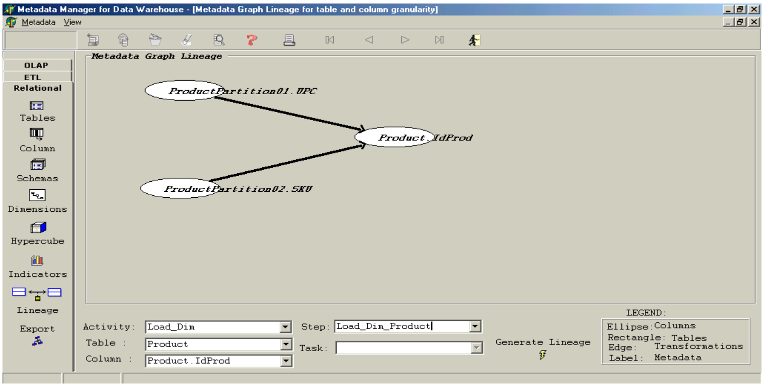
```
INTO Product
     (IdProd,
      Desc,
      TradeMark,
      Category,
      SKU)

SELECT
           UPC + 10000000000 AS Expr1,
           Description,
```

Expression

ASSIGN(Product.IdProd,SUM(ProductPartition01.UPC,10000000000))

CD_TASK	CD_SOURCE	CD_TARGET	CD_FEA
t_InserePart01	ProductPartition01.UPC	Product.IdProd	ASSIGN
t_InserePart01	ProductPartition01.Descricao	Product.Desc	ASSIGN
t_InserePart01	ProductPartition01.Marca	Product.TradeMark	ASSIGN
t_InserePart01	ProductPartition01.Categ	Product.Category	ASSIGN

Fig. 6. Transformation and View Expression.



Activity: Load_Dim

Table : Product

Column : Product.IdProd

Step: Load_Dim_Product

Task:

LEGEND:
Ellipse: Columns
Rectangle: Tables
Edge: Transformations
Label: Metadata

Fig. 7. The Metadata Lineage Graph.

7 Conclusion

This work presented the problem of data and metadata lineage, as well as it described important concepts of transformations required in the data lineage process. The purpose was to represent and integrate the metadata generated by these processes using the CWM metamodel. In order to provide these functionalities, a tool including algorithms to support data transformations and data and metadata lineage tracing at different levels of granularity has been developed. Additionally, this tool provides: technical and ETL processes metadata management; the ability of visualizing a transformation through its expression tree; and the integrated metamodel that can be exported in XML, based on XMI. This latter functionality represents an important mechanism to provide interoperability between other DW systems. As future work we intend to implement other aggregation transformations not contemplated in this work, as well as to provide lineage at other granularity levels (XML document, for example) and to extend the tool to other CWM packages.

References

1. S. Abiteboul. S. Cluet. T. Milo. *Tools for Data Translation and Integration*. IEEE Data Engineering Bulletin. 22(1):3-8; March 1999.
2. P. Bernstein and T. Bergstraesser. *Meta-Data Support for Data Transformations Using Microsoft Repository*. IEEE Data Engineering Bulletin. 22(1): 19-24;March 1999.
3. A. Brodsky, Doney Gary. Grose Timothy. Mastering XMI – Java Programming with XMI. XML and UML. OMG Press 2002
4. Y. Cui. *Lineage Tracing for Data Warehouse*.PhD Thesis. Department of Computer Science of Stanford University. December 2001.
5. Y. Cui. J. Widom. J.L.Wiener. *Tracing the Lineage of View Data in a Warehousing Environment.* ACM TODS. vol. 25. n^0 2. June 2000. pages 179-227.
6. Y. Cui. J. Widom. *Lineage Tracing for General Data Warehouse Transformations*. Proc.of 27th International Conference on Very Large Data Bases. Roma. September 2001.
7. H. Galhardas, D. Florescu et al. Improving Data Cleaning Quality Using a Data Lineage Facility. DMDW'01 - Proceedings of the International Workshop on Design and Management of Data Warehouses. Interlaken, Switzerland.
8. N.I. Hachem. K. Qiu. M. Gennert. *Managing Derived Data in the Gaea Scientific DBMS*. VLDB. Pages 1-12. Dublin-Ireland. August 1993.
9. M. Jarke, P. Vassiliadis at al. Architecture and Quality in Data Warehouses. CAiSE 98 - Conference on Advanced Information Systems Engineering. Pisa, Italy.
10. M. Jarke, P. Vassiliadis et al. A Model for Data Warehouse Operational Processes. CAiSE 00 - Conference on Advanced Information Systems Engineering. Stockholm, Sweden.
11. T. Lee. S. Bressan and S. Madnick. *Source Attribution for Querying against Semi-Structured Documents*. Workshop on Web Information and Data Management. Pages 33-39. November 1998.
12. OMG – *CWM Specification*. www.omg.org/technology/cwm.
13. *Oracle Warehouse Builder*. www.oracle.com.
14. J. Poole. D. Chang. D. Tolbert. *Common Warehouse Metamodel – An Introduction to the Standard for Data Warehouse Integration*. OMG Press. 2002.
15. A. S. Santana. *A Tool for Supporting Transformation and Data Lineage in Data Warehouse Environment* (in Portuguese). Master Thesis. IME-RJ. June 2003.
16. C. Squire. *Data Extraction for the Data Warehouse*. ACM SIGMOD International Conference on Management of Data. Pages 446-447. May 1995.
17. M. Staudt, A. Vaduva et al. Metadata Management and Data Warehousing. DMDW'99 Proceedings of the International Workshop on Design and Management of Data Warehouses. Heidelberg, Germany.

Classification Based on Attribute Dependency

Yue-Shi Lee and Show-Jane Yen

Department of Computer Science and Information Engineering, Ming Chuan University
5 The-Ming Rd., Gwei Shan District, Taoyuan County 333, Taiwan, R.O.C.
{leeys,sjyen}@mcu.edu.tw

Abstract. The decision tree learning algorithms, e.g., C5, are good at dataset classification. But those algorithms usually work with only one attribute at a time. The dependencies among attributes are not considered in those algorithms. Thus, it is very important to construct a model to discover the dependencies among attributes and to improve the accuracy of the decision tree learning algorithms. Association mining is a good choice for us to concern with the problems of attribute dependencies. Generally, these dependencies are classified into three types: categorical-type, numerical-type, and categorical-numerical-mixed dependencies. This paper proposes a CAM (Classification based on Association Mining) model to deal with such kind of dependency. The CAM model combines the association mining technologies and the traditional decision-tree learning capabilities to handle the complicated and real cases. According to the experiments on fifteen datasets from the UCI database repository, the CAM model can significantly improve both the accuracy and the rule size of C5. At the same time, the CAM model also outperforms the existing association-based classification models, i.e., ADT (Association-based Decision Tree) and CBA (Classification Based on Association).

1 Introduction

One important application of data mining is the ability to perform classification in a huge amount of data [1]. The decision-tree learning algorithm is one of the most important results [2]. Basically, they build decision trees by recursively partitioning the dataset according to the selected attributes. In this framework, they usually deal with only one attribute at a time. The dependencies among attributes are not considered. Unfortunately, in real world, most datasets contain attributes, which are dependent. Thus, it is very important to construct a model to discover the dependencies among attributes and to improve the accuracy of the decision tree algorithms.

Association mining is a good choice for us to concern with the problems of attribute dependencies. Generally, these dependencies are classified into three types: categorical- type, numerical-type, and categorical-numerical-mixed dependencies [3]. This paper proposes a CAM (Classification based on Association Mining) model to deal with such kind of dependency. The CAM model combines the association rule mining technologies and the traditional decision-tree learning capabilities to handle the complicated and real cases.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 259–268, 2004.

Several related works concerned about categorical-type dependency and association-based classification. Chen [4] firstly proposed the concepts for categorical-type dependencies. However, they did not describe how these dependencies could be obtained efficiently. Liu, Hsu and Ma proposed a CMAR (Classification based on Multiple Association Rules) model based on association rule mining [5]. Basically, this method firstly generated a complete association rule set, and then pruned the remaining rules based on the correlation analysis. Wang, Zhou and He also proposed an ADT (Association-based Decision Tree) model to generate decision rules using association rule mining [6]. ADT firstly generates a complete set of association rules satisfying only minimum confidence threshold, and then prunes over-fitting rules on an accuracy-driven basis. In general, CMAR and ADT proposed novel techniques to replace the traditional decision tree learning algorithms, e.g., C5. Besides, they only compare the classification accuracy with C5. The comparisons on decision tree size are ignored. The decision tree size is also an important criterion to evaluate a classification system. The differences among our CAM model and the above models are described as follows. First, our CAM model is proposed to enhance the traditional decision tree algorithms, e.g., C5. Second, our goal is to reduce both the classification error and the decision tree size. This paper is organized as follows. Section 2 introduces the concept of attribute dependency. Section 3 then describes our CAM model. Before concluding, the experimental results are demonstrated in Section 4.

2 Attribute Dependency

To describe the concept of attribute dependency, Table 1 shows a transportation dataset with 8 tuples. It illustrates the relations among gender (male or female), income (low or high), distance (away from home) and the target – vehicle (bike or car).

Table 1. Transportation Dataset with 8 Tuples

Gender	Income	Distance	Target: Vehicle
Male	Low	Far	Car
Male	Low	Far	Car
Female	High	Far	Car
Female	High	Near	Car
Male	High	Near	Bike
Male	High	Near	Bike
Female	Low	Near	Bike
Female	Low	Far	Bike

Based on the entropy-based decision-tree learning algorithm, i.e., C5, a decision tree with tree size 6 is returned by the algorithm. It is shown in Fig. 1. If we consider and combine more than one attribute, e.g., gender and income, at a time, a more compact decision tree with tree size 4 can be obtained. It is shown in Fig. 2. A compact decision tree is preferable, since it is more general and its predictive power is often higher than that of a complex decision tree.

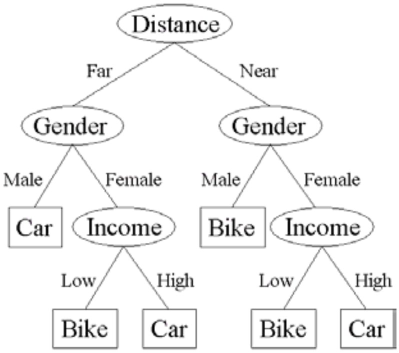


Fig. 1. The Decision Tree for the Concept Vehicle

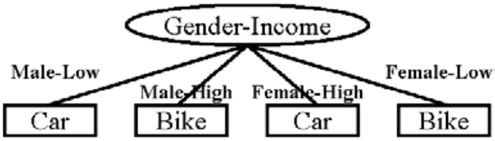


Fig. 2. A Compact Decision Tree for the Concept Vehicle

This example illustrates the drawbacks of the traditional decision tree algorithms, because they usually work with only one attribute at a time. That is, the dependencies among attributes are not considered in those algorithms.

Because both gender and income in this example are all categorical-type attributes, we call this kind of dependency is the categorical-type dependency. If the dependency is among numerical-type attributes, we call this kind of dependency is the numerical-type dependency. Moreover, the dependency among categorical-type and numerical-type attributes is called the categorical-numerical-mixed dependency. In next section, we'll propose a model to discover these three kinds of dependencies.

3 CAM (Classification Based on Association Mining) Model

The major difficulty to discover attribute dependency is how these dependencies could be obtained efficiently. If we consider and combine any kinds and any number of attributes to find the best one, this is a combinatorial explode problem. Thus, our CAM (Classification based on Association Mining) model is proposed to overcome this problem. The architecture of our CAM model is depicted in Fig. 3. In this architecture, the training data is firstly sent to the module of association rule mining. Association rule mining has been studied popularly in data mining research [7, 8, 9, 10, 11]. In this paper, the multidimensional association analysis is used to find the dependencies among attributes.

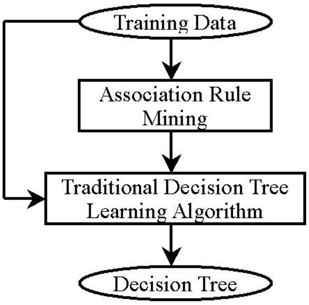


Fig. 3. The Architecture of the CAM Model

The training data and the results obtained by the module of association rule mining are then sent to the traditional decision-tree learning algorithm, i.e., C5, to improve the performance of C5. The followings illustrate the rule extraction steps in CAM model over the numerical-categorical-mixed dataset.

Step1. Because the module of association rule mining cannot accept the numerical-type attributes as input, we must to discretize the attribute value of each numerical-type attribute into the categorical-type attribute value. Thus, we use C5 to transform each numerical-type attribute into the categorical one according to the relationships between the numerical-type attribute and the target attribute.

Step2. We create an attribute-value hierarchy according to the relationships between the attributes and the attribute values. In this hierarchy, we give each attribute value a unique number. Based on this attribute-value hierarchy, we encode each attribute value into its corresponding number for multidimensional association analysis.

Sep3. According to the encoded dataset and a predefined minimum support count, the frequent itemsets are generated.

Step4. Using the attribute-value hierarchy created in Step 2 and the frequent itemsets generated in Step 3, the attribute dependencies for conditional attributes are identified according to the strength of the association between conditional attributions and target attribute. Finally, the attribute dependencies for conditional attributes are depicted by a linkage diagram.

Step5. Based on the linkage diagram, the new dataset is formed. The new dataset is sent to the C5 to generate the decision tree.

To clearly describe these steps, the followings describe the details about how to infer the decision tree and rules for Table 2. In Table 2, “Income” is a numerical-type attribute. Thus, a dataset for “Income” is generated. It is shown in Table 3. Based on Table 3, a decision tree is generated by C5. It is shown in Fig. 4.

Table 2. Transportation Dataset with 8 Tuples

Gender	Income	State	Target: Vehicle
Male	32,000	NY	Car
Male	13,000	CA	Car
Female	53,000	NY	Car
Female	61,000	NY	Car
Male	90,000	CA	Bike
Male	74,000	CA	Bike
Female	23,000	NY	Bike
Female	33,000	CA	Bike

Table 3. Dataset for "Income"

Income	Target: Vehicle
32,000	Car
13,000	Car
53,000	Car
61,000	Car
90,000	Bike
74,000	Bike
23,000	Bike
33,000	Bike

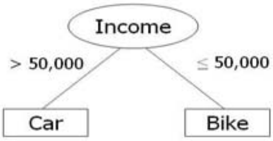


Fig. 4. The Decision Tree for Table 3

Table 4. Transformed Dataset

Gender	Income	State	Target: Vehicle
Male	Low	NY	Car
Male	Low	CA	Car
Female	High	NY	Car
Female	High	NY	Car
Male	High	CA	Bike
Male	High	CA	Bike
Female	Low	NY	Bike
Female	Low	CA	Bike

According to Fig. 4, if the attribute vale of "Income" is greater than 50,000, we transform it into "High"; otherwise, we transform it into "Low". After this treatment, the original dataset are transformed into a new one. It is shown in Table 4.

Next, we create an attribute-value hierarchy according to the relationships between the attributes and the attribute values. It is shown below. In this hierarchy, we give each attribute value a unique number from 1 to 8.

Based on this attribute-value hierarchy, we then encode each attribute value into its corresponding number. The encoded results are shown in Table 5.

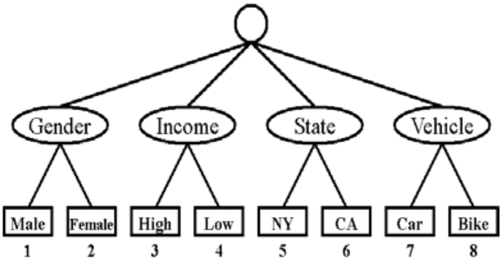


Fig. 5. Attribute-Value Hierarchy

Table 5. Encoded Dataset

Gender	Income	State	Target: Vehicle
1	4	5	7
1	4	6	7
2	3	5	7
2	3	5	7
1	3	6	8
1	3	6	8
2	4	5	8
2	4	6	8

According to this encoded dataset and a predefined minimum support count, the frequent itemsets are generated. It is shown in Table 6. In this case, the minimum support count is set to 2. The support count for each frequent itemset in Table 6 is listed in the parenthesis.

Table 6. Frequent Itemsets

	Frequent Itemsets
L1	{1} (4), {2} (4), {3} (4), {4} (4), {5} (4), {6} (4), {7} (4), {8} (4)
L2	{1,3} (2), {1,4} (2), {1,6} (3), {1,7} (2), {1,8} (2), {2,3} (2), {2,4} (2), {2,5} (3), {2,7} (2), {2,8} (2), {3,5} (2), {3,6} (2), {3,7} (2), {3,8} (2), {4,5} (2), {4,6} (2), {4,7} (2), {4,8} (2), {5,7} (3), {6,8} (3)
L3	{1,3,6} (2), {1,3,8} (2), {1,4,7} (2), {1,6,8} (2), {2,3,5} (2), {2,3,7} (2), {2,4,8} (2), {2,5,7} (2), {3,5,7} (2), {3,6,8} (2)
L4	{1,3,6,8} (2), {2,3,5,7} (2)

For discovering the attribute dependencies, only L3 and L4 are considered. At the same time, only the itemsets correlated to the target attribute are reserved. According to these treatments, the reserved frequent itemsets are listed in Table 7.

Table 7. Reserved Frequent Itemsets

	Frequent Itemsets
L3	{1,3,8} (2), {1,4,7} (2), {1,6,8} (2), {2,3,7} (2), {2,4,8} (2), {2,5,7} (2), {3,5,7} (2), {3,6,8} (2)
L4	{1,3,6,8} (2), {2,3,5,7} (2)

Based on the attribute-value hierarchy in Fig. 5, we then translate each frequent itemset in Table 7 into its attribute level. For example, we translate the frequent itemset {1, 3, 8} into its attribute level {Gender, Income, Vehicle} with support count 2. After translation, the same itemsets are merged into one and we sum their support counts. After these steps, the results are listed below.

Table 8. Attribute Associations

	Frequent Itemsets
L3	{Gender, Income, Vehicle} (8), {Gender, State, Vehicle} (4), {Income, State, Vehicle} (4)
L4	{Gender, Income, State, Vehicle} (4)

In Table 8, "Gender" and "Income" have stronger association with "Vehicle" (the strength of the association is 8). The "Gender", "Income" and "State" have the secondary stronger association with "Vehicle" (the strength of the association is 4). According to the strength of the association, we depict the linkage diagram shown below.

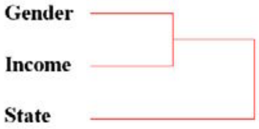


Fig. 6. Linkage Diagram for Conditional Attributes

According to Fig. 6, the following steps are adopted for clustering and combining conditional attributes. Initially, we select the number of cluster is 3 (maximum number of cluster). Then, we set the number of cluster to 2, and verify whether the accuracy of C5 in two clusters is better than three clusters or not. If two clusters get better performance than three clusters, we set the number of cluster to 1, and repeat the same processing; otherwise stop it. In this example, when the number of cluster is 2, the best performance can be reached. Thus, we cluster three attributes into two clusters (Cluster 1: "Gender" and "Income"; Cluster 2: "State"). That is, "Gender" and "Income" are merged into a new attribute "Gender-Income", and a combined dataset is formed. It is shown in Table 9.

After putting this combined dataset into C5, we get a compact decision tree with tree size 4 (Before applying the CAM model, the decision tree size is 6). Thus, it is obviously that the CAM model can actually discover the dependencies among attributes. In next section, we'll demonstrate the CAM model in several UCI datasets.

Table 9. Combined Dataset

Gender-Income	State	Target: Vehicle
Male-Low	NY	Car
Male-Low	CA	Car
Female-High	NY	Car
Female-High	NY	Car
Male-High	CA	Bike
Male-High	CA	Bike
Female-Low	NY	Bike
Female-Low	CA	Bike

Table 10. Experimental Results (Error Rate)

Dataset Name	#	Error Rate (%)			
		C5	CBA	ADT	CAM
Australia	690	14.69%	15.08%	14.49%	14.05%
Balance	625	21.53%	31.68%	20.32%	17.86%
Cars	392	02.71%	06.18%	07.94%	02.26%
Diabetes	768	25.15%	27.06%	26.14%	24.93%
Flare	1066	17.20%	19.88%	16.99%	17.10%
Iris	150	05.79%	08.00%	08.00%	04.81%
Monk-1	432	0.07%	00.00%	00.00%	00.00%
Monk-2	432	32.90%	23.66%	27.00%	32.90%
Monk-3	432	00.00%	02.88%	01.09%	00.00%
New-Thry	215	06.93%	05.60%	04.65%	05.49%
Nursery	10344	04.30%	01.87%	01.73%	04.30%
Post-Ope	90	30.33%	47.66%	28.88%	28.80%
Voting	290	10.28%	05.96%	05.05%	09.45%
Wine	178	07.28%	08.02%	06.86%	05.20%
Zoo	101	07.34%	06.00%	06.00%	03.93%
Average		12.43%	13.96%	11.67%	11.40%

4 Experimental Results

We evaluate CAM model using fifteen datasets from UCI database repository [12]. These datasets can be obtained from the following web site: http: //www1.ics.uci.edu /~mlearn/MLRepository.html. Based on these datasets, we compare our CAM model with C5, ADT, and CBA. In the experiments, we adopt the 5-fold cross validation scheme. Table 10 shows the error rate for all models, i.e., C5, CBA, ADT and CAM. The last row contains the average error rate of each model. From Table 10, CAM model is superior in several ways. First, CAM model has the lowest average error rate, i.e., 11.40%. Second, there are ten datasets that CAM model is better than the other models in error rate. Table 11 shows the rule size for all models. The last row contains the average rule size of each model. From Table 11, CAM model is also superior in several ways. First, CAM model has the smallest rule size, i.e., 27.47%. It means that CAM model can generate more compact rules. Second, there are twelve datasets that CAM model is better than the other models in rule size. From the above experiments, the CAM model remarkably improves both the classification accuracy and the decision tree size of C5. Interestingly, the CAM model also outperforms the existing association-based classification models, i.e., ADT and CBA.

Table 11. Experimental Results (Rule Size)

Dataset Name	#	Rule Size			
		C5	CBA	ADT	CAM
Australia	690	20.02	129.2	43.80	12.61
Balance	625	40.73	117.2	22.20	18.05
Cars	392	28.37	168.8	194.4	28.30
Diabetes	768	29.11	38.80	16.00	22.95
Flare	1066	01.50	28.80	01.00	01.00
Iris	150	04.71	05.60	04.60	04.17
Monk-1	432	27.92	10.20	15.00	27.84
Monk-2	432	01.00	117.8	111.8	01.00
Monk-3	432	14.00	16.60	04.20	13.50
New-Thry	215	08.10	11.40	11.20	08.10
Nursery	10344	269.0	411.6	269.8	250.0
Post-Ope	90	01.33	23.40	01.00	01.00
Voting	290	12.80	31.80	25.20	11.28
Wine	178	05.37	10.20	15.00	05.26
Zoo	101	08.36	08.60	15.20	07.00
Average		31.48	75.33	50.02	27.47

5 Concluding Remarks

This paper proposes a CAM (Classification based on Association Mining) model to deal with the problems of attribute dependencies. It combines the association rule mining technologies and the traditional decision-tree learning capabilities to handle the complicated and real cases. In experiments, we use the real datasets from UCI database repository. We compare CAM model with C5, ADT, and CBA. The experimental results show that the CAM model remarkably improves the classification accuracy and the decision tree size based on C5. At the same time, the CAM model also outperforms the existing association-based classification models, i.e., ADT (Association-based Decision Tree) and CBA (Classification Based on Association). However, this paper adopts the linkage diagram for combining attributes. The CAM model needs to perform several trials for searching the best result. How to reduce these additional trials need to be investigated further.

Acknowledgement

Research on this paper was partially supported by National Science Council grant NSC92-2213-E-130-007, NSC92-2622-E-130-002-CC3, NSC92-2213-E-030-019.

References

1. Chen, M.S., Han, J., Yu, P.S.: Data Mining: An Overview from a Database Perspective. IEEE Transaction on Knowledge and Data Engineering, Vol. 8, No. 6, (1996) 866-882.
2. Quinlan, J.R.: Improved Use of Continuous Attributes in C4.5. Journal of Artificial Intelligence Approach, Vol. 4, (1996) 77-90.
3. Lee, Y.S., Yen, S.J.: Neural-Based Approaches for Improving the Accuracy of Decision Trees. Proceedings of International Conference on Data Warehousing and Knowledge Discovery, (2002) 114-123.
4. Chen, M.S.: On the Evaluation of Using Multiple Attributes for Mining Classification Rules. Proceedings of IEEE International Conference on Tools with Artificial Intelligence, (1998) 130-137.
5. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. Proceedings of International Conference on Knowledge Discovery and Data Mining, (1998) 80-86.
6. Wang, K., Zhou, S.Q., He, Y.: Growing Decision Trees on Support-Less Association Rules. Proceedings of International Conference on Knowledge Discovery and Data Mining, (2000) 265-269.
7. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. Proceedings of ACM SIGMOD International Conference on Management of Data, (1993) 207-216.
8. Park, J.S., Chen, M. S., Yu, P.S.: Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 5, (1997) 813-825.
9. Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., Yang, D.: H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proceedings of IEEE International Conference on Data Mining, (2001) 441-448.
10. Han, J., Pei, J., Yin, Y., Mao, R.: Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery, Vol. 1, (2004) 53-87.
11. Kamber, M., Han J., Chiang, J.Y.: Metarule-Guided Mining of Multidimensional Association Rules Using Data Cubes. Proceedings of International Conference on Knowledge Discovery and Data Mining, (1997) 207-210.
12. Merz, C.J., Murphy, P.: UCI repository of machine learning databases. http://www.cs.uci. edu/mlearn/MLRepository.html, (1996).

OWDEAH: Online Web Data Extraction Based on Access History

Zhao Li[1], Wee-Keong Ng[1], and Kok-Leong Ong[2]

[1] Nanyang Technological University, Centre for Advanced Information Systems
Nanyang Avenue, N4-B3C-14, Singapore 639798
liz@pmail.ntu.edu.sg, awkng@ntu.edu.sg
[2] School of Information Technology, Deakin University
Waurn Ponds, Victoria 3217, Australia
leong@deakin.edu.au

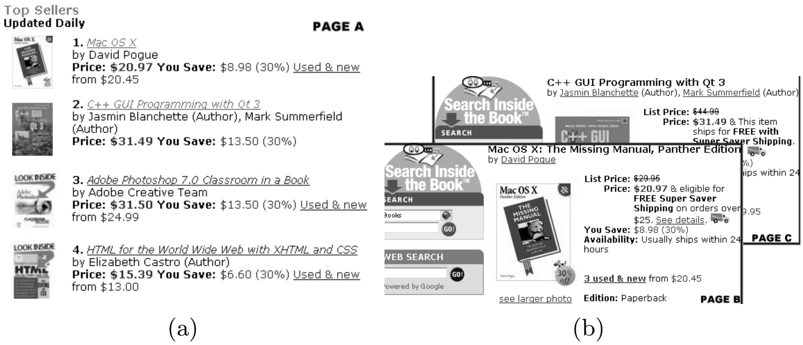
Abstract. Web data extraction systems are the kernel of information mediators between users and heterogeneous Web data resources. How to extract structured data from semi-structured documents has been a problem of active research. Supervised and unsupervised methods have been devised to learn extraction rules from training sets. However, trying to prepare training sets (especially to annotate them for supervised methods), is very time-consuming. We propose a framework for Web data extraction, which logged users' access history and exploit them to assist automatic training set generation. We cluster accessed Web documents according to their structural details; define criteria to measure the importance of sub-structures; and then generate extraction rules. We also propose a method to adjust the rules according to historical data. Our experiments confirm the viability of our proposal.

1 Introduction

In recent years, the volume of data on the Web has overran the capability of manual processing. Many information agents have appeared as mediators between users and heterogeneous Web data resources to assist users to manage data on the Web. The main objective of these agents is to extract structured data from semi-structured documents, such that data on the Web can be processed efficiently and automatically. For instance, if one extracts structured data from pages A, B and C in Figure 1 and store them in the relational tables as shown in Figure 2, then a query such as *"return the list prices of the top sellers"* can be very easily expressed in SQL and be automatically processed.

There is an observation that many semi-structured documents are generated by applying the same set of rules to the underlying structured database, e.g., *page B* and *C*. This suggests that some schemata exist in the set of documents. However, unlike structured data, those schemata in semi-structured documents are usually hidden from the users. How to detect the schemata from such documents and to figure out the informative portions is a core problem for Web data extraction systems. Figure 3(a) shows a typical framework for Web data

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 269–278, 2004.



Top Sellers
Updated Daily

PAGE A

1. Mac OS X
by David Pogue
Price: $20.97 You Save: $8.98 (30%) Used & new from $20.45

2. C++ GUI Programming with Qt 3
by Jasmin Blanchette (Author), Mark Summerfield (Author)
Price: $31.49 You Save: $13.50 (30%)

3. Adobe Photoshop 7.0 Classroom in a Book
by Adobe Creative Team
Price: $31.50 You Save: $13.50 (30%) Used & new from $24.99

4. HTML for the World Wide Web with XHTML and CSS
by Elizabeth Castro (Author)
Price: $15.39 You Save: $6.60 (30%) Used & new from $13.00

(a)

C++ GUI Programming with Qt 3
by Jasmin Blanchette (Author), Mark Summerfield (Author)
List Price: ~~$44.99~~
Price: **$31.49** & This item ships for **FREE with Super Saver Shipping**.

PAGE C

Mac OS X: The Missing Manual, Panther Edition
by David Pogue
List Price: ~~$29.95~~
Price: **$20.97** & eligible for **FREE Super Saver Shipping** on orders over $25. See details.
You Save: $8.98 (30%)
Availability: Usually ships within 24 hours
3 used & new from $20.45
Edition: Paperback

see larger photo

PAGE B

(b)

Fig. 1. Sample Pages: (a) Top selling books; (b) the details of each book.

BookTitle	Author	Price	...
Mac...	David...	20.97	...
C++...	Jasmin...	31.49	...
...	...	...	...

BookTitle	Author	ListPrice	...
Mac...	David...	29.95	...
C++...	Jasmin...	44.99	...
...	...	...	...

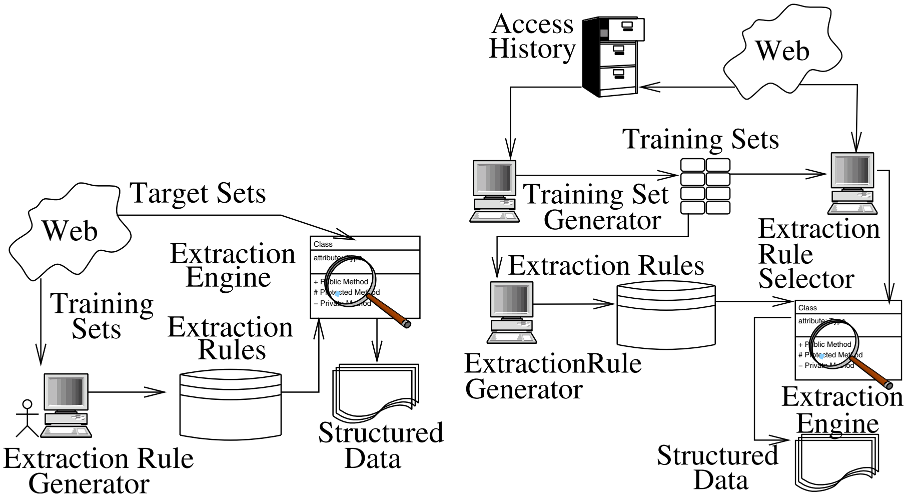
Fig. 2. Extracted data stored in relational database.

extraction systems. In this framework, the process of Web data extraction is abstracted into two phases. First, extraction rules are learnt from sets of training documents. Here, extraction rules include the knowledge of document schema and the know-how to extract structured data from each document. In the second phase, the learnt rules are applied to target documents (i.e., the new documents) to complete the remaining extraction. While there are many research to learn extraction rules from training sets, the task of preparing training sets and target sets is seldom addressed by current literature. In this paper, we suggest a framework that uses access history to generate document sets automatically. We also introduce methods that detect informative contents from documents to assist extraction rule generation.

Our paper is organized as follows: Section 2 presents a framework for Web data extraction. Section 3, introduces our approaches to prepare document sets, including training sets and target sets. Section 4 describes how extraction rules are learnt from generated training sets. Section 5 discusses the experimental results, and Section 6 presents our conclusions and future works.

2 OWDEAH's Framework

We begin with an introduction of our **OWDEAH** framework. OWDEAH is the acronym for *Online Web Data Extraction based on Access History*. In this framework, we analyze the access history of users – including Web documents accessed and those parts in a document that users see as important. When a document arrives, we select a set of documents that are most similar with the new



(a) (b)

Fig. 3. (a) a typical extraction frameworks; (b) the **OWDEAH** framework.

document from the historical data. Extraction rules learnt from this document set are then applied to the new document. This is what *Online* means, i.e., we can figure out extraction rules corresponding to the new document on-the-fly, instead of collecting a set of documents to train the extraction rule generator. Figure 3(b) shows the OWDEAHframework.

Currently, our target documents (to be extracted by our system) are restricted to those with a hidden schema. Such documents have some kind of schemata, i.e., the same type of data in the documents would be organized by the same set of structures. These schemata are hidden from the users. To discover these schemata, we need to collect documents containing records of the same type of information. For example, *page A* in Figure 1 contains information records of 4 books; and that both *page B* and *page C* have a record with the same type. In current systems, these documents are collected manually. However, it is very time-consuming and error-prone. To overcome this shortcoming, we prepare these documents by analyzing access history from users and cluster similar documents together automatically. The history may be a crawl history of crawlers, or the browse history of individual users.

Training Set Generator first detects common structures from history documents, then clusters documents based on these structures and each cluster will be maintained as a training set. Here we make an assumption that if two fragments in documents have similar structure, then they contain the same type of information in all probability. This is also the background assumption of most current methods; e.g., ExAlg [1], IEPAD [2], etc.

Given training sets generated and common structures detected, we devise some methods to measure the importance of these structures. Important structures will be encoded into extraction rules by **Extraction Rule Generator**. In this paper, we only discuss importance measurement. The detailed process of encoding is not covered in this paper; readers can refer to [3] about detailed introduction to extraction rules of our system.

So far, we have obtained a series of training sets and important structures corresponding to history documents. We introduce a unified method to represent a document or a document set as a vector of structures detected, and to measure the similarity among those vectors. Corresponding documents or document sets of most similar vectors will be treated as the most similar. Given an incoming document to be extracted, **Extraction Rule Selector** finds the most similar document set of the new document, and uses the corresponding extraction rules of this document set to execute extraction.

Extraction Engine is the very component in our system that extracts data from documents and transforms them into structured format. It interprets extraction rules applied to a given documents, decodes these rules to structures, matches these structures with structures included in the document and reorganizes those matched parts.

The comprehensive framework of OWDEAH is designed to cover more aspects in the whole process of Web data extraction. It is more serious to provide efficient and practical approaches to implement components than just to suggest the framework. In following sections, we will formally formulate problems to be solved and introduce our solutions.

3 Training Set Preparation

In OWDEAH, we cluster those structural similar documents together to prepare training sets. To collect structure information of semi-structured documents, e.g., HTML, XML documents, we first model them as trees, as defined below:

Definition 1. *A document tree is a rooted, labeled tree, which can be defined as a 4-tuple: $T = (V, E, r, \gamma)$, where V is the set of nodes, E is the set of edges and r is the root node, γ is a function that labels each node. An edge $e{=}(u, v)$, where u is the parent node of node v.*

It is straightforward to parse HTML and XML documents to document trees. We can use any HTML, XML parser to build DOM trees according to them. Each DOM tree is a document tree, in which γ labels each node with the name of corresponding element, i.e., we will label all text elements as the same. Some systems [4, 5] match text contents in text elements with some regular expressions, elements matched with the same expression are given the same label. In this paper, we will not consider details of these methods.

Some works [6, 7] addressed the problem of measuring similarity among documents using a tree structure, which is based on the concept of edit distance [8]. However, these methods do not consider common structures appearing in documents, which is very important to extract data. In the following sections, we shall introduce a method to measure document similarity based on common structures in documents.

Algorithm 1

```
1: Initialize an empty array of BETCs – $\mathcal{D}$. We use the subscript of a BETC
   as its class ID.
2: Read all leaf nodes in $F$. If any node $t$ does not belong to any BETC in $\mathcal{D}$,
   insert $BETC(t)$ to $\mathcal{D}$. Assign class ID to all those leaf nodes.
3: If all child nodes of a node $p$ in the forest have been read, read $p$. If $p$ does
   not belong to any BETC in $\mathcal{D}$, insert $BETC(p)$ to $\mathcal{D}$.
```

3.1 Common Structure Detection

We hope that documents in the same cluster have similar information. As website builders tend to use the same or similar structures to organize the same kind of information, we can assume that there exist many common subtrees in the document forest corresponding to a cluster. *"Common subtree"* is a concept with different definition in various papers [9, 10]. We will formally define a class of common subtrees that can represent Web documents very well. Before that, we give some preliminary definition.

Definition 2. *In a document forest F, an exact Equivalent Tree Class (ETC) is a set of subtrees that are isomorphic to one another. The ETC size of a class C, denoted with $\|C\|_{ETC}$, is equal to the cardinality of the node set of any tree belonging to C. If $t \in C$, we call t as an object of C. Each ETC can be identified with a unique integer ID, denoted as $ID(C)$. We assign this ID as identifier to object belonging to the ETC, which is so-called the* Class ID *of t, denoted as $id(t)$, i.e., if $t \in C$, then $id(t) = ID(C)$. A corresponding document tree with Class ID d is denoted as $tree(d)$, and we say it is a common subtree in F.*

Intuitively, we can measure the similarity between two documents by computing the ratio of the number of ETCs appearing in both documents to the number of ETCs appearing in anyone. However, the problem of to detect all common subtrees can be reduced to the problem of detecting the largest common subtrees from a forest (LCST problem), which is an NP-hard problem [11]. Some methods [9, 10] consider only those subtrees appearing with large frequency using data mining techniques. In our system, we only consider common subtrees appearing in a limited locality. Based on the observation that most contents to be extracted in Web document trees is near leaf nodes, our restriction may not lose much information, as defined below:

Definition 3. *Given a document forest F, a Bottom ETC (BETC) is an ETC, in which all leaf nodes of every object are also leaf nodes in F. If C is a BETC and tree $t \in C$ we say $BETC(t) = C$. We can represent $BETC(t)$ as $[\gamma(r), BETC(t_1), ..., BETC(t_n)]$, where t_1 to t_n are child trees of root node r.*

Given a forest F, we use Algorithm 1 to detect all common structures belonging to BETCs. In Definition 3, all common subtrees belonging to the same BETC are isomorphic to one another. This may make subtrees containing same kind of information to be inserted to different BETCs. For example, the first and the second records in *page A* both contain information of a book. However, the second entry does not contain secondhand price; according to Definition 3, corresponding subtrees will be assigned different class ID. To solve this problem, we define a class of common subtrees as below:

Definition 4. *Two BETCs* $C_1 = [\gamma(r_1), BETC(t_{1.1}), ..., BETC(t_{1.n})]$ *and* $C_2 = [\gamma(r_2), BETC(t_{2.1}), ..., BETC(t_{2.n})]$. *Let* $S_1 = \{BETC(T_{1.1}) ,..., BETC(t_{1.n})\}$, $S_2 = \{BETC(T_{2.1}) ,..., BETC(t_{2.n})\}$, S_{un} *is the union of* S_1 *and* S_2, S_{in} *is the intersection of* S_1 *and* S_2. *Given a value between (0,1), we say* C_1 *and* C_2 *belong to the same Approximate BETC (ABETC), if* $\sum_{C_i \in S_{in}} \|C_i\|_{ETC} > s \times \sum_{C_i \in S_{un}} \|C_i\|_{ETC}$

To detect all those common subtrees belonging to ABETCs and assign class ID to them, we only need to modify Step 3 in Algorithm 1: *If all child nodes of a node p in the forest have been read, read p. If p does not belong to any BETC that belongs to the same ABETC as anyone in* $\mathcal{D}$, *insert BETC(p) to* $\mathcal{D}$.

3.2 Clustering Based on Document Structure

After detecting all ABETCs C_1 to C_n from a forest, we give a class weight to each ABETC in each document tree as:

$$\omega(C_i) = \ln(|C_i|/max_{j=1}^{n}|C_j|) \times \|C_i\|_{ETC} / max_{j=1}^{n} \|C_j\|_{ETC} \tag{1}$$

where $|C_i|$ is the cardinality of C_i. Equation 1 reflects those classes with larger ETC size have larger weight and classes containing more objects have larger weight. So far, we may represent a document as a vector of ABETCs $[\omega(C_1), \ldots, \omega(C_n)]$ and the similarity between two document trees can be measured by computing the Cosine Similarity between two vectors:

$$\mathbb{S}(T_1, T_2) = \frac{\sum_{i=1}^{k}(\omega_1(C_i) \times \omega_2(C_i))}{\sqrt{\sum_{i=1}^{k}\omega_1(C_i)^2 \times \sum_{i=1}^{k}\omega_2(C_i)^2}}$$

where $\omega_1(C_i)$ and $\omega_2(C_i)$ are class weight of C_i in document tree 1 and 2 respectively. In the similarity calculation phase, we can obtain similarity square matrix $\mathbb{M}_{m \times m}$, where m is the number of document trees and $\mathbb{M}_{i,j}$ is the similarity between document i and document j. Given the similarity matrix, we choose bisecting k-means clustering algorithm, implemented in **CLUTO** toolkit [12, 13], which can cluster documents into k clusters, from $\mathbb{C}_1$ to $\mathbb{C}_k$, such that the maximum value of the following is obtained

$$\sum_{i=1}^{k}\sqrt{\sum_{T_1, T_2 \in \mathbb{C}_i} \mathbb{S}(T_1, T_2)}$$

where $\mathbb{S}(T_1, T_2)$ is the similarity between document tree T_1 and T_2.

4 Extraction Rule Generation and Selection

In our system, extraction rules are encoded subtrees that are stored in XML format [3]. In last section, we introduced how to detect common subtrees from document trees and how to cluster document trees to generate training sets. We can analyze training sets and select those important common subtrees to encode them to extraction rules. In this section, we introduce how to measure the importance of common subtrees. We also introduce how to use feedback from users to adjust this importance value.

In Figure 1, we suppose each record in *page A, B, C* is organized using a tree – *"book"*; page title in *page A* is organized using a tree – *"title"*; each search box in *page C, D* is organized using a tree – *"search"*. Obviously, book information is the most important in those pages. If we cluster these three documents in Figure 1 together as a training set, let us consider how to figure out the subtree *"book"* is the most important.

In our system, we suggest a method that represents the relation between common subtrees and documents as weighted graph and uses link analysis techniques like HITS [14] to figure out important common subtrees. We introduced how to use a vector of weighted common subtrees to represent a document, e.g., those documents in Figure 4(a). Thus, we can represent all documents in a training set as a matrix. For example, *page A, B, C* can be represented using the matrix in Figure 4(b). We build an auxiliary matrix M_w as that in Figure 4(c), where $M^{'}$ is the transposed matrix of M. We see that M_w is the adjacency matrix of the weighted graph in Figure 4(d). In this graph, if a common subtree appears in a document, then there is an edge between them, which is weighted with the weight of the subtree in the document.

$$\begin{aligned} &\text{page A}=[1\ 1\ 0]\\ &\text{page B}=[0\ 1\ 1]\\ &\text{page C}=[0\ 1\ 1]\end{aligned}$$

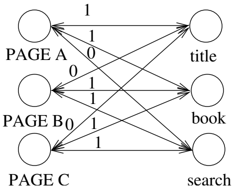
(a)

$$M = \begin{bmatrix} 1\,1\,0 \\ 0\,1\,1 \\ 0\,1\,1 \end{bmatrix}$$

(b)

$$M_w = \begin{bmatrix} 0 & M^{'} \\ M & 0 \end{bmatrix}$$

(c)



(d)

Fig. 4. The Relation among Common Trees and Documents: (a) documents; (b) document space; (c) auxiliary matrix; (d) link relation.

So far, we represent relation among common subtrees and documents as a weighted graph. We know the basic idea of most graph link analysis, such as HITS [14] and PageRank [15], methods is to compute principal eigenvector of corresponding matrix, and to denote vectors corresponding to the largest items in the eigenvector as the most important. The different point is how to assign weights to elements in the matrix. We compute weight using Equation 1.

Having generated the auxiliary matrix, we calculate its principal eigenvector. The rank result of eigenvector can be treated as rank of the importance of corresponding common subtrees. In the matrix in Figure 4(c), we find that *"book"* ranks first in the principal eigenvector. We adjust this rank based on the idea that we can emphasize a node in the graph by adding links or improving the weight of its input links. In our system, we provide an interface that allows users to do the feedback. For example, in Figure 1, users access book information much more than the search box. We will analyze users' access history and improve the weight of incoming links to *"book"* accessed with high frequency. Thus, we obtain a new importance rank of common subtrees. Based on this rank list, we encode a given ratio of common subtrees to extraction rules.

Given some document sets, we can classify a document tree to a set that is most similar with it. In our system, when a new incoming document arrives, we can classify it to a training set, and use the extraction rules learnt from the training set to extract data from this document. We can refer to XRules [10] for some initial works of documents classification based on tree structure of documents. XRules mines some rules like $t \Rightarrow \mathbb{C}$, which describes the relation between appearance of a tree structure t in a document and the document belongs to class $\mathbb{C}$. During classification phase, XRules combine the evidences of structures appearing in a document and suggest a class for it.

However, XRules does not consider the occurrences of structure T in documents to be classified. Here, we suggest a generic method: From the results of clustering phase, we can use a vector $\mathbb{W}_{\mathbb{C}} = [\omega_{\mathbb{C}}(C_1), ..., \omega_{\mathbb{C}}(C_n)]$ to represent a cluster, where $\omega_{\mathbb{C}}(C_i)$ is the weight of ABETC C_i in cluster $\mathbb{C}$. Previously, we also introduced how to represent a document tree using a weight vector. Assume that $\mathbb{W}_T = [\omega_T(C_1), ..., \omega_T(C_n)]$ is the weight vector of document tree T, the similarity between a document tree and a cluster is:

$$\mathbb{S}(T, \mathbb{C}) = \frac{\sum_{i=1}^{k} (\omega_T(C_i) \times \omega_{\mathbb{C}}(C_i))}{\sqrt{\sum_{i=1}^{k} \omega_T(C_i)^2 \times \sum_{i=1}^{k} \omega_{\mathbb{C}}(C_i)^2}} \tag{2}$$

5 Experiments

We conducted some experiments to show the practicability of our framework. To set up the experiments, we choose datasets used by some important Web data extraction systems – **ExAlg**[1], **IEPAD**[2], **RISE**[3], **RoadRunner**[4], **WIEN**[5]. These datasets collect Web documents from various websites and each document can be assigned a category. We mix up documents from various categories to generate **MIX** dataset. We also crawled `Amazon.com` to get some documents and to generate the **ETC** dataset.

[1] http://www-db.stanford.edu/~arvind/extract/

[2] http://140.115.156.70/iepad/

[3] http://www.isi.edu/muslea/RISE/

[4] http://www.dia.uniroma3.it/db/roadRunner/index.html

[5] http://www.cs.ucd.ie/staff/nick/home/research/wrappers/

Table 1. Results of our accuracy experiments.

D	T	CLUTO
E1	90%	47.5%
R1	100%	66.7%
E2	100%	100%
W	100%	72.5%
R2	89.6%	70.8%
All	95.7%	71.2%

(a)

DataSet	E1	E2	W	R1	R2	Total
T(%)	97.5	100	100	100	100	99.5
R(%)	77.08	86.78	89.58	89.58	87.66	86.14

(b)

We chose the bisecting k-means algorithm to cluster our documents. Table 1(a) shows the results of our accuracy test obtained from the MIX dataset, which is a set of documents drawn from datasets including: E1 (ETC), E2 (ExAlg), W (WIEN), R1 (RISE), and R2 (RoadRunner). We used CLUTO's implementation of bisecting k-means on the text documents. Our choice for CLUTO is its suitability for clustering free texts [13]. From the table, except for documents from ExAlg, our methods outperforms CLUTO. On the instance that CLUTO performs better than our method, it is due to the presence of long paragraphs in the documents from ExAlg. Meanwhile, the ETC dataset are crawled from Amazon.com, and the structures of these documents are complicated without long paragraphs. In this case, CLUTO perform poorly.

Table 1(b) lists the comparison results of Rainbow[6] and our classification method. Row 1 indicates those datasets: E1 (ETC), E2 (ExAlg), W(WIEN), R1 (RISE), R2 (RoadRunner). The second row and the third row are results of our method and Rainbow respectively. Rainbow is a toolkit for free-text document classification. We performed a 4-fold cross-validation strategy to evaluate Rainbow and record its average accuracy. The accuracy results obtained by Rainbow vary from 77.08% to 89.58%. On the other hand, our method classifies documents based on Equation 2 and delivers 100% accuracy on all dataset except ETC, and the average accuracy of our method is about 13% higher than Rainbow.

On the ETC dataset that includes documents with complicated structures, Rainbow's results are poor; although in other datasets it achieves an accuracy higher than 85%. Our method is almost unaffected by the differences among datasets. On the ETC dataset, one page was classified to the error class by our method. This is a page linking other sub-categories, which makes it difficult to judge its class.

6 Conclusion

In this paper, we present a similarity measure based on the structural information embedded in semi-structured documents. We introduced an efficient technique

[6] http://www-2.cs.cmu.edu/~mccallum/bow/

to discover frequent common structures embedded in Web documents by clustering and classifying the results using this similarity measure. Experimental results show that our approach performs well across different real-world data sets. Unlike present Web data extraction systems, our method allows training data preparation and Web data extraction to be integrated into one coherent system. As part of our future work, we plan to improve our system by considering the semantic information embedded in the Web documents.

References

1. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: Proc. of the 2003 ACM SIGMOD. (2003) 337 – 348
2. Chang, C.H., Lui, S.C.: IEPAD: information extraction based on pattern discovery. In: Proc. of the 10th WWW, Hong Kong (2001) 681 – 688
3. Liu, Z., Li, F., Ng, W.K.: WICCAP data model: Mapping physical websites to logical views. In: Proc. of the 21st International Conference on Conceptual Modelling. (2002)
4. Baumgartner, R., Flesca, S., Gottlob, G.: Visual web information extraction with lixto. In: Proc. of 27th VLDB. (2001) 119–128
5. Rajaraman, A., Ullman, J.D.: Querying websites using compact skeletons. In: Proc. of the 12th PODS. (2001) 16 – 27
6. Wang, J.T.L., Zhang, K.: Finding similar consensus between trees: An algorithm and a distance hierarchy. Pattern Recognition **34** (2001) 127–137
7. Zhang, K., Statman, R., Shasha, D.: On the editing distance between unordered labeled trees. Information Processing Letters **42** (1992) 133–139
8. Tai, K.C.: The tree-to-tree correction problem. Journal of the ACM **26** (1979) 422 – 433
9. Wang, K., Liu, H.: Discovering structural association of semistructured data. IEEE Transactions on Knowledge and Data Engineering **12** (2000) 353–371
10. Zaki, M.J., Aggarwal, C.C.: Xrules: An effective structural classifier for xml data. In: Proc. of the 9th SIGKDD. (2003) 316 – 325
11. Akutsu, T.: An RNC algorithm for finding a largest common subtree of two trees. IEICE Trans. Information and Systems **E75-D** (1992) 95–101
12. Karypis, G.: A clustering toolkit. Technical Report TR#02-017, Univ. of Minnesota (2002)
13. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of KDD Workshop on Text Mining. (2000)
14. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46** (1999) 604–632
15. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks **30** (1998) 107–117

Data Mining Approaches to Diffuse Large B–Cell Lymphoma Gene Expression Data Interpretation

Jesús S. Aguilar-Ruiz[1], Francisco Azuaje[2], and José C. Riquelme[3]

[1] University of Seville, Seville, Spain
aguilar@lsi.us.es
[2] University of Ulster, North Ireland
fj.azuaje@ulster.ac.uk
[3] University of Seville, Seville, Spain
riquelme@lsi.us.es

Abstract. This paper presents a comprehensive study of gene expression patterns originating from a diffuse large B–cell lymphoma (DLBCL) database. It focuses on the implementation of feature selection and classification techniques. Thus, it firstly tackles the identification of relevant genes for the prediction of DLBCL types. It also allows the determination of key biomarkers to differentiate two subtypes of DLBCL samples: *Activated B–Like* and *Germinal Centre B–Like DLBCL*. Decision trees provide knowledge–based models to predict types and subtypes of DLBCL. This research suggests that the data may be insufficient to accurately predict DLBCL types or even detect functionally relevant genes. However, these methods represent reliable and understandable tools to start thinking about possible interesting non–linear interdependencies.

1 Introduction

Lymphomas are divided into two general categories: Hodgkin's disease (HD) and non–Hodgkin's lymphoma (NHL). Over the past 20 years HD rates have declined, accounting for only 1% of all cancer. By contrast, NHL cases have increased by more than 50% during the same period in the United States [10]. They represent 4% of all cancer cases, becoming the fifth most common malignancy in that country. An analysis of NHL incidence trends between 1985 and 1992 in seven European countries showed an average increase of 4.2% per year, in the absence of an increase in the incidence of HD. In Spain, their death rate per 100.000 people during the periods 1965–69 and 1995–98 increased 212.7% for men and 283.9% for women [13]. These figures reveal the significance of developing advanced diagnostic and prognostic systems for these diseases.

In the last two decades, a better understanding of the immune system and the genetic abnormalities associated with NHL have led to the identification of several previously unrecognized types of lymphoma. However, this is a complex and expensive task. For instance, distinctions between Burkitt's lymphoma

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 279–288, 2004.

and diffuse large B–cell lymphoma (DLBCL) often prove to be difficult, since High–grade B–cell Burkitt–like lymphoma appears to be clinically similar to DLBCL [17]. DLBCL is the most common type of NHL. There are no reliable morphological or immunohistochemical indicators that can be used to recognize subtypes of DLBCL. Moreover, it is fundamental to develop treatments specially tailored to the individual requirements and profiles of patients [9].

Others authors, such as Shipp et al. [16], have studied the distinction between DLBCL and a related germinal centre B–cell lymphoma, known as follicular lymphoma (FL), which over the time acquires morphological and clinical features observed in DLBCLs.

Technological advances now allow to screen the expression of thousands of genes in parallel. Thus, gene expression profiling has become crucial for the development of powerful diagnostic and prognostic methods for these types of cancers. It offers an opportunity to characterise the biological behaviour of specific tumours. Such an approach, which is based on the use of micro–array techniques, may provide massive amounts of information. Therefore, there is a need for sophisticated algorithms capable of extracting novel and useful knowledge from a biomedical point of view. In fact, gene–expression profiling is thought to be a revolutionary technique to support the diagnosis of cancers [1].

A basic approach to the study of expression data consists of applying traditional statistical techniques. In many problems these methods have shown to be unable to extract relevant knowledge from data. Thus, the Knowledge Discovery in Databases approach (KDD) [8] represents a useful framework for addressing the challenges of gene expression analysis. In particular, feature selection techniques may significantly support diagnostic studies, based on the identification of relevant genes or biomarkers. *Clustering* is another important task within KDD, which aims to organize the information in terms of their similarity patterns [6]. *Supervised classification* has also become an important goal in this type of studies [5]. Techniques such as *decision trees* [4] or *decision lists* [15] may represent more effective and understandable tools for aiding in the prediction of types or subtypes of diseases.

In this paper, we carry out a broad study of a well–known database generated by Alizadeh et al. [2], who investigated the identification of lymphomas and DLBCL subtypes based on expression patterns. The data comprise 96 samples described by the expression values of 4026 genes.

Firstly, feature selection techniques are implemented to identify relevant genes for the prediction of lymphoma cancer types. Moreover, it allows the detection of biomarkers to distinguish two subtypes of DLBCL: *Activated B–Like* and *Germinal Centre B–Like Lymphomas.* The following methods have been implemented: Information gain criterion, based on the entropy measure, the Relief method and $\chi 2$ ranking and filtering. Decision trees are constructed to perform classification tasks initially based on the original classes, and afterwards using the DLBCL subtypes.

This study reveals that not only the genes identified by Alizadeh et al. are relevant for the prediction of the two subtypes of DLBCL, but many others groups

Table 1. Most relevant genes provided by Relief, InfoGain and χ^2, ordered by ranking.

ReliefF	Freq.	**InfoGain**	Freq.	$\boldsymbol{\chi^2}$	Freq.
GENE1610X	• • •	GENE707X	• •	GENE2400X	• • •
GENE1636X	•	GENE655X	• • •	GENE788X	• •
GENE1648X	•	GENE694X	• • •	GENE3639X	• •
GENE1622X	• • •	GENE1622X	• • •	GENE707X	• •
GENE1702X	•	GENE844X	• •	GENE655X	• • •
GENE653X	• • •	GENE1635X	• •	GENE1992X	•
GENE1637X	•	GENE2400X	• • •	GENE1675X	• •
GENE712X	• •	GENE1610X	• • •	GENE694X	• • •
GENE1607X	•	GENE717X	• •	GENE3767X	•
GENE611X	•	GENE711X	•	GENE769X	• •
GENE1647X	•	GENE639X	• •	GENE2387X	• •
GENE708X	• •	GENE2402X	• •	GENE1622X	• • •
GENE1651X	•	GENE769X	• •	GENE1610X	• • •
GENE2402X	• •	GENE641X	•	GENE2032X	•
GENE537X	•	GENE628X	•	GENE467X	• •
GENE1658X	•	GENE669X	•	GENE3685X	•
GENE654X	• •	GENE2403X	• • •	GENE2403X	• • •
GENE1608X	•	GENE647X	•	GENE1371X	•
GENE2393X	•	GENE712X	• •	GENE2033X	•
GENE1641X	•	GENE783X	• •	GENE646X	• •
GENE721X	•	GENE653X	• • •	GENE753X	• • •
GENE651X	• •	GENE691X	•	GENE783X	• •
GENE1644X	•	GENE753X	• • •	GENE764X	•
GENE1635X	• •	GENE2495X	•	GENE639X	• •
GENE753X	• • •	GENE651X	• •	GENE2428X	•

may also be considered as relevant markers. In general, KDD techniques demonstrate to be efficient for extracting valid and useful knowledge from biomedical data.

2 Feature Selection for Gene Expression Data

Feature subset selection is the process of identifying and removing irrelevant or redundant attributes. Decreasing the dimensionality of the data reduces the size of the hypothesis space and allows learning algorithms to operate faster and more effectively. It leads to smaller and easy–to–understand knowledge models of the target concept. Feature selection techniques produce ranked lists of attributes, providing the data analyst with insight into their data by clearly demonstrating the relative merit of individual attributes.

In this study, we used three feature selection techniques, both belonging to the *filter* category [7], *Information Gain Attribute Ranking*, ReliefF [11, 12] and χ^2 [14]. The information gain attribute ranking is often used where the sheer dimensionality of the data precludes more sophisticated attribute selection techniques, as the case being investigated here, which consist of 4026 attributes. ReliefF works by randomly sampling an instance from the data and then locating its nearest neighbour from the same and a different class. When dealing with noisy data the k nearest neighbours should be obtained. If the data contains multiple classes, the contributions of the k nearest neighbours can be weighted using the prior probabilities associated with each class. The values of the attributes of the nearest neighbours are compared to the sampled instance and used to up-

Table 2. Most relevant genes provided by Relief, InfoGain and χ^2, ordered by ranking.

ReliefF	Freq.	**InfoGain**	Freq.	χ^2	Freq.
GENE717X	● ●	GENE467X	● ●	GENE2202X	●
GENE2403X	● ● ●	GENE646X	● ●	GENE2199X	● ●
GENE2270X	●	GENE3639X	● ●	GENE844X	● ●
GENE784X	●	GENE2395X	● ●	GENE777X	● ●
GENE2486X	●	GENE2668X	● ●	GENE654X	● ●
GENE1603X	●	GENE788X	● ●	GENE1990X	●
GENE2489X	●	GENE1672X	●	GENE2424X	● ●
GENE703X	●	GENE2379X	●	GENE276X	●
GENE692X	●	GENE770X	● ●	GENE2862X	●
GENE2271X	●	GENE648X	●	GENE794X	●
GENE2401X	●	GENE642X	●	GENE770X	● ●
GENE1653X	●	GENE593X	●	GENE768X	●
GENE1646X	●	GENE1606X	●	GENE2778X	●
GENE2244X	●	GENE734X	●	GENE3764X	●
GENE694X	● ● ●	GENE604X	●	GENE2395X	● ●
GENE655X	● ● ●	GENE777X	● ●	GENE2374X	● ●
GENE538X	●	GENE1673X	●	GENE1324X	●
GENE731X	●	GENE2374X	● ●	GENE1343X	●
GENE2668X	● ●	GENE2199X	● ●	GENE2795X	●
GENE584X	●	GENE649X	●	GENE653X	● ● ●
GENE1776X	●	GENE708X	● ●	GENE1320X	●
GENE713X	●	GENE1675X	● ●	GENE3334X	●
GENE2400X	● ● ●	GENE2387X	● ●	GENE2000X	●
GENE710X	●	GENE2424X	● ●	GENE473X	●
GENE714X	●	GENE706X	●	GENE1323X	●

date relevance scores for each attribute. The rationale is that a useful attribute should differentiate instances from different classes, and has the same value for instances belonging to the same class. χ^2 statistic conducts a significance test on the relationship between the values of an attribute and the classes.

This study presents results based on the original set of genes (4026), and on a subset of 50 relevant genes extracted from them, in order to compare our result to that identified by Alizadeh et al. [2].

2.1 Lymphoid Malignancies

The three methods provided different results and they were compared to find coincidences. Tables 1 and 2 show the genes selected by each method, and ordered by their relevance from top to bottom. The genes are represented with the identifiers used by Alizadeh et al. Figure 1 provides the gene names associated with each identifier. We found 8 common genes for the three methods, 25 common pairs of genes and 76 genes that were selected by only one method. Note that there are 105 different genes in Tables 1 and 2, from 150 possible selections.

Figure 1 shows these genes in terms of degrees of relevance: Very high and high relevance. *Very highly relevant* genes are those which have been selected by all of the feature selection methods. *Highly relevant* genes are those which have been identified by any two of the methods. In total, there are 8 very highly relevant genes and 25 highly relevant genes.

The next task is to know whether these 8 very highly relevant genes or, in the worst case, the 33 genes including the highly relevant ones, can predict a cancer class. This will be shown using decision trees in section 3. Alizadeh et al.

	ID	Name
VERY HIGH RELEVANCE	GENE653X	Lactate dehydrogenase A
	GENE655X	GRSF-1=cytoplasmic G-rich mRNA sequence binding factor
	GENE694X	Cyclin A
	GENE753X	Similar to MCM2 = DNA replication licensing factor
	GENE2400X	Unknown
	GENE2403X	Unknown
	GENE1610X	Mig=Humig=chemokine targeting T cells
	GENE1622X	CD63 antigen (melanoma 1 antigen)
HIGH RELEVANCE	GENE467X	C-1-Tetrahydrofolate Synthase, cytoplasmic
	GENE639X	hepatoma-derived growth factor
	GENE646X	nm23-H2=NDP kinase B=Nucleoside dephophate kinase B
	GENE651X	tubulin-beta
	GENE654X	dystrobrevin B DTN-B2=dystrophin-associated protein A0
	GENE707X	Topoisomerase II alpha (170kD)
	GENE708X	Ki67 (long type)
	GENE712X	Cyclin B1
	GENE717X	aurora/IPL1-related kinase
	GENE769X	14-3-3 epsilon
	GENE770X	14-3-3 epsilon
	GENE777X	semaphorin V=homologue of nerve growth cone guidance signaling proteins
	GENE783X	Glycyl tRNA synthetase
	GENE788X	SRPK1=serine kinase
	GENE844X	ets-2=ets family transcription factor
	GENE1635X	osteonectin=SPARC=basement membrane protein
	GENE1675X	FCERI=Fc epsilon receptor gamma chain=High affinity immunoglobulin epsilon receptor gamma
	GENE2199X	Unknown UG Hs.71252 ESTs
	GENE2374X	PKC beta =Protein kinase C, beta
	GENE2387X	Unknown UG Hs.181297 ESTs
	GENE2395X	Unknown UG Hs.59368 ESTs
	GENE2402X	Unknown
	GENE2424X	Similar to neuropathy target esterase
	GENE3639X	KIAA0053
	GENE2668X	Mad2=MXI-1=MAX-binding protein=antagonizer of myc transcriptional activity= =Mxi-1/Max heterodimers repress c-myc targets

Fig. 1. Relevant genes to differentiate the lymphoma classes from the complete dataset (96 patients and 4026 attributes). The 50 most relevant genes for each feature selection method were selected. *very highly relevant* means that the gene was relevant for the three methods; *highly relevant* means that it was relevant for any two methods. There is no order of relevance in the list.

discovered 50 relevant genes to differentiate the GC B–Like from the Activated B–Like subtypes of DLBCL. However, only one of them, GENE2395X, has been identified in our analysis.

It is important to note that several attributes are strongly correlated with others (Pearson's correlation coefficient greater than 0.90), even among those genes selected as very highly or highly relevant in Figure 1. This fact shows that most methods for feature selection do not take into account the correlation among extracted features. Therefore, this information needs to be post–processed, removing genes of similar functionality. For instance, GENE769X and GENE770X, shown in Figure 1, or GENE1719X and GENE1720X, in Figure 2.

2.2 DLBCL Subtypes

In this section the 45 DLBCL samples are analysed. This category is divided into to subtypes: Activated B–like DLBCL (ACL) and Germinal Centre B–like

	ID	Name
VERY HIGH RELEVANCE	GENE1296X	MCL1=myeloid cell differentiation protein
	GENE1719X	TTG-2=Rhombotin-2=translocated in t(11;14)(p13;q11) T cell acute lymphocytic leukemia=cysteine rich protein with LIM motif
	GENE1720X	TTG-2=Rhombotin-2=translocated in t(11;14)(p13;q11) T cell acute lymphocytic leukemia=cysteine rich protein with LIM motif
	GENE3228X	JNK3=Stress-activated protein kinase
	GENE3254X	Unknown UG Hs.145058 ESTs
	GENE3255X	Unknown
	GENE3256X	JAW1=lymphoid-restricted membrane protein
	GENE3258X	JAW1=lymphoid-restricted membrane protein
	GENE3259X	Unknown UG Hs.124922 ESTs
	GENE3261X	Unknown
	GENE3314X	Unknown
	GENE3315X	FMR2=Fragile X mental retardation 2=putative transcription factor=LAF-4 and AF-4 homologue
	GENE3318X	CD10=CALLA=Neprilysin=enkepalinase
	GENE3325X	Unknown UG Hs.120245 Homo sapiens mRNA for KIAA1039 protein, partial cds
	GENE3326X	Unknown UG Hs.105261 EST
	GENE3327X	Unknown UG Hs.169565 ESTs, Moderately similar to !!!! ALU SUBFAMILY SB WARNING ENTRY !!!! [H.sapiens]
	GENE3328X	Unknown UG Hs.136345 ESTs
	GENE3329X	Unknown UG Hs.224323 ESTs, Moderately similar to alternatively spliced product using exon 13A [H.sapiens]
	GENE3330X	Unknown
	GENE3331X	Unknown UG Hs.208410 EST, Moderately similar to !!!! ALU SUBFAMILY SB WARNING ENTRY !!!! [H.sapiens]
	GENE3332X	Unknown UG Hs.120716 ESTs
	GENE3335X	myb-related gene A=A-myb
	GENE3355X	Unknown
	GENE3939X	Unknown UG Hs.169081 ets variant gene 6 (TEL oncogene)
	GENE3968X	Deoxycytidylate deaminase

Fig. 2. Relevant genes to differentiate Activated B–like DLBCL from Germinal Centre B–like DLBCL (45 patients and 4026 attributes). The 50 most relevant genes for each feature selection method were selected. There is no order of relevance in the list.

DLBCL (GCL). Among these 45 examples, 22 belong to class GCL and 23 to class ACL.

The Relief algorithm, InfoGain and χ^2 methods have been applied to select the most relevant attributes to differentiate these sub–classes. These methods provided 25 common attributes, which will be considered as very high relevant, and they are enumerated in Figure 2.

Figures 1 and 2 do not include genes in common. It suggests that the genes required to differentiate among types of lymphoma cancer may be different to those distinguishing DLBCL subtypes. Nevertheless, several genes which experts had identified as having some functionality associated with lymphoma, are present in the subset, among them, TTG–2 and CD10. Furthermore, others like MCL1, JNK3 or FMR2, have not been linked to DLBCL.

3 Decision Trees

Decision trees are a useful technique in the context of supervised learning. They perform classification by a sequence of tests whose semantics are intuitively clear and easy to understand. Some tools, like J48, construct decision trees selecting the best attribute by using a statistical test to determine how well it alone classifies the training examples. Our experiments were performed by using the WEKA library for machine learning [18].

To avoid over–estimating the prediction accuracy that occurs when a model is trained and evaluated with the same samples, the "leave–one–out" testing method has been used. In this case 96–fold cross–validation and 45–fold cross–validation procedures are implemented when the lymphoma types and DLBCL subtypes are analysed respectively.

```
GENE1602X <= -0.44
| GENE2426X <= 0.59
| | GENE3959X <= 0.33: FL
| | GENE3959X > 0.33: GCB
| GENE2426X > 0.59: CLL
GENE1602X > -0.44
| GENE563X <= -0.52
| | GENE3701X <= -1.2: RAT
| | GENE3701X > -1.2: RBB
| GENE563X > -0.52
| | GENE717X <= -0.5: ABB
| | GENE717X > -0.5
| | | GENE694X <= 1.54: DLBCL
| | | GENE694X > 1.54: TCL

Tree Size = 15
Number of genes = 7 (from 4026)
Training Error = 2.08%
96-Fold CV Error = 20.84%
```

(a)

```
GENE646X <= -0.61
| GENE844X <= -1.02: CLL
| GENE844X > -1.02
| | GENE2387X <= -0.68: RAT
| | GENE2387X > -0.68
| | | GENE2374X <= 0.61: FL
| | | GENE2374X > 0.61: RBB
GENE646X > -0.61
| GENE717X <= -0.5
| |  GENE3639X <= -0.28: NIL
| |  GENE3639X > -0.28: ABB
| GENE717X > -0.5
| | GENE694X <= 1.54
| | | GENE2402X <= 1.45: DLBCL
| | | GENE2402X > 1.45: FL
| | GENE694X > 1.54: TCL

Tree Size = 17
Number of genes = 8 (from 33)
Training Error = 5.38%
96-Fold CV Error = 26.05%
```

(b)

```
GENE1622X <= -1.17
| GENE753X <= 0.61
| | GENE753X <= -0.88
| | | GENE653X <= -2.24: CLL
| | | GENE653X > -2.24
| | | | GENE655X <= -1.34: RBB
| | | | GENE655X > -1.34: CLL
| | GENE753X > -0.88: FL
| GENE753X > 0.61
| | GENE694X <= 0.6: RAT
| | GENE694X > 0.6
| | | GENE2403X <= 0.06: DLBCL
| | | GENE2403X > 0.06: GCB
GENE1622X > -1.17
| GENE694X <= -0.83: ABB
| GENE694X > -0.83
| | GENE694X <= 1.54
| | | GENE1610X <= -0.79: ABB
| | | GENE1610X > -0.79: DLBCL
| | GENE694X > 1.54: TCL

Tree Size = 21
Number of genes = 7 (from 8)
Training Error = 6.79%
96-Fold CV Error = 19.80%
```

(c)

Fig. 3. Decision trees to differentiate lymphoma classes. All of the decision trees were generated using 96 patients. In (a) 4026 genes were used (the whole set); in (b) 33 genes (very high and high relevant); and in (c) only 8 genes (very high relevant). Labels were assigned to each type of lymphoma cancer, based on the study by Alizadeh et al.: Diffuse large B–cell lymphoma (DLBCL), Germinal centre B (GCB), NI. lymphoma node/tonsil (NIL), Activated blood B(ABB), Resting/activated T (RAT), Transformed cell lines (TCL), Follicular lymphoma (FL), Resting blood B (RBB) and Chronic lymphocytic leukaemia (CLL).

3.1 Prediction of Lymphoid Malignancies

Three decision trees were generated to differentiate among types of lymphoma. The first decision tree algorithm (Figure 3a) used the complete set of genes as input. However, the resulting tree comprises only 7 genes, producing an error rate of 2.08% (training set as test set), and 20.84% (leave–one–out method). The second decision tree (Figure 3b) selected 8 genes among the 33 genes extracted by the feature selection methods, making an error rate of 5.28% (training) and 26.05% (leave–one–out). The third decision tree (Figure 3c) provides an error rate of 6.79% (training) and 19.80% (leave–one–out). Thus, from a prediction point of view, only those genes categorized as very higly relevant allow the generation of the best decision trees.

The gene GENE694X (cyclin A) seems to be decisive in the prediction of lymphoma types, as it is the only one that appears in all of the decision trees. In the first one, differentiates DLBCL from TCL; in the second one, DLBCL and FL from TCL; and in the third one, RAT from DLBCL and GCB, and ABB and TCL from DLBCL (although the gen GENE1610X plays an important role to separate ABB from DLBCL). This machine learning method has remarkably recognised a key gene, which has been previously linked to the process of cell proliferation. Furthermore, a high protein expression of cyclin A has been associated with prognosis outcomes in non-Hodgkin's lymphomas [19].

```
GENE3330X <= 0.38
|   GENE518X <= -0.23: GCL
|   GENE518X > -0.23: ACL
GENE3330X > 0.38: GCL

Tree Size = 5
Number of genes = 2 (from 4026)
Training Error = 2.23%
45-Fold CV Error = 17.78%
```

(a)

```
GENE3330X <= 0.38
|   GENE3939X <= -0.82: GCL
|   GENE3939X > -0.82: ACL
GENE3330X > 0.38: GCL

Tree Size = 5
Number of genes = 2 (from 25)
Training Error = 2.43%
45-Fold CV Error = 17.78%
```

(b)

```
GENE3330X <= 0.38
|   GENE3512X <= -0.61: GCL
|   GENE3512X > -0.61: ACL
GENE3330X > 0.38: GCL

Tree Size = 5
Number of genes = 2 (from 2)
Training Error = 2.23%
45-Fold CV Error = 8.89%
```

(c)

Fig. 4. Decision trees for DLBCL sub-classes. All of the decision trees were generated using 45 patients. In (a) 4026 genes were used (the whole set); in (b) 25 genes (very high relevant); and in (c) only 2 genes (1 was randomly chosen from the data).

3.2 Prediction of DLBCL Subtypes

Figure 4 illustrates three decision trees generated to differentiate among the DLBCL subtypes, ACL and GCL. Two genes were sufficient to build the trees. The gene GENE3330X has been included in all of the trees, which indicates its relevance to achieve this classification.

However, this gene can be combined with many others without considerably increasing the error rate. In fact, we randomly selected an a priori non–relevant gene, the gene GENE3512X, and the error rate was even lower than earlier, about 8.8% (leave–one–out). Therefore, one may state that the difference among the more important genes in terms of their relevance is very slight.

Based on medical research about the significance of specific genes to differentiate subtypes of DLBCL, 5 genes have been selected. They are those encoding CD10 (GENE3317X, GENE3318X and GENE3319X), BCL–6 (GENE3340X and GENE3341X), TTG–2 (GENE1719X and GENE1720X), IRF–4 (GENE1212X and GENE1213X) and BCL–2 (GENE2514X, GENE2536X, GENE2537X, GENE2538X), and some genes belonging to the BCL–2 family (GENE385X, GENE386X, GENE387X, GENE3619X and GENE3620X). The importance of these genes has been demonstrated by Azuaje by means of the simplified fuzzy ARTMAP model, which is a neural network–based model [3].

Results provided by the decision tree used only four genes (GENE1719X, GEN3318X, GENE3340X and GENE385X) and the error rate was 0% (training) and 26.67% (leave–one–out). The overfitting was very high, and therefore, these genes are not appropriate to predict accurately the subtype of DLBCL by using a decision tree.

4 Conclusions

A broad study of the database generated by Alizadeh et al. [2] was presented in this paper. It focused on both the feature selection and classification tasks.

From a biomedical point of view, the relevance of specific genes reported by Alizadeh et al. is not observed in our results. This is perhaps because other genes may also play an important role in processes associated with this disease. However, this conclusion may not be strongly supported by results, as these have been obtained from a small amount of patients, in comparison to the number of genes.

These analyses indicate that the data are insufficient to state indisputable conclusions. Many subsets of genes can achieve a good prediction performance, although most of them would provide an overfitted decision tree. From a classification point of view, some genes are indeed very important, but more data should be included to support these observations. Alizadeh et al. inferred that a subset of genes could accurately differentiate among two subtypes of DLBCL. Nevertheless, none of such subsets have been identified by our KDD framework. The results also suggest that several subsets can attain the same classification aims. In fact, many decision trees can be built by using non–identified–as–relevant genes, producing similar error rates for the classification task. Furthermore, this research indicates that a deep study on the non–linear inter–relationship among genes might reveal interesting properties, as it has been discussed in [3]. With regard to the analysis of non–linear inter–relationships among genes for distinguishing lymphoma subtypes, we have recently built a neural network classifier based on 25 genes selected by the Relief method (with 3 nearest neighbours). This model, which consisted of two hidden layers and was tested with 45–fold cross–validation, produced an error rate equal to 0%.

This study highlights the importance of data mining techniques to extract interesting patterns from biological data, the significance of the results in contrast to statistics, and their future projection.

Acknowledgements

The research was supported by the Spanish Research Agency CICYT under grant TIC2001–1143–C03–02.

References

1. A. A. Alizadeh, M. Eisen, D. Botstain, P. O. Brown, and L. M. Staudt, "Probing lymphocyte biology by genomic-scale gene expression analysis," *Journal of clinical immunology*, no. 18, pp. 373–379, 1998.
2. A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, truc Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. H. Jr., L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, W. Dennis D, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt, "Distinct types of diffuse large b–cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503–511, 2000.
3. F. Azuaje, "A computational neural approach to support discovery of gene function and classes of cancer," *IEEE Transactions on biomedical engineering*, vol. 48, no. 3, pp. 332–339, 2001.
4. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees.* Belmont, CA: Wadsworth International Group, 1984.
5. A. D. Gordon, *Classification.* Chapman & Hall/CRC, 1999.

6. K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighborhood," *Pattern Recognition*, vol. 10, pp. 105–112, 1977.
7. M. A. Hall, "Correlation–based feature selection for machine learning," Ph.D., Department of Computer Science, University of Waikato, New Zealand, 1998.
8. J. Han and M. Kamber, *Data Mining – Concepts and Techniques.* Morgan Kaufmann, 2001.
9. N. L. Harris, E. S. Jaffe, J. Diebold, G. Flandrin, H. K. Muller-Hermelink, J. Vardiman, T. A. Lister, and C. D. Bloomfield, "World health organization classification of neoplastic diseases of the hematopoietic and lymphoid tissues: Report of the clinical advisory committee meeting–airlie house, virginia, november 1997," *Journal of clinical oncology*, vol. 17, pp. 3835–3849, 1999.
10. C. W. Hooper, R. C. Holman, M. J. Clarke, and T. L. Chorba, "Trends in non–hodgkin's lymphoma (NHL) and HIV–associated NHL deaths in the united states," *American Journal of Hematology*, vol. 66, pp. 159–166, 2001.
11. K. Kira and L. Rendell, "A practical approach to feature selection," in *Proceedings of the Ninth International Conference on Machine Learning*, 1992, pp. 249–256.
12. I. Kononenko, "Estimating attributes: analysis and extensions of relief," in *Proceedings of European Conference on Machine Learning.* Springer-Verlag, 1994.
13. F. Levi, F. Lucchini, E. Negri, and C. L. Vecchia, "Trends in mortaligy from non–hodgkin's lymphomas," *Leukemia Research*, vol. 26, pp. 903–908, 2002.
14. H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, 1995.
15. R. L. Rivest, "Learning decision lists," *Machine Learning*, vol. 1, no. 2, pp. 229–246, 1987.
16. M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutor, R. C. T. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, T. S. Ray, M. A. Koval, K. W. Last, A. Norton, T. A. Lister, J. Mesirov, D. S. Neuberg, E. S. Lander, J. C. Aster, and T. R. Golub, "Diffuse large b–cell lymphoma outcome prediction by gene–expression profiling and supervised machine learning," *Nature Medicine*, vol. 8, no. 1, pp. 68–74, 2002.
17. The Non-Hodgkin's Lymphoma Classification Project, "A clinical evaluation of the international of the international lymphoma study group. classification of non-hodgkin's lymphoma," *Blood*, vol. 89, no. 11, pp. 3909–3918, 1997.
18. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 2000.
19. D. Wolowiec, F. Bergera, P. Ffrench, P. Byron, and M. Ffrench, "CDK1 and cyclin A expression is linked to cell proliferation and associated with prognosis in non–hodgkin's lymphomas," *Leuk Lymphoma*, vol. 1–2, pp. 147–157, 1999.

Deriving Multiple Topics to Label Small Document Regions

Henner Graubitz* and Myra Spiliopoulou

Otto-von-Guericke Universität Magdeburg, Germany
{graubitz,myra}@iti.cs.uni-magdeburg.de

Abstract. Information retrieval can be greatly enhanced if the semantics of document contents are made explicit as labels that can be queried by markup-sensitive languages. We focus on labelling small text fragments, such as parts of sentences or paragraphs, with frequent topics. We propose `WORDtrain`, a sequence miner that builds topics for small document regions, such as sentences with many subsentences. `WORDtrain` splits regions in such a way that non-overlapping fragments are built *and* the topics derived for them are frequent.
`WORDtrain` discovers frequent topics rather than choosing from a predefined reference list. This raises the issue of evaluating the quality of its resuls. To this purpose, we have designed two evaluation schemes, one requiring expert involvement and an automatic one. Our first experiments with these schemes show that `WORDtrain` yields promising results.

1 Introduction

Information seekers that pose queries against large collections are often confronted with ranked lists of complete documents, in which they must browse to find those (potentially few) text fragments of relevance. A solution to this problem lays in identifying the topics of text fragments and making them explicit, e.g. as surrounding labels in XML, upon which a question-answering system or any markup-sensitive query language like Xquery or Xpath can be applied.

Document annotation with labels from an existing ontology is an often pursued target [2, 3, 6–8]. However, acquiring an appropriate set of concepts for semantic labelling is not trivial. Hence, many researchers focus on the automated establishment of ontologies through the discovery of concepts and relationships among them from the documents themselves [1, 9, 10, 12]). Hotho et al show that a conceptual graph structure, such as an ontology, is useful in finding documents or regions associated with a specific concept [7].

However, deriving topics that describe arbitrary document fragments is inverse to finding the fragments associated with a given ontology by browsing the ontology. Topic derivement poses the challenge of simultaneously (a) partitioning the document into fragments and (b) discovering *frequent* topics for them. This problem is apparent for even small document regions like sentences: They may be so heterogeneous that no single frequent topic can describe them properly.

* Work funded by the German Research Society, grant no. SP 572/4-3 DIAsDEM.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 289–298, 2004.

This problem is relevant to cluster labelling, as addressed e.g. in [11]. However, Rauber and Merkl derive labels for whole documents rather than regions [11]. In our previous work [4], we have designed an algorithm that derives topics for sentences by clustering them on term similarity, evaluating cluster quality and deriving labels for qualitatively acceptable clusters. However, the issue of topics inside the sentences is not addressed. In a different thread, Handschuh et al discover entities with help of automatically derived metadata [5, 6], using information extraction techniques to this purpose. Since they focus on recognition and subsequent tagging of entities, they are interested in finding the components of those entities rather than deriving topics that describe the surrounding parts of the documents, nor in specifying the borders of these parts.

To solve the problem of finding frequent topics that describe arbitrary portions of document regions and can be used as labels for them, we propose `WORDtrain`, a sequence miner that builds topics as frequent sequences of words inside regions or region partitions. `WORDtrain` is singular in satisfying two interdependent constraints: First, the derived topics must describe non-overlapping fragments of each document region, because they are intended as XML tags. Second, the sizes of the partitions and the spliting points must be chosen in such a way that the derived topics are frequent enough for querying. Currently, `WORDtrain` focusses on sentences as regions.

When meta-information is extracted automatically without a reference to compare with, one faces the question of proper evaluation of the results. We have designed two evaluation schemes, one with expert involvement and one without, in which we build a framework for evaluation in the absence of a priori background knowledge. Using these schemes we performed a preliminary evaluation of `WORDtrain`, acquiring promising results.

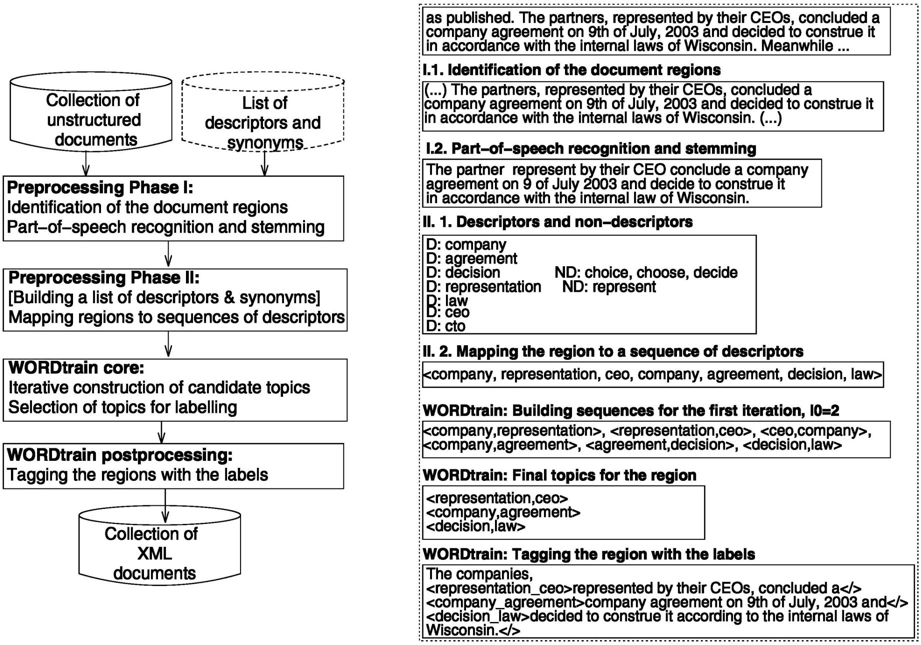
The paper is organised as follows: In section 2, we present the overall process of `WORDtrain` and then the individual modules and their tasks. In section 3, we describe the evaluation methodology, used in the experiments of section 4. The last section concludes with a summary and important subjects for future work.

2 Building Semantic Labels with the `WORDtrain` Algorithm

Goal of `WORDtrain` is the discovery of topics to label small document regions. Each topic/label (i) must reflect the content of the region or a *partition* of it and (ii) must be frequent. Currently, we define a "region" as a sentence.

We observe the document archive D as the collection of all small regions of the documents. We map each region into a sequence of descriptors from a given list. In the absence of a pre-compiled list, we use a utility described below to create a simple one. We then find frequent sequences within each region, starting with single frequent descriptors and expanding iteratively (obviously remaining inside the region borders). These frequent sequences become topics/labels in the regions, thereby preventing overlapping labels.

We depict the overall process at the left of Fig. 1. For the detailed description, we use the example long sentence at the right of the same figure.


Collection of unstructured documents
List of descriptors and synonyms
Preprocessing Phase I:
Identification of the document regions
Part–of–speech recognition and stemming
Preprocessing Phase II:
[Building a list of descriptors & synonyms]
Mapping regions to sequences of descriptors
WORDtrain core:
Iterative construction of candidate topics
Selection of topics for labelling
WORDtrain postprocessing:
Tagging the regions with the labels
Collection of XML documents
I.1. Identification of the document regions
I.2. Part–of–speech recognition and stemming
II. 1. Descriptors and non–descriptors
II. 2. Mapping the region to a sequence of descriptors
WORDtrain: Building sequences for the first iteration, l0=2
WORDtrain: Final topics for the region
WORDtrain: Tagging the region with the labels

as published. The partners, represented by their CEOs, concluded a company agreement on 9th of July, 2003 and decided to construe it in accordance with the internal laws of Wisconsin. Meanwhile ...

(...) The partners, represented by their CEOs, concluded a company agreement on 9th of July, 2003 and decided to construe it in accordance with the internal laws of Wisconsin. (...)

The partner represent by their CEO conclude a company agreement on 9 of July 2003 and decide to construe it in accordance with the internal law of Wisconsin.

D: company
D: agreement
D: decision ND: choice, choose, decide
D: representation ND: represent
D: law
D: ceo
D: cto

<company, representation, ceo, company, agreement, decision, law>

<company,representation>, <representation,ceo>, <ceo,company>, <company,agreement>, <agreement,decision>, <decision,law>

<representation,ceo>
<company,agreement>
<decision,law>

The companies, <representation_ceo>represented by their CEOs, concluded a</> <company_agreement>company agreement on 9th of July, 2003 and</> <decision_law>decided to construe it according to the internal laws of Wisconsin.</>

Fig. 1. Deriving region-level topics for semantic labelling with `WORDtrain`

2.1 Linguistic Preprocessing

The first processing phase encompasses region (currently: sentence) splitting, POS-tagging and stemming, for which standard NLP tools can be used, like the GATE toolsuite (www.gate.ac.uk) or the multilingual TreeTagger (www.ims.uni-stuttgart.de/projekte/corplex/treetagger/). Named-entity extraction may also be performed but is optional since it is orthogonal to topic discovery.

At the end of this phase, the archive is mapped into a collection of regions/sentences D. Adjointness of regions is no more taken into account. Since a region may appear more than once in D, we assign invisible identifiers to the regions, so that we can observe D as a set. At the right side of Fig. 1, we show these preprocessing steps for an example text excerpt.

2.2 Mapping Document Regions to Sequences of Descriptors

In the second phase, we need a thesaurus or at least a list of descriptors as feature space. At the right side of Fig. 1, a tiny list of descriptors and non-descriptors is used, according to the terminology of the ISO 2788-1986 standard.

Building a Vector Space for the Document Regions. In many applications, there is no precompiled list of descriptors. Occasionally, ontologies or thesauri exist but do not fit well to the collection content. A *collection-based* list of descriptors is more appropriate in such cases.

For WORDtrain, we derive such a list using word statistics and a general purpose lexicon: We first compute the term frequency (TF) and the inverse document frequency (IDF) of all words upon D. We then remove very frequent and very rare words and those with low TF/IDF, retaining the rest into a list $\mathcal{W}$. We then use a general purpose lexicon $\mathcal{L}$ that contains synonyms: Scanning $\mathcal{W}$ from the word with the highest TF/IDF value downwards, we attach to each word w those synonyms from $\mathcal{L}$ that also appear in $\mathcal{W}$ but are less frequent than w in D. Further synonyms from $\mathcal{L}$ are ignored as infrequent. The synonyms from $\mathcal{W}$ attached to w are removed from $\mathcal{W}$. This module outputs a list $\mathcal{T}$ of frequent descriptors and their less frequent synonyms as associated non-descriptors.

Document Regions as Sequences. We use the list $\mathcal{T}$ to map each region to a *sequence* of descriptors, whereby a non-descriptor appearing in a region is mapped to its corresponding descriptor. We build sequences rather than vectors, because we need the order of appearance of terms to derive multiple non-overlapping topics inside small regions. In such a sequence, a descriptor may appear more than once, but *adjacent* appearances are collapsed to a single one.

Hereafter, we use the term "original region" for the original document regions from D, the term "region" for the sequences of descriptors to which the former are mapped and the term "sequence" for an arbitrary ordered list of descriptors. We denote the set of regions as D'. D' is a subset of the set of all sequences $\mathcal{T}^*$.

2.3 Deriving Frequent Topics for Regions

WORDtrain derives topics as frequent sequences of consecutive descriptors upon the document regions. We first formalise the notion of frequency.

Frequency of Sequences. Let $s \in \mathcal{T}^*$ be a sequence of descriptors, let n be its length and let $r \in D'$ be a region of length n_r. The set $C(s,r)$ consists of all starting positions of s in r.

$$C(s,r) = \{i \in \{1,\dots,n_r\} | \forall j = 1,\dots,n : s[j] = u[i+j-1]\} \tag{1}$$

For the example in Fig. 1, let two sequences $s_1 =< \mathit{company}, \mathit{agreement} >$ and $s_2 =< \mathit{company}, \mathit{ceo} >$. Then, $C(s_1,r) = \{4\}$ and $C(s_2,r) = \emptyset$.

For $s \in \mathcal{T}^*$, the non-normalised number of appearances of s in a set of regions $X \subseteq D'$ is:

$$\mathit{appears}(s,X) = \sum_{r \in X} \mathit{card}\,(C(s,r)) \tag{2}$$

where $\mathit{card}(\cdot)$ denotes cardinality. For the example above, $\mathit{appears}(s_1,\{r\}) = 1$ and $\mathit{appears}(s_2,r) = 0$.

For a sequence $s \in \mathcal{T}^*$ of length n and a set of regions $X \subseteq D'$, the frequency of s in X is defined as:

$$freq(s, X) = \frac{appears(s, X)}{\sum_{y \in T^n} appears(y, X)} \tag{3}$$

with the simplification $freq(s) \equiv freq(s, D')$.

Input Parameters for `WORDtrain`. Before describing `WORDtrain` in detail, we must introduce the parameters governing its behaviour. To this purpose, we temporarily observe `WORDtrain` as a conventional sequence miner: It starts by finding all frequent sequences of length 1 (i.e. single descriptors), subject to a threshold $\tau_{appears}$ (cf. Eq. 2) and then expands them iteratively, until the expansions are no more frequent. Although the expansion phase of `WORDtrain` has no equivalent among conventional sequence miners, this oversimplification is adequate for the discussion of its input parameters.

First, we note that frequent sequences upon partitions of document regions reflect the topics of these partitions. Occasionally, a topic consisting of a single descriptor might be too simplistic. Hence, the user may specify that a topic should consist of at least $l_0 \geq 1$ descriptors, with default $l_0 = 1$.

After the first mining iteration, the frequent sequences of length l_0 will be expanded. Each expansion is at most as frequent as the original sequence. There is a tradeoff between shorter and more frequent topics/labels versus longer ones, which are intuitively more informative. `WORDtrain` uses a cutoff parameter, τ_{expand}, which determines the percentage of sequences to be expanded in each iteration. If τ_{expand} is set to 1, `WORDtrain` will build sequences of maximal length, as done in conventional sequence mining. Since sequence frequency is bounded by $\tau_{appears}$, values of τ_{expand} close to 1 will result in a few long labels, while values close to 0 will result in many short labels.

Building Candidate Topics Iteratively. In the first iteration, `WORDtrain` traverses each region $r \in D'$ and generates all n-grams in it for $n = l_0$, i.e. for the minimum label length. An n-gram s is retained only if $appears(s, D') \geq \tau_{appears}$. The output of this step is a set of frequent l_0-long sequences $\mathcal{R}_{l_0}$.

In each further iteration $i = l_0 + 1, \ldots$, `WORDtrain` expands each sequence by one descriptor from $\mathcal{T}$, building a set of expansions E_i, from which sequences with $appears(s, D') < \tau_{appears}$ are immediately removed. Then, we juxtapose E_i and $\mathcal{R}_{i-1}$ to built the set of retained i-long sequences $\mathcal{R}_i$:

1. A sequence $s \in E_i$ contributes at most $contributions(s) = appears(s, D')$ topics. This is the maximum contribution *potential* only, because some appearances of s in regions may overlap.
   We retain the m sequences with the largest contribution potential, where $m = card(E_i) \times \tau_{expand}$, using the threshold on the percentage of sequences to expand τ_{expand}. We remove the remaining sequences from E_i.

2. We compute the *contribution factor* of E_i vs $\mathcal{R}_{i-1}$ as the ratio:
$$f_i = \frac{\sum_{x \in E_i} contributions(x)}{\sum_{s \in \mathcal{R}_{i-1}} contributions(s)} \tag{4}$$
reflecting the maximum portion of D' that could be labelled if the sequences in E_i were used for labelling and those in $\mathcal{R}_{i-1}$ were not used at all. Since a sequence is at least as frequent as all its expansions together, $f_i \in [0,1]$.
3. We traverse E_i from the most frequent sequence downwards. For each $x \in E_i$, let s be the subsequence consisting of the first $i-1$ descriptors in x; obviously, $s \in \mathcal{R}_{i-1}$ and x is its expansion.
   (a) We compute the relative contribution of x as $w(x) := contributions(x) - f_i \times contributions(s)$.
   If $w(x) \leq 0$, we retain s in $\mathcal{R}_{i-1}$ and we remove from E_i *all* expansions of s, not just x.
   Otherwise, we append x to $\mathcal{R}_i$, we remove x from E_i and we adjust the value of $contributions(s)$ by subtracting $contributions(x)$.
   (b) If $contributions(s) < \tau_{appears}$, we remove s from $\mathcal{R}_{i-1}$.

In the internal loop of step 3, `WORDtrain` considers each retained sequence s against its expansions: An expansion x is retained if it appears an adequate number of times in regions. Then, x will be used as label in these regions *instead* of s, so that the contributions of s should be reduced accordingly. Thus, if many expansions of s are retained, $contributions(s)$ may drop below $\tau_{appears}$.

This procedure stops at iteration k, for which $\mathcal{R}_{k+1}$ is empty. Then, `WORDtrain` outputs all topics/labels as the union of all sets of retained sequences $\cup_{i=l_0}^{k} \mathcal{R}_i$.

2.4 Topic Selection for Overlap-Free Labelling

The frequent sequences in $\cup_{i=l_0}^{k} \mathcal{R}_i$ may overlap, while labels may not. To select an appropriate subset of sequences for labelling, we first order the sequences in each $\mathcal{R}_i$ on their $contributions(\cdot)$ values. Then, starting with $\mathcal{R}_k$ and moving towards l_0, we consider for each $\mathcal{R}_i$ the sequences in descending order of their contributions. For each sequence s, let $s[j]$ denote the descriptor at position j, where $j = 1, \ldots, length(s)$.

The document regions are labelled with the longest sequences first: For each sequence s, we scan each region in D for pieces of text that match s, i.e. consisting of adjacent terms that are either the descriptors in s or their associated non-descriptors in the same order of appearance. A matching piece of text may not contain any additional terms, but can contain stopwards, instances of named entities etc. Matching pieces of text that already contain a previously inserted opening or closing tag are ignored, because tags may not overlap. The remaining matches are still unlabelled, non-overlapping region partitions; s becomes their label. The tag positioning for labels is explained below.

At this point, we count these matching partitions into $labels(s, D)$. This number may be less than $contributions(s)$, because labels may not overlap and because some other sequences may have prevented the use of s as label. If it is even less than $\tau_{appears}$, then s is too rare to be used a label. Then, `WORDtrain` removes the labels from the regions again. Since `WORDtrain` scans $\cup_{i=l_0}^{k}\mathcal{R}_i$ towards $\mathcal{R}_{l_0}$, the most frequent (but shortest) sequences are considered last. Hence, the affected regions may be labelled by another frequent sequence.

To position the opening and closing tag for s, we use two rules: (i) The first tag appears just before the first term of the partition, unless this is the first term of the region, in which case the opening tag appears at the beginning of the region. (ii) The closing tag is placed just before the first term not belonging to the partition, if any, i.e. after all text following the last term associated with the topic. At the right side of Fig. 1, the lowermost box shows the two topics built for the sentence and how their surrounding tags are positioned.

3 Evaluation Schemes for Derived Labels

The evaluation of the `WORDtrain` results against a manually annotated set of sentences is not appropriate: The expert may assign a label to a region that consists of rare terms or terms that not appear in the region. Moreover, `WORDtrain` may build topics that the expert was not aware of. Hence, we propose two evaluation schemes. The first one derives quality indicators, while the second one specifies how a "controlled set" of labels and region partitions can be built by the expert.

3.1 Evaluation Scheme Without Expert Involvement

In this scheme, `WORDtrain` is evaluated across the following guidelines:

1. The larger part of the archive is assigned labels, the better.
2. Long topics are more informative and thus preferable to short ones.
3. Each topic should appear in a minimum number of regions.
4. The more labels a region has, the better.

Using these guidelines, we derive the following quality indicators: (i) *Coverage* as portion of the labelled regions within the archive, (ii) total number of labels, (iii) length of these labels and (iv) number of labelled partitions per region.

3.2 Evaluation Scheme with Expert Involvement

In this evaluation scheme, the expert identifies region partitions and labels them. To prevent labels from terms unknown to `WORDtrain`, less frequent than the expert-specified $\tau_{appears}$ or shorter than the minimum length l_0, we require that:

1. The labels assigned by the expert must be among those proposed by `WORDtrain`.
2. A label assigned by an expert inside a region can only be a sequence of adjacent terms in the region; this determines the partition to be tagged, spanning from the label's first term to the last (cf. 2.4).
3. Labels assigned by the expert may not be nested nor overlapping.

Despite being restrictive, the above requirements still allow for multiple alternative labels in each region: The expert may decide to split the region in different ways, thus assigning different labels to the resulting partitions. We further distinguish between the case that the expert takes the same decision as `WORDtrain` vs approving the decision of `WORDtrain`. In particular, the expert may *approve* the partitions and labels proposed by `WORDtrain` as correct, although she may herself decide to split and label the region differently. We thus come to the following definition of precision and recall:

Definition 1 *For a region $v \in D$, we denote as $WT(v)$ the set of labels proposed by* `WORDtrain` *for v, $Expert(v)$ the set of labels assigned by the expert and $Approved(v) \subseteq WT(v)$ the set of* `WORDtrain` *labels that the expert has approved for v. We define "precision" as the ratio of approved to proposed labels and "recall" as the ratio of common labels to those assigned by the expert.*

$$precision = \sum_{v \in D} \frac{card(Approved(v))}{card(WT(v))} \qquad recall = \sum_{v \in D} \frac{WT(v) \cap Expert(v))}{card(Expert(v))}$$

4 Testing `WORDtrain`

The Document Collection. We have experimented with `WORDtrain` on one of the archives of the German Commercial Register (GCR). Companies with seat in Germany are obliged to report in the GCR all activities of relevance to contractual agreements, such as company establishments and seat relocations. A GCR entry has one structured part and one unstructured document of typically one (often large) paragraph. In our experiments, we process only the latter.

We used a subcollection of 1145 documents from the GCR of the city of Potsdam, the 1999 company foundations. These documents have many sentences with more than one topics, e.g. target areas of a company or functions of an appointed manager. For sentence splitting we used a previously developed sentence splitter that takes abbreviations, dates and other expressions with delimiters into account [4]; it produced 10705 sentences. For POS-tagging and stemming, we applied the TreeTagger[1]. We then used an existing list of 127 descriptors and 111 non-descriptors for the feature space [4].

Running `WORDtrain`. For the experiments, we have used a slight variation of our algorithm, in which $\tau_{appears}$ was used only in the first iteration of label construction (cf. 2.3). Despite this simplification, the topic selection (cf. 2.4) uses $\tau_{appears}$ and thus guarantees that no infrequent topics become labels.

We run a Java 1.4.2 implementation of `WORDtrain` on an AMD Athlon 2400 MHz with 512 MB RAM under Linux. The execution time was between 9 and 15 seconds. It was mostly influenced by the minimum topic length l_0, reaching the maximum of 15 seconds for $l_0 = 1$, i.e. when the largest number of candidates was considered. The number of descriptors in a topic varied between 1 and 23 with an average length of 3.36.

[1] http://www.ims.uni-stuttgart.de/projekte/corplex/treetagger/

Evaluation Without Expert Involvement. We set $\tau_{appears} = 50$ and varied the minimum length of candidate topics l_0 and the percentage cutoff for sequence expansions per iteration τ_{expand}. First, setting τ_{expand} to 0.1, we varied the value of l_0. Table 1 shows the effects upon the coverage in labelled sentences, the number of labelled (sub)sentences and the total number of output labels.

Table 1. The impact of l_0

l_0	Coverage		Number of labels of length				
	Num	Percent	1	2	3	4	≥5
1	10395	97.10%	10	7	2	2	2
2	**7585**	**70.85%**	–	**20**	**8**	**0**	**0**
3	4440	41.47%	–	–	12	4	1
4	3875	36.19%	–	–	–	11	5

The coverage in labelled sentences drops when one-descriptor topics are not considered. The number of labels does not drop, though; it rather reaches its maximum for $l_0 = 2$. These 28 labels have still a large coverage of 70%. This is important because multi-descriptor topics are more informative. Hence, we set $l_0 = 2$ for the rest of the experiments.

For a minimum label length $l_0 = 2$, we varied the τ_{expand} between 0.05 and 0.3 with a step of 0.05. The impact of τ_{expand} upon the number of sentences with none, one or more labels (i.e. the number of labelled partitions) and upon the absolute coverage in labelled sentences are shown in Table 2.

Table 2. The impact of τ_{expand}

Labels in sentence	Coverage $\tau_{expand} = 0.05$	Coverage $\tau_{expand} \geq 0.1$
1	3435	3531
2	2185	2273
3	1659	1635
4	72	88
5	10	52
≥6	2	6
none	3342	3120

We see that any τ_{expand} value in [0.1,0.3] results in *both* a larger coverage *and* in more labelled partitions. However, varying τ_{expand} between 0.1 and 0.3 has no effect. One explanation is that only the 10% most frequent sequences have expansions with adequately high contribution factor (cf. Eq. 4).

Evaluation with Expert Involvement. Finally, we used the evaluation scheme of subsection 3.2 on a random sample of 1% of the collection's sentences. We have provided the expert with the 28 labels proposed by `WORDtrain` for $l_0 = 2$, $\tau_{appears} = 50$ and $\tau_{expand} = 0.3$ (cf. Table 1). 30 sentences could not be labelled, because there was no frequent pair of descriptors in them. For the remaining 74 sentences, the precision was 0.96 and the recall was 0.93 (cf. Def. 1).

5 Conclusion

We have proposed `WORDtrain`, a miner for topic discovery and subsequent labelling of document regions and region partitions. The major challenge mastered by `WORDtrain` is the splitting of regions into non-overlapping partitions for which frequent topics are built as sequences of adjacent terms. We have further proposed two schemes for the evaluation of such an algorithm which discovers new topics and cannot be juxtaposed to a gold standard. Our experiments using these schemes show that WORDtrain is good in region partitioning and frequent topic discovery for the partitions.

We plan to extend WORDtrain for the derivement of nested topics on regions of varying granularities and also improve the tag positioning mechanism. We further intend to generalise our evaluation schemes for further methods on unsupervised topic discovery and ontology learning.

References

1. Philipp Cimiano, Steffen Staab, and Julien Tane. Automatic acquisition of taxonomies from text: Fca meets nlp. In *Proc. of the ECML/PKDD Workshop on Adaptive Text Extraction and Mining*, pages 10–17, Cavtat, Croatia, Sept. 2003.
2. Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proc. of the 12th Int. World Wide Web Conf.*, pages 178–186, Budapest, Hungary, 2003. ACM Press.
3. M. Erdmann, A. Maedche, H.-P. Schnurr, and S. Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. *ETAI Journal - Section on Semantic Web*, 6, 2001.
4. Henner Graubitz, Myra Spiliopoulou, and Karsten Winkler. The DIAsDEM framework for converting domain-specific texts into XML documents with data mining techniques. In *Proc. of the 1st IEEE Intl. Conf. on Data Mining,*, pages 171–178, San Jose, CA, Nov. 2001. IEEE.
5. Siegfried Handschuh, Steffen Staab, and F. Ciravegna. S-cream – semi-automatic CREation of metadata. In *Proc. of the European Conf. on Knowledge Acquisition and Management*, 2002.
6. Siegfried Handschuh, Steffen Staab, and Raphael Volz. On deep annotation. In *Proc. of the 12th Int. World Wide Web Conf.*, pages 431–438, Budapest, Hungary, 2003. ACM Press.
7. Andreas Hotho, Steffen Staab, and Gerd Stumme. Explaining text clustering results using semantic structures. In *Proc. of ECML/PKDD 2003*, LNAI 2838, pages 217–228, Cavtat-Dubrovnik, Croatia, Sept. 2003. Springer Verlag.
8. Jianming Li, Zhang Lei, and Yong Yu. Learning to generate semantic annotation for domain specific sentences. In *Proc. of the "Knowledge Markup and Semantic Annotation" Workshop of the K-CAP 2001 Conference*, 2001.
9. Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In *Proc. of ECAI'2000*, pages 321–325, 2000.
10. Alexander Maedche and Steffen Staab. Mining ontologies from text. In *Proc. of Knowledge Engineering and Knowledge Management (EKAW 2000)*, LNAI 1937. Springer, 2000.
11. Andreas Rauber and Dieter Merkl. Mining text archives: Creating readable maps to structure and describe document collections. In *Int. Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pages 524–529, 1999.
12. Karsten Winkler and Myra Spiliopoulou. Structuring domain-specific text archives by deriving a probabilistic XML DTD. In *6th European Conf. on Principles and Practice of Knowledge Discovery in Databases, PKDD'02*, pages 461–474, Helsinki, Finland, Aug. 2002. Springer Verlag.

Deriving Efficient SQL Sequences via Read-Aheads*

A. Soydan Bilgin[1], Rada Y. Chirkova[1], Timo J. Salo[2], and Munindar P. Singh[1]

[1] Computer Science Dept, North Carolina State University, Raleigh NC 27695, USA
asbilgin@unity.ncsu.edu, {chirkova,mpsingh}@csc.ncsu.edu
[2] IBM, Research Triangle Park, Raleigh NC 27709, USA
tjsalo@us.ibm.com

Abstract. Modern information system architectures place applications in an application server and persistent objects in a relational database. In this setting, we consider the problem of improving application throughput; our proposed solution uses data prefetching (read-aheads) to minimize the total data-access time of an application, in a manner that affects neither the application code nor the backend DBMS. Our methodology is based on analyzing and automatically merging SQL queries to produce query sequences with low total response time, in ways that exploit the application's data-access patterns. The proposed approach is independent of the application domain and can be viewed as a component of container managed persistence that can be implemented in middleware. This paper describes our proposed framework for using generic data-access patterns to improve application throughput and reports preliminary experimental results on discovering key parameters that influence the trade-offs in producing efficient merged SQL queries. The approach is evaluated in the context of a financial domain, which yields the kinds of natural conceptual relationships where our approach is valuable.

1 Introduction

Three-tier application architectures place business logic on an application server and the necessary persistent objects on backend relational database-management systems (RDBMSs). Applications issue requests to the RDBMS for stored data; accessing persistent objects as they are requested may result in unacceptable physical disk access and network processing overheads. In current practice, programmers can spend inordinate amounts of time tuning their applications to reduce the overhead; an additional drawback of tuning is that it must be repeated each time the database schema or application logic are modified.

An alternative way to reduce the data-access overhead is to reduce the number of database roundtrips required to fulfill an application's request for stored objects. There are two types of possible approaches – caching and prefetching. *Caching* [8, 11, 12] refers to storing recently accessed objects, thereby avoiding

* This research is supported under NCSU Center for Advanced Computing and Communication Grant 11019.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 299–308, 2004.

unnecessary requests to the database. *Prefetching* [9, 2, 14, 10] stands for fetching data based on a prediction of an application's future requests. (Cooperative query answering [7] is a related methodology.) Both caching and prefetching can result in significant payoffs in data-access performance [13, 1, 2]. Simple prefetch mechanisms, which create read-ahead threads for certain queries that return large quantities of data sequentially from a single table, are already used by commercial data servers and by object managers or containers in application servers.

In this paper we propose an adaptive prefetching approach [3] to reducing the number of data-access roundtrips. The approach can be used to automatically iteratively improve data throughput without modifying the application code or the DBMS; as such, the approach can be used in middleware in managed object environments. The proposed framework comprises interactive query exploration and automatic query analysis based on application behavior, and can be used as a component of autonomic self-tuning data-access systems for data-intensive applications.

We develop our approach for settings where an application's likely next few queries are known beforehand (e.g., canned interfaces); application domains that will benefit from our approach include health care, financial domains, human resources, and all domains with decision and analysis support. For these settings, we propose a set of techniques for automatically analyzing an application's data-access patterns and for using the results of the analysis and predefined guidelines to prefetch the answers to the future queries using sequences of read-ahead queries (in SQL) with low total response time. Our results are not meant to be used by individual applications to rewrite queries in a static fashion; rather, the objective is to create application-independent techniques that a middleware system would use as the basis for providing an online, adaptive way of implementing read-aheads. Thus, our contributions are to autonomic computing, namely to the development of automated systems in managed object environments, with the ability to iteratively analyze data-access patterns and choose the appropriate guidelines for merging application queries, at application runtime and with no human intervention. The main feature of our approach is the discovery and use of guidelines for selecting, based on an application's access patterns and additional parameters, efficient ways of merging the application's data requests into read-ahead statements. The proposed framework can also incorporate additional (empirical) rules added by experts or developers.

In this paper we describe the proposed framework for using generic data-access patterns to improve application throughput (Section 3), and report preliminary experimental results on studying the key parameters that influence the trade-offs in producing efficient merged SQL queries (Section 4). The approach is evaluated in the context of a financial domain, which yields the kinds of natural conceptual relationships where our approach is valuable.

2 Problem Formulation and Assumptions

Many applications have structure: they access mutually related objects, whose relationships are expressed in the schema of the underlying database. Stored data are accessed according to these associations, and most of the time the associations are intuitive and work just as you would expect. Many important applications, including those in health care, financial domains, or human resources, use the same query templates repeatedly. For example, an application can request the due date of a credit-card payment after requesting the balance, or it can request the transactions of the same card in the last billing period. Such associations and dependencies that can be found in different domains form a basis for formulating useful application-independent data-access patterns. Moreover, in many applications, the likely next few queries are known beforehand.

In settings where persistent objects are stored in a RDBMS and where an application's likely next few queries are known beforehand, we consider the problem of reducing the number of data-access roundtrips in furnishing the application's data requests, by generating efficient sequences of read-ahead queries (in SQL). Our goal is to develop application-independent methods that could be automatically applied and tuned both for multiple unrelated applications and for a single application over time in a changing environment. Specifically, our goal is to come up with a set of general guidelines and techniques for automatically analyzing the data-access patterns of each given application and constructing a sequence of read-ahead queries with low total response time. We assume that for each application, we have available a graph of the application's data-access requests (for an example, see Figure 1), which expresses the dependencies between the likely queries in the application. We discuss this structure in more detail later in this section.

Problem Formulation. Given (1) an application's data-access request graph, (2) a database, (3) application-independent guidelines for obtaining efficient merged read-ahead queries, and (4) values of parameters[1], in the application and its environment. Produce: sequences of read-ahead queries that maximize the likelihood of furnishing the application's future data requests and have minimal total response time on the database. Formally, for a database D, for a set $\mathcal{Q} = \{q_1, \ldots, q_n\}$ of queries in an application, and for all sets $\mathcal{S}_i = \{s_{i1}, \ldots, s_{ik_i}\}$ of read-ahead queries that return at least the same information as $\mathcal{Q}$ on D, the set $\mathcal{S}_i$ of queries that has minimal response time on D is $argmin_i \Sigma_{j=1}^{k_i} cost(s_{ij}, D)$, where $cost(q, D)$ is the response time of a query q on a database D.

We consider read-ahead queries that result from merging two or more SQL queries as posed by the given application. The sequence of merged read-ahead queries produces at least the same outputs as the original queries, but because there are fewer of them, the results are obtained with fewer DBMS calls and network round-trips. Note that devising a single read-ahead query for all the application's queries is not guaranteed to produce a query with minimal response

[1] We discuss parameters in detail in Sections 3 and 4; examples of parameters include the number of joins in a query and the size of a query answer.

time. One reason is that in many realistic settings the resulting read-ahead query will be extremely complex and thus not amenable to significant optimization. Instead, we are looking for break-even points between the extremes of multiple individual SQL statements (low complexity but high number of queries) and a single complex read-ahead query (minimal number of queries but high complexity). In our current work, we seek to improve the application throughput.

Assumptions. We restrict our attention to applications making individual, rather than batch, queries: We focus on queries such as *find the name of the customer with given ID* and not *find the name of all customers.* We assume that applications generate unnested select-project-join SQL queries, possibly with aggregation. In addition, we assume that we are given a graph of the application's data-access requests (for an example, see Figure 1), which expresses the probabilistic transitions between the likely queries in the application; the graph can be used to predict which queries the application is likely to use next.

3 The Proposed Framework

Determining, in a general, automated, and cost-based manner, which read-ahead queries to use, involves (1) a module for generating merged queries by combining two or more SQL queries; (2) a module for answering the original query from the merged queries; and (3) a cost-based query optimizer.

In our approach, we use the query optimizer in the RDBMS that stores persistent objects, and we do not consider the problem of answering the original query from the combined queries (for possible solutions to this problem see, e.g., [8]).

We focus on the problem of determining beneficial and efficient read-ahead queries based on the knowledge of the likely future queries of a given application and on the general knowledge of how certain values of parameters, in queries and in the application's environment, affect the efficiency of read-ahead queries. There are two main components of the problem. The first is to design application-independent – *generic* – guidelines for determining efficient sequences of read-ahead queries. The second component is to develop scalable algorithms for fine-tuning the generic guidelines. Specifically, we are taking the following steps:

1. Use case studies to determine application-independent parameters, including generic data-access patterns, that can affect the efficiency of read-ahead queries.
2. Develop automatic methods for merging two or more SQL queries into a single query that provides the answers to the input queries.
3. Develop application-independent guidelines for generating read-ahead query sequences – specifically for obtaining read-ahead queries merging two or more queries in an application – based on the values of the parameters.
4. Given specific applications and databases, validate the parameters and guidelines, by checking whether they produce sequences of read-ahead queries that maximize the likelihood of furnishing the application's data requests and have minimal total response time on the database.

In Section 4, we report preliminary results on discovering some parameters that could influence the trade-offs in producing efficient merged SQL queries, in the context of a financial domain. Our next step is to study the degree of performance improvement given for frequently occurring query sequences, in setups with multiple clients and application servers and with a single database server.

We study application-independent data-access patterns and generate SQL queries that implement access patterns that are part of the given access sequence. To generate various types of read-ahead queries, we can use various prefetch methods such as (1) prefetch only primary keys of the associated objects, and (2) prefetch via traversing the inheritance-extension, one-to-many associations.

While generic data-access patterns help determine useful read-ahead sequences, they are not adequate for determining sequences with minimal total response time. Finding such sequences involves determining:

1. *How much to read ahead?* Answering this question requires figuring out which tables and columns may be subsequently accessed, and how the structure of the stored data (e.g., presence of indexes) and relational constraints affect a read-ahead query.
2. *How deep to read ahead?* Answering this question requires figuring out the levels of the hierarchy that may be subsequently accessed. One question is how the number of joins affects the efficiency of a sequence.
3. *In what direction to read ahead?* In each level of the hierarchy, each object can be associated with many objects. Answering this question requires figuring out the trade-off among the potentially many data traversal directions.

In our current experiments we consider sequences of queries with 100% probability that each future query will be issued by the application. We are also working on methods for determining read-ahead queries for more general settings, such as those in Figure 1. In these settings, for each sequence that may be considered, there is a probability that it will be useful, there are penalties that will be incurred from the execution of queries, and there is an expected benefit. We are working on a cost-benefit analysis of the utility of merged queries, which will take into account these factors to determine promising read-ahead sequences.

4 Research Methodology and Experiments

We construct a testbed by using a slightly modified version of a standard data model for financial services (e.g., banking and investment services) [15] to experiment with the parameters that affect the cost of read-ahead queries. A more complete set of information about the testbed and results can be found in [4]. The data model has 55 tables and maximum fan-out of 7 relationships. We chose finance domain, because it features complex queries, meaningful relationship traversals and diverse set of meaningful query sequences.

4.1 Methodology

We model frequent data accesses that can be issued by the applications in one directed graph, whose vertices correspond to simple SQL queries and edges correspond to transition probabilities between the queries. Figure 1 shows a sample graph with six SQL queries. The edge from A to B means that whenever A is accessed there is a 40% chance that B will be accessed next. We assume an acyclic graph that can have multiple sources and sinks.

There are two principal ways to combine two or more requests:

- *Merged execution*, when the requests are executed at the same time. The corresponding operator * is commutative and associative. We use *outer joins* to merge the requests.
- *sequential execution*, when the requests are executed one after the other. The corresponding operator + is associative. For convenience, + has lower syntactic precedence than *.

For example, an access sequence $ABEF$ can be supported by a variety of prefetch sequences, including $A*B*E+F$, $A*B*E*F$, $A*B*D+D*E*F$, $A*B+E*F$, and so on. Figure 2 represents the level-by-level formation of read-ahead sequences as a directed graph. Here, each vertex represents the set of SQL queries that can be executed at the same time or one after the other. This representation reduces the number of vertices that we must consider.

Let the cost of evaluating an SQL query Q be the response time of Q. The efficiency ratio of a read-ahead SQL query is the fraction of the minimal cost of the previous level read-ahead SQL sequences to the cost of the current read-ahead SQL query. The previous level read-ahead SQL sequences are computed by applying dynamic programming on the current node. For example, the efficiency ratio of the read-ahead query $A*B*E*F$ is the fraction of $\texttt{min}(cost(A+B*E*F, A*B+E*F, A*B*E+F))$ to $cost(A*B*E*F)$. In our experiments, our goal is to find the read-ahead SQL queries whose have efficiency ratio is close to 1.0 or greater than 1.0 by estimating the cost of the read-ahead SQL sequences. If the efficiency ratio of the previous level read-ahead sequences are all smaller than 1.0, then we should stop exploring down the graph to find more efficient read-ahead SQL queries that can be used for the given SQL sequence.

By testing the possible read-ahead scenarios for the SQL queries in Figure 1, we can come up with many meaningful linear sequences of queries (X and Y are variable names). A more detailed list of SQL statements for the queries can be found in [4].

A Find the IDs and balance of the accounts that customer X 'owns'

B Find the IDs and balance of the accounts, where another account Y uses as 'overdraft' account(s)

C Find the account IDs and balance of the customer, who is 'co-owner' of the account Y

D Find the IDs, feature name, and the value of the products that account Y is linked

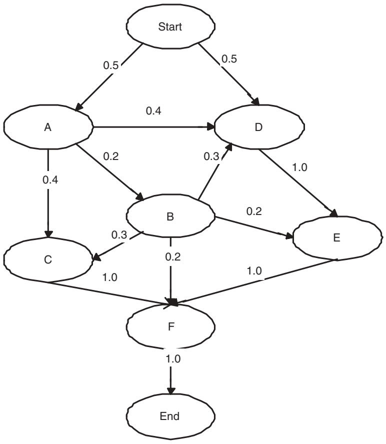


Fig. 1. A query transition graph

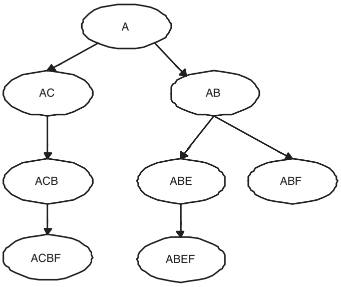


Fig. 2. The read-ahead scenarios graph for Fig. 1

E Find the IDs and amount of the transactions on account Y

F Find the agreement IDs, the parties and the roles of these parties that are linked to these agreements, where account Y is linked via an ownership or co-ownership relation

Queries A+B+E could arise in a situation where the customer representative first issues A to learn the account information for customer X, then he chooses one of these accounts, and inquires the 'overdraft' accounts that is related to this account via B. In the end, he issues E to learn the transactions of an account. The above list can be extended by using more complex query transition graphs. However, simple SQL queries are enough to derive complex and meaningful read-ahead scenarios, which help us in determining the rules of thumb regarding what strategies to use for efficient sequences of read-ahead SQL queries.

4.2 Experiments and Results

We generated our test scenarios for the first set of experiments according to the following incremental methodology:

- Merge two simple queries with n unshared join tables (n>=1)
  – Add one shared join table to the queries. For example, we have Account for A*B
  – Add one more unshared table to one of these tables which is the case for B*E so that we can understand the effect of the number of unshared tables in queries.
  – Add more than one shared join table as in A*C
- Merge more than two queries via using simple (e.g. A) and compound queries A*B

– Merge the compound query with simple queries with different number of shared join tables. For example, *A*D* can be merged with *E* to form *A*D*E* with no shared join tables, *A*B* can be merged with *F* to form *A*B*F* with one shared join table, and so on.

– Merge compound queries to form more complex compound statements with different number of shared join tables such as *ABCF* and *ADEF*. For example, *ABCF* can be formed by merging *AB* and CF that have two shared tables.

The tests were done on 2.0 GHz Intel Xeon CPU with 2GB memory and 6x36 SCSI 10K RMP hard drive under Windows NT server. We use Oracle 9i to host our database. Above methodology assumes that each added table results in an addition of a join restriction. We also varied the cardinality of the tables (10K, 100K, 1M), the cardinality of returned data-set for each simple query (1, 1%, 10% selection). Table 1 lists the small subset of the results. We list the following situations according to the result of our preliminary experiments [4]:

- The increase in the number of joined tables did not increase the efficiency ratio of the merged query. For example, *A* and *C* have three common join tables, and *A* and *B* have only one, but *A*B* has better efficiency ratio than *A*C* which is mainly due to the wider intermediate tuples which results from extra joins. On the other hand, with small table sizes we get slightly better efficiency for *A*B*C* than for *A*B* which is due to the extra savings from database latency.
- The efficiency of *A*B* is not affected by the cardinality of shared joined table (ACCOUNT), but decreases if we increase the cardinality of the tables that are not shared in *A* and *B*. Also the efficiency increases with the increase in the number of the returned tuples that was selected by the selection predicates or join restriction via keys. For this situation, for relations *R* and *S*, the selection predicates such as *R.key=X* and *S.key=Y* have more negative impact than *R.key=S.key*. Also the bigger merged query may not result in better efficiency. For example, *A*B*F* has slightly better efficiency than *A*B*C*F* for small query answer size, because *A*B*C*F* requires require complex query optimaztionby the db engine.
- For *A*B*F*, its efficiency is much worse than *A*B+F*, although all the tables have one common table. Here, the main problem is the size of the redundant data (extra *null* fields) that returns as a result of the join. This inefficiency is especially affected by the cardinality of the result data-set of *F*.

5 Related Work

Most relevant approach to our work is semantic prefetching which considers the efficient query execution strategies for nested queries via predicting the correlations among these nested queries and then merging them according to these correlations [5]. Their main parameters are the transition probabilities among queries in the same context. They also consider in detail how to answer client

Table 1. The efficiency ratio of different merged SQL queries with different variables including shared table's size(ACCOUNT) and the query answer size

		Table Size		
SQL Query	**Result Selection Cardinality**	**10K**	**100K**	**1M**
A*B	1	1.04	1.05	1.04
	10%	1.24	1.23	1.23
A*C	1	0.98	0.96	0.96
	10%	1.03	1.02	1.04
A*B*C	1	1.08	1.07	1.04
	10%	1.11	1.09	1.09
A*B*F	1	0.98	0.09	<0.01
	10%	0.03	<0.01	<0.01
A*B*C*F	1	0.96	0.14	0.03
	10%	0.11	0.02	<0.01

queries using prefetch results. Other approaches include (1) optimizing computing overlapping queries that generate Web pages (e.g., online shopping, where users narrow down the search space as they navigate through a sequence of pages) [9]; (2) predicate-based caching, where the cache content is used to answer future queries [12]; and (3) prefetch the objects whose object identifiers were returned as part of the query [10]. Curently, our approach takes into account the cost of the data-access queries and various parameters that affect this cost for the given data-access sequences. We also consider unnested queries.

The context of an object can be used as a predictor for future accesses in navigational applications, where future access to the data can be unpredictable. [2] describes an approach that uses context in which an object *O* is loaded; when an object's state is loaded, similar state for other objects in *O*'s context is prefetched. The prefetching methods include heuristics such as prefetching all the attributes of the requested objects. However, this work performs only one-level prefetching for referenced objects, and does not answer the question of how deep to read ahead. Further, this work seeks to minimize database latency for future queries and does not explore the cost of efficient sequence of data-access queries.

6 Discussion and Future Work

Previous work does not consider the prefetching problem in terms of both query optimization parameters and navigational access patterns. By providing a mechanism for merging simple requests to find an efficient sequence of data-access queries, we use a different aspect for multi-query optimization [6] where dependencies or common subexpressions between the queries in a sequence are exploited. Our work can also be integrated with *Container Managed Persistence* containers or *Java Data Object* drivers as a performance tuning technique.

We use applications' common behaviors and the cost of database interactions to generate read-ahead rules that can provide a significant performance gain for systems where many concurrent data-intensive read-only applications access huge databases. We plan to accommodate considerations such as network latency and complex index structures for our rules. It is also necessary to use another domain for verifying the correctness and increasing the diversity of our rules. We also plan to find the threshold of the transition probabilities to use in the read-ahead scenarios efficiency formula. In further research, we will explore the effect of object-to-relational mapping techniques on our generic rules and will develop and test learning algorithms to increase the efficiency of our generic rules for the data-access patterns of particular applications.

References

1. Adali, S., Candan, K., Papakonstantinou, Y., Subrahmanian., V.: Query caching and optimization in distributed mediator systems. In: ACM SIGMOD Conf. on management of data. (1996) 137–148
2. Bernstein, P.A., Pal, S., Shutt, D.: Context-based prefetch - an optimization for implementing objects on relations. VLDB Journal **9** (2000) 177–189
3. Bilgin, A.S.: Incremental read-aheads. In: Proc. ICDE/EDBT Ph.D. Workshop, Boston, MA (2004)
4. Bilgin, A.S., Chirkova, R.Y., Salo, T.J., Singh, M.P.: Deriving efficient sql sequences via read-aheads, full version (2004) http://www4.ncsu.edu/∼asbilgin/Papers/readAheads.pdf.
5. Bowman, I.T., Salem, K.: Optimization of query streams using semantic prefetching. In: ACM SIGMOD, Paris, France (2004)
6. Choenni, R., Kersten, M., Saad, A., Akker, J.: A framework for multi-query optimization. In: Proc. COMAD 8th Int. Conference on Management of Data. (1997) 165–182
7. Chu, W.W., Chen, Q.: A structured approach for cooperative query answering. IEEE Trans. Knowl. Data Eng. **6** (1994) 738–749
8. Dar, S., Franklin, M.J., Jonsson, B.T., Srivastava, D., Tan, M.: Semantic data caching and replacement. In: Proceedings of the 22th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (1996) 330–341
9. Florescu, D., Levy, A., Suciu, D., Yagoub, K.: Optimization of run-time management of data intensive web sites. In: Proc 25th VLDB Conf, Edinburgh, Scotland (1999) 627–638
10. Haas, L.M., Kossmann, D., Ursu, I.: Loading a cache with query results. In: Proc 25th VLDB Conf. (1999) 351–362
11. Kapitskaia, O., Ng, R.T., Srivastava, D.: Evolution and revolutions in LDAP directory caches. In: Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings. Volume 1777 of Lecture Notes in Computer Science., Springer (2000) 202–216
12. Keller, A.M., Basu, J.: A predicate-based caching scheme for client-server database architectures. VLDB Journal **5** (1996) 35–47
13. Kroeger, T.M., Long, D.D.E., Mogul, J.C.: Exploring the bounds of web latency reduction from caching and prefetching. In: USENIX Symposium on Internet Technologies and Systems. (1997)
14. Palmer, M., Zdonik, S.B.: Fido: A cache that learns to fetch. In: Proc 17th VLDB Conf, Barcelona, Spain (1991) 255–264
15. Silverston, L.: The Data Model Resource Book. Volume 2. John Wiley and Sons, New York (2001)

Diversity in Random Subspacing Ensembles

Alexey Tsymbal[1], Mykola Pechenizkiy[2], and Pádraig Cunningham[1]

[1]Department of Computer Science, Trinity College Dublin, Ireland
{Alexey.Tsymbal,Padraig.Cunningham}@cs.tcd.ie

[2]Department of Computer Science and Information Systems,
University of Jyväskylä, Finland
mpechen@it.jyu.fi

Abstract. Ensembles of learnt models constitute one of the main current directions in machine learning and data mining. It was shown experimentally and theoretically that in order for an ensemble to be effective, it should consist of classifiers having diversity in their predictions. A number of ways are known to quantify diversity in ensembles, but little research has been done about their appropriateness. In this paper, we compare eight measures of the ensemble diversity with regard to their correlation with the accuracy improvement due to ensembles. We conduct experiments on 21 data sets from the UCI machine learning repository, comparing the correlations for random subspacing ensembles with different ensemble sizes and with six different ensemble integration methods. Our experiments show that the greatest correlation of the accuracy improvement, on average, is with the disagreement, entropy, and ambiguity diversity measures, and the lowest correlation, surprisingly, is with the Q and double fault measures. Normally, the correlation decreases linearly as the ensemble size increases. Much higher correlation values can be seen with the dynamic integration methods, which are shown to better utilize the ensemble diversity than their static analogues.

1 Introduction

A popular method for creating an accurate classifier from a set of training data is to construct several classifiers, and then to combine their predictions. It was shown in many domains that an ensemble is often more accurate than any of the single classifiers in the ensemble. Dietterich [6] has presented the integration of multiple classifiers as one of the four most important directions in machine learning research.

Both theoretical and empirical research have demonstrated that an ensemble is good if the base classifiers in it are both accurate and tend to err in different parts of the instance space (that is have diversity in their predictions). Another important issue in creating an effective ensemble is the choice of the function for combining the predictions of the base classifiers. It was shown that if the integration method does not properly utilize the ensemble diversity, then no benefit arises from integrating multiple models [3].

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 309–319, 2004.

One effective approach for generating an ensemble of diverse base classifiers is the use of different feature subsets, or so-called *ensemble feature selection* [12]. By varying the feature subsets used to generate the base classifiers, it is possible to promote diversity. One efficient way to do ensemble feature selection is the random subspace method or random subspacing [9].

Measuring diversity is not straightforward – there are a number of ways to measure diversity in ensembles of classifiers, and not much research has been done about the appropriateness and superiority of one measure over another.

In this paper, we consider different measures of the ensemble diversity, which could be used to measure the total ensemble diversity, as a general characteristic of ensemble goodness. The goal of this paper is to compare the considered measures of diversity in the context of random subspacing with different integration methods and with different ensemble sizes. In the existing literature, comparing different measures of the ensemble diversity is normally done by analyzing their correlation with various other ensemble characteristics. Such characteristics are the ensemble accuracy, and the difference between the ensemble accuracy and the average or maximal base classifier accuracy [11,15]. In this paper, we compare eight measures of diversity with regard to their correlation with the accuracy improvement due to ensembles.

The paper is organized as follows. In Section 2 we review ensemble feature selection and random subspacing. In Section 3 we consider the question of integration of an ensemble of classifiers and review different integration methods. In Section 4 we present eight different measures for diversity in classification ensembles. In Section 5 we present our experiments with these measures and conclude in the next section with a summary and assessment of further research topics.

2 Ensemble Feature Selection and Random Subspacing

The task of using an ensemble of models can be broken down into two basic questions: (1) what set of learned models should be generated?; and (2) how should the predictions of the learned models be integrated? [6].

One way for building models with homogeneous representations, which proved to be effective, is the use of different subsets of features for each model, also known as *ensemble feature selection* [12].

Ho [9] has shown that simple random selection of feature subsets may be an effective technique for ensemble feature selection because the lack of accuracy in the ensemble members is compensated for by their diversity. This technique is called the random subspace method or simply Random Subspacing (RS).

Instead of selecting a fixed number of features as in [9] (she used approximately half of the features for each base classifier) we use probabilistic feature selection in our implementation of RS. We consider all the features as having equal probability of being selected to the feature subset. This probability is selected randomly from the interval (0,1) before defining each feature subset. Thus, the initial feature subsets include different numbers of features. It was shown in experiments in [18] that this implementation of RS provides ensembles with greater diversity, and consequently, greater accuracy.

RS has much in common with bagging [17], but instead of sampling instances, one samples features. Like bagging, RS is a parallel learning algorithm, that is, the generation of each base classifier is independent. This makes it suitable for parallel implementation for fast learning. It was shown that, like in bagging, the ensemble accuracy could be only increased with the addition of new members, even when the ensemble complexity grew [9].

3 Integration of an Ensemble of Models

Brodley and Lane [3] have shown that simply increasing diversity of an ensemble is not enough to insure increased prediction accuracy. If an integration method does not utilize diversity, then no benefit arises from integrating multiple classifiers.

A number of *selection* and *combination* approaches have been proposed in the literature. One of the most popular and simplest techniques used to *combine* the results of the base classifiers, is simple voting [1]. Weighted Voting (WV), where each prediction receives a weight proportional to the estimated generalization performance of the corresponding classifier, works usually better than the simple majority voting [1].

One of the most popular and simplest *selection* techniques is Cross-Validation Majority (CVM, we call it simply Static Selection, SS, in our experiments) [14]. In CVM, the cross-validation accuracy for each base classifier is estimated, and then the classifier with the highest accuracy is selected.

The described above approaches are *static*. They select one "best" model for the whole data space or combine the models uniformly. In *dynamic* integration each new instance to be classified is taken into account. Usually, better results can be achieved with dynamic integration.

We consider in our experiments three dynamic integration techniques based on the same local accuracy estimates: Dynamic Selection (DS) [13], Dynamic Voting (DV) [13], and Dynamic Voting with Selection (DVS) [19]. At the learning phase, they estimate the local classification errors of each base classifier for each instance of the training set according to the 1/0 loss function using cross validation. The learning phase finishes with training the base classifiers on the whole training set. The application phase begins with determining *k*-nearest neighbourhood for a new instance using a distance metric. Then, weighted nearest neighbour regression is used to predict the local classification errors of each base classifier for the new instance.

Then, DS simply selects a classifier with the least predicted local error. In DV, each base classifier receives a weight that is proportional to the estimated local accuracy. In DVS, the base classifiers with highest local errors are discarded (the classifiers with errors that fall into the upper half of the error interval of the base classifiers) and locally weighted voting (DV) is applied to the remaining base classifiers.

4 Measures of the Ensemble Diversity

In this section we consider eight different measures of the ensemble diversity, six of which are pairwise as they are able to measure diversity in predictions of a pair of classifiers. The total ensemble diversity is the average of the diversities of all the pairs of classifiers in the ensemble. The two non-pairwise measures evaluate diversity in predictions of the whole ensemble (entropy and ambiguity).

The *plain disagreement* measure is probably the most commonly used measure for diversity in the ensembles of classifiers with crisp predictions. For example, in [9] it was used for measuring the diversity of decision forests, and its correlation with the forests' accuracy. In [18] it was used as a component of the fitness function guiding the process of ensemble construction. For two classifiers *i* and *j*, the plain disagreement is equal to the proportion of the instances on which the classifiers make different predictions:

$$div\_plain_{i,j} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{Diff}(\mathrm{C}_i(\mathbf{x}_k), \mathrm{C}_j(\mathbf{x}_k)) , \tag{1}$$

where *N* is the number of instances in the data set, $\mathrm{C}_i(\mathbf{x}_k)$ is the class assigned by classifier *i* to instance *k*, and Diff(*a*,*b*)=0, if a=b, otherwise Diff(*a*,*b*)=1.

The *fail/non-fail disagreement* is the percentage of test instances for which the classifiers make different predictions but for which one of them is correct [16]:

$$div\_dis_{i,j} = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}} , \tag{2}$$

where N^{ab} is the number of instances in the data set, classified correctly (*a*=1) or incorrectly (*a*=0) by the classifier *i*, and correctly (*b*=1) or incorrectly (*b*=0) by the classifier *j*. (2) is equal to (1) for binary classification problemsIt can be also shown that $div\_dis_{i,j} \le div\_plain_{i,j}$.

The *Double Fault* measure (*DF*) [8] is the percentage of test instances for which both classifiers make wrong predictions:

$$div\_DF_{i,j} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}} , \tag{3}$$

where N^{ab} has the same meaning as in (2). In [11,15] *DF* was shown to have reasonable correlation with the Majority Voting and Naïve Bayes integration methods.

The following measure is based on Yule's *Q statistic* used to assess the similarity of two classifiers' outputs [11]:

$$div\_Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} , \tag{4}$$

where N^{ab} has the same meaning as in (2) and (3). In [11] *Q* was recommended as the best measure for the purposes of developing ensembles, taking into account the experimental results, and especially its simplicity and comprehensibility.

One problem, which we have noticed with this measure in our pilot studies, was its insensitivity on small data sets. For a small number of instances N^{00} is often equal to 0. *Q* in this case is equal to –1 (maximal diversity) no matter how big the values of

N^{01} and N^{10} are, which is not a good reflection of the true differences in classifiers' outputs.

The *correlation coefficient* between the outputs of two classifiers i and j can be measured as [11]:

$$div\_corr_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11}+N^{10})(N^{01}+N^{00})(N^{11}+N^{01})(N^{10}+N^{00})}}, \tag{5}$$

where N^{ab} have the same meaning as in (2), (3) and (4). The numerator in (5) is the same as in (4), and for any two classifiers i and j, $div\_corr_{i,j}$ and $div\_Q_{i,j}$ have the same sign, and it can be proven that $\left|div\_corr_{i,j}\right| \le \left|div\_Q_{i,j}\right|$ [11]. This measure, as well as the fail/non-fail disagreement, the DF measure, and the Q statistic were considered among the group of 10 measures in the comparative experiments in [11].

Let N_{ij} be the number of instances in the data set, recognized as class i by the first classifier and as class j by the second one, N_{i*} is the number of instances recognized as i by the first classifier, and N_{*i} is the number of instances recognized as i by the second classifier. Define then Θ_1 and Θ_2 as

$$\Theta_1 = \frac{\sum_{i=1}^{l} N_{ii}}{N}, \text{ and } \Theta_2 = \sum_{i=1}^{l}\left(\frac{N_{i*}}{N}\cdot\frac{N_{*i}}{N}\right), \tag{6}$$

where l is the number of classes and N is the total number of instances. Θ_1 estimates the probability that the two classifiers agree, and Θ_2 is a correction term for Θ_1, which estimates the probability that the two classifiers agree simply by chance. The pairwise diversity $div\_kappa_{i,j}$ is then defined as follows [5]:

$$div\_kappa_{i,j} = \frac{\Theta_1 - \Theta_2}{1-\Theta_2}. \tag{7}$$

Dietterich [5] used this measure in scatter plots called "κ-error diagrams", where kappa was plotted against mean accuracy of the classifier pair. κ-error diagrams are a useful tool for visualising ensembles.

A non-pairwise measure of diversity, associated with a conditional-entropy error measure, is based on the concept of *entropy* [4]:

$$div\_ent = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{l} -\frac{N_k^i}{S}\cdot\log_l\left(\frac{N_k^i}{S}\right), \tag{8}$$

where N is the number of instances in the data set, S is the number of base classifiers, l is the number of classes, and N_k^i is the number of base classifiers that assign instance i to class k. This measure was evaluated on a medical prediction problem and was shown to predict the accuracy of the ensemble well [4].

The next non-pairwise measure of diversity is associated with the variance-based measure of diversity proposed for regression problems in [10], called *ambiguity*. This diversity has been proven to have a direct relation with the ensemble error and this motivated us to use an associated diversity measure for classification also. The classification task can be decomposed into l regression tasks, where l is the number of classes. The output in the regression tasks will be the class membership of the instance (binary output 0/1 in the case of crisp classification considered in this paper). The diversity of the classification ensemble can then be calculated as the average ambiguity over these pseudo-regression tasks for each of the instances:

$$div\_amb = \frac{1}{lN}\sum_{i=1}^{l}\sum_{j=1}^{N} Ambiguity_{i,j} = \frac{1}{lNS}\sum_{i=1}^{l}\sum_{j=1}^{N}\sum_{k=1}^{S}\left(\mathrm{Is}(\mathrm{C}_k(\mathbf{x}_j)=i)-\frac{N_i^j}{S}\right)^2 , \tag{9}$$

where l is the number of classes, N is the number of instances, S is the number of base classifiers, N_i^j is the number of base classifiers that assign instance j to class i, $\mathrm{C}_k(\mathbf{x}_j)$ is the class assigned by classifier k to instance j, and Is() is a truth predicate.

In our experiments we normalize all the measures to vary from 0 to 1, where 1 corresponds to the maximum of diversity for the sake of simplicity and to avoid the unnecessary complication in understanding the results of correlations with different signs.

5 Experimental Investigations

The experiments are conducted on 21 data sets taken from the UCI machine learning repository [2]. These data sets include real-world and synthetic problems, vary in characteristics, and were previously investigated by other researchers. For our experiments, we used an updated version of the experimental setting presented in [18] to test the EFS_SBC algorithm (Ensemble Feature Selection with the Simple Bayesian Classification). We extended it with an implementation of seven new measures of diversity besides the existing plain disagreement.

We used Simple Bayes (SB) as the base classifier in the ensembles. It has been recently shown experimentally and theoretically that SB can be optimal even when the “naïve” feature-independence assumption is violated by a wide margin [7]. Second, when SB is applied to the sub-problems of lower dimensionalities as in random subspacing, the error bias of the Bayesian probability estimates caused by the feature-independence assumption becomes smaller. It also can easily handle missing feature values of a learning instance allowing the other feature values still to contribute. Besides, it has advantages in terms of simplicity, learning speed, classification speed, and storage space. It was shown [18] that only one “global” table of Bayesian probabilities is needed for the whole ensemble when SB is employed in ensemble feature selection (for each feature of the base classifiers the corresponding probabilities from this table are simply taken). We believe that dependencies and conclusions presented in this paper do not depend on the learning algorithm used and would be similar for most known learning algorithms.

To estimate the ensemble performance with random subspacing, we have used 70 test runs of stratified random-sampling cross validation with 70 percent of instances in the training sets. We experimented with five different ensemble sizes: 5, 10, 25, 50, and 100. At each run of the algorithm, we collect accuracies for the six types of ensemble integration: Static Selection (SS), Majority Voting (V), Weighted Voting (WV), Dynamic Selection (DS), Dynamic Voting (DV), and Dynamic Voting with Selection (DVS). In the dynamic integration strategies DS, DV, and DVS, the number of nearest neighbors (*k*) for the local accuracy estimates was pre-selected from the set of seven values: 1, 3, 7, 15, 31, 63, 127 ($2^n - 1$, $n = 1,...,7$), for each data set separately, if the number of instances in the training set permitted. Heterogeneous Euclidean-Overlap Metric (HEOM) [13] was used for calculation of the distances (for numeric features, the distance is calculated using the Euclidean metric, and for categorical features the simple 0/1 overlap metric is used).

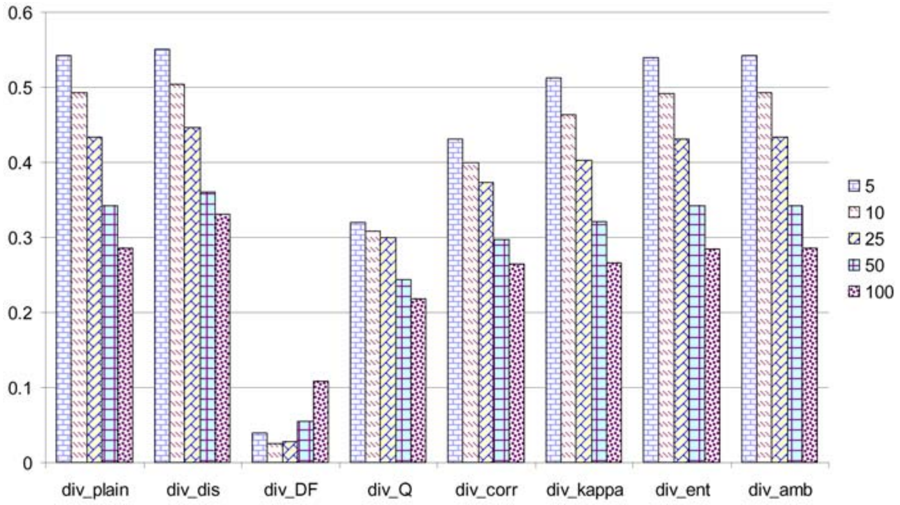


Fig. 1. The correlations for the eight diversities and five ensemble sizes averaged over the data sets and integration methods

In Figure 1 the correlations between diversity and improvement in the classification accuracy due to ensembles (the difference between the ensemble accuracy and the average base classifier accuracy) for the eight diversities and five ensemble sizes averaged over the data sets and integration methods are shown (Pearson's correlation coefficient *r* is used). It can be seen from the picture that the highest correlation is with *div_dis*. *Div_plain*, *div_ent*, and *div_kappa* are very close to the best *div_dis* (the difference is at most 0.03 for each ensemble size). The lowest correlation values are with *div_corr*, *div_Q*, and *div_DF*. Surprisingly, the worst correlations are with the *div_Q* and especially *div_DF* measures. As could be seen from the results, *Div_Q* and *div_corr* behave in a similar way, which reflects the similarity in their formulae. An-

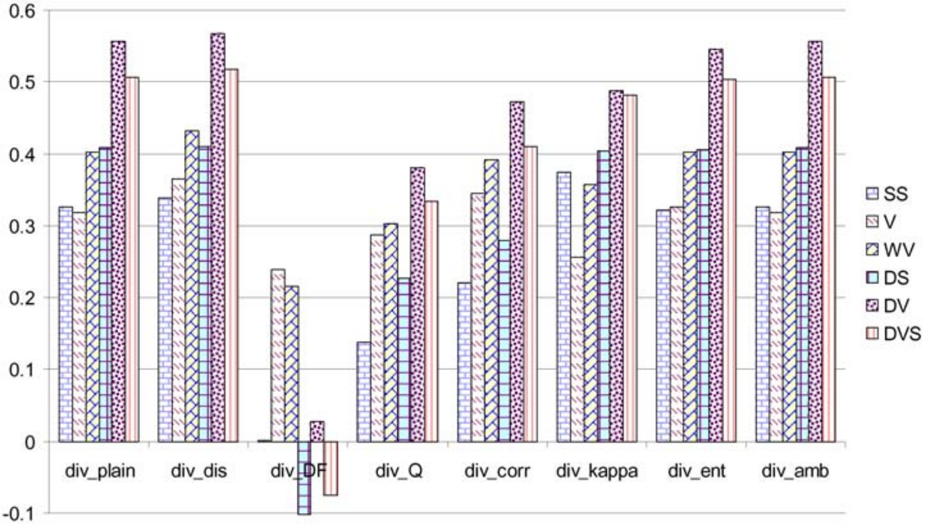


Fig. 2. The correlations for the eight diversities and six integration methods averaged over the data sets and ensemble sizes

other interesting finding is that the correlations decrease approximately linearly with the increase in the ensemble size. The best correlations are shown for 5 base classifiers. This is not true for the *div_DF* measure, where there is no clear pattern in the change (probably because the correlation is not significant for this measure).

In Figure 2 the correlations between diversity and improvement in the classification accuracy due to ensembles for the eight diversities and six integration methods averaged over the data sets and ensemble sizes are shown using Pearson's *r* correlation coefficient. The ranking of the diversities is the same as previously reported (in order of goodness): *div_dis*, *div_plain*, *div_amb*, *div_ent*, *div_kappa*, *div_corr*, *div_Q*, and *div_DF*. We also noticed that the correlation values are almost the same for *div_plain* and *div_amb*. The difference was at most 0.001, probably due to rounding in the computations. Supposedly, this similarity can be explained theoretically. The correlations differ significantly with the six integration methods. Always the dynamic methods (DS, DV, and DVS) have better correlations than the static ones (SS, V, and WV). Normally WV has better correlations than the other two static methods (SS and V). We believe that these differences can be explained by the fact that the dynamic methods make better use of diversity than the static methods, and WV makes better use of diversity than SS and V. These dependencies do not hold true for *div_DF* again, because of the same reason of low correlations.

To check the presented dependencies we recalculated the correlations using Spearman's rank correlation coefficient (*RCC*) as suggested in [11]. All the trends remained the same, and the difference in the averaged values was at most 0.01 and in the particular correlation values - at most 0.05.

To validate the findings and conclusions presented before and to check the dependency of the results on the selection of the data sets, we divided all the data sets into two groups in the following two ways: (1) with greater than the average improvement due to ensembles (10 data sets), and with less than or equal to the average improvement (11 data sets); and (2) with less than 9 features (10 data sets), and with greater or equal to 9 features (11 data sets); and checked all the dependencies for these groups.

The results for these groups supported our previously reported findings in this paper (the ranking of the diversities, the correlation decrease with the ensemble size's increase, and the ranking of the integration methods). Expectedly, the correlations for the group with better improvements were greater than for the other group (by up to 0.15 on average). Unexpectedly, greater correlations (by up to 0.15 on average) were for the group with larger amounts of features than for the group with fewer features. This needs further research.

We noticed also interesting behaviour with the selected *k*-neighbourhood values for dynamic integration. DS needs higher values of *k*. This can be explained by the fact that its prediction is based on only one classifier being selected, and thus, it is very unstable. Higher values of *k* provide more stability to DS. The average selected *k* is equal to 33 for DS, and it is only 14 for DV. For DVS, as a hybrid strategy, it is in between at 24 (for the ensemble size of 5). The selected values of *k* do not change significantly with the change of the ensemble size. The only change noticed is that DS with more ensemble members needs even greater *k* (up to 43 for 100 ensemble members).

6 Conclusions

In our paper, we have considered eight ensemble diversity metrics, six of which are pairwise measures (the plain disagreement, *div_plain*; the fail/non-fail disagreement, *div_dis*; the Double Fault measure, *div_DF*; the *Q* statistic, *div_Q*; the correlation coefficient, *div_corr*; and the *kappa* statistic, *div_kappa*), and two are non-pairwise measures (entropy, *div_ent*; and ambiguity, *div_amb*). To integrate the base classifiers generated with random subspacing, we used six integration methods: Static Selection (SS), Majority Voting (V), Weighted Voting (WV), Dynamic Selection (DS), Dynamic Voting (DV), and Dynamic Voting with Selection (DVS). We considered five ensemble sizes: 5, 10, 25, 50, and 100.

In our experiments, to check the goodness of each measure of diversity, we calculated its correlation with the improvement in the classification accuracy due to ensembles. The best correlations were shown by *div_plain*, *div_dis*, *div_ent*, and *div_amb*. *Div_Q* and *div_corr* behaved in a similar way, supported by the similarity of their formulae. Surprisingly, *div_DF* and *div_Q* had the worst average correlation. The correlation coefficients for *div_amb* were almost the same as for *div_plain*. All the correlations changed with the change of the integration method, showing the different use of diversity by the integration methods. The best correlations were shown with DV. The correlations decreased almost linearly with the increase in the ensemble size. The best correlations are for the ensemble size 5.

It would be interesting to check the presented dependencies and conclusions in other contexts in the future. For example, other ensemble generation strategies and integration methods can be tried.

Acknowledgments

This material is based upon works supported by the Science Foundation Ireland under Grant No. S.F.I.-02IN.1I111. This research is partly supported by the COMAS Graduate School of the University of Jyväskylä, Finland.

References

1. Bauer E., R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Machine Learning, 36 (1,2) (1999) 105-139.
2. Blake C.L., E. Keogh, C.J. Merz, UCI repository of machine learning databases [http:// www.ics.uci.edu/ ~mlearn/ MLRepository.html], Dept. of Information and Computer Science, University of California, Irvine, CA, 1999.
3. Brodley C., T. Lane, Creating and exploiting coverage and diversity, in: Proc. AAAI-96 Workshop on Integrating Multiple Learned Models, Portland, OR, 1996, pp. 8-14.
4. Cunningham P., J. Carney, Diversity versus quality in classification ensembles based on feature selection, in: R.L. deMántaras, E. Plaza (eds.), Proc. ECML 2000 11th European Conf. On Machine Learning, Barcelona, Spain, LNCS 1810, Springer, 2000, pp. 109-116.
5. Dietterich T.G., An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2) (2000) 139-157.
6. Dietterich T.G., Machine learning research: four current directions, AI Magazine 18(4) (1997) 97-136.
7. Domingos P., M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning, 29 (2,3) (1997) 103-130.
8. Giacinto G., F. Roli. Design of effective neural network ensembles for image classification processes. *Image Vision and Computing Journal*, 19(9-10):699–707, 2001.
9. Ho T.K., The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (8) (1998) 832-844.
10. Krogh A., J. Vedelsby, Neural network ensembles, cross validation, and active learning, In: D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems, Vol. 7, Cambridge, MA, MIT Press, 1995, pp. 231-238.
11. Kuncheva L.I., C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2) (2003) 181-207.
12. Opitz D., Feature selection for ensembles, in: Proc. 16th National Conf. on Artificial Intelligence, AAAI Press, 1999, pp. 379-384.

13. Puuronen S., V. Terziyan, A. Tsymbal, A dynamic integration algorithm for an ensemble of classifiers, in: Z.W. Ras, A. Skowron (eds.), Foundations of Intelligent Systems: 11th Int. Symp. ISMIS'99, Warsaw, Poland, LNAI 1609, Springer, 1999, pp. 592-600.
14. Schaffer C., Selecting a classification method by cross-validation, Machine Learning 13 (1993) 135-143.
15. Shipp C.A., L.I. Kuncheva, Relationship between combination methods and measures of diversity in combining classifiers, Information Fusion 3 (2002) 135-148.
16. Skalak D.B., The sources of increased accuracy for two proposed boosting algorithms, in: AAAI-96 Workshop on Integrating Multiple Models for Improving and Scaling Machine Learning Algorithms (in conjunction with AAAI-96), Portland, Oregon, USA, 1996, pp. 120-125.
17. Skurichina M., R.P.W. Duin, Bagging and the random subspace method for redundant feature spaces, in: J. Kittler, F. Roli (Eds.), Proc. 2nd Int. Workshop on Multiple Classifier Systems MCS 2001, Cambridge, UK, 2001, pp. 1-10.
18. Tsymbal A., S. Puuronen, D. Patterson, Ensemble feature selection with the simple Bayesian classification, *Information Fusion*, Elsevier Science 4 (2) (2003) 87-100.
19. Tsymbal A., S. Puuronen, I. Skrypnyk, Ensemble feature selection with dynamic integration of classifiers, in: Int. ICSC Congress on Computational Intelligence Methods and Applications CIMA'2001, Bangor, Wales, U.K, 2001, pp. 558-564.

Partitioned Approach to Association Rule Mining over Multiple Databases*

Himavalli Kona and Sharma Chakravarthy

CSE Department, The University of Texas at Arlington
{hima,sharma}@cse.uta.edu

Abstract. Database mining is the process of extracting interesting and previously unknown patterns and correlations from data stored in Data Base Management Systems (DBMSs). Association rule mining is the process of discovering items, which tend to occur together in transactions. If the data to be mined were stored as relations in multiple databases, instead of moving data from one database to another, a partitioned approach would be appropriate. This paper addresses the partitioned approach to association rule mining for data stored in multiple Relational DBMSs. This paper proposes an approach that is very effective for partitioned databases as compared to the main memory partitioned approach. Our approach uses SQL-based K-way join algorithm and its optimizations. A second alternative that trades accuracy for performance is also presented. Our results indicate that beyond a certain size of data sets, the accuracy is preserved in addition to improving performance. Extensive experiments have been performed and results are presented for the two partitioned approaches using IBM DB2/UDB and Oracle 8i.

1 Introduction

Association rule mining [3, 4, 5] makes correlation between items that are grouped into transactions deducing rules that define relationships between itemsets. Here an attempt is made to identify if customer who buys item 'A' also buys item 'B'. Association rules are of the form A => B where A is the antecedent and B is the consequent. Several association rule algorithms have been proposed that work on data in a file [5, 6, 7]. Data base approach to association rules using SQL have been explored as well [8, 9, 11, 13, 14].

Parallel mining algorithms have been developed to overcome the limitations of main memory approaches by using the aggregate power and memory of many processors. A multi-database system [1, 2] is a federation of autonomous and heterogeneous database systems. Data is distributed over multiple databases (typically 2 or 3) in many organizations. Each of the databases may get updated frequently and independently. Most of the organizations today have multiple data sources distributed at differ-

* This work was supported, in part, by the Office of Naval Research, the SPAWAR System Center-San Diego & by the Rome Laboratory (grant F30602-02-2-0134), and by NSF (grants IIS-0123730 and ITR 0121297). We would like to acknowledge the contributions of Akshay Arora in improving the readability of this paper.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 320–330, 2004.

ent locations, which need to be analyzed to generate interesting patterns and rules. An effective way to deal with multiple data sources (where data to be mined is distributed among several relations on different DBMSs) is to mine the association rules at different sources and forward the rules to other systems rather than sending the data to be mined which is likely to be very large. Interactive mining has been proposed as a way to bring decision makers into the loop to enhance the utility of mining and to support goal oriented mining. Partitioned and incremental approaches can be applied to each of the data sources independently, which would require the transmission of intermediate results between the databases.

If the raw data from each of the local databases were sent to a common database for mining and generation of rules, certain useful rules, which would aid in making decisions about local datasets, would be lost. For example a rule such as "50% of the branches in the north saw a 10% increase in the purchase of printers when digital cameras and memory cards were purchased together" would not be generated if the raw data was transferred and processed as a whole. In such a case the organization may miss out certain rules that were prominent in certain branches and were not found in the other branches as in the above example. Generating such rules would aid in making decisions at each branch independently.

This paper addresses the problem of directly mining data stored in multiple relations or multiple databases. The main memory approaches are not applicable here unless data is siphoned out of each relation/database which is what we are trying to avoid in the first place. The cost models for data transfer in our case are very different from that of partitioned main memory algorithms. Furthermore, the utility of mining each data set independently as well as together is important for many applications. In this paper, we address the partitioned approaches to discover association rules on data residing in multiple databases. Although incremental approaches have been developed, we will not discuss them for lack of space. Please refer to [17] for details.

The rest of the paper is organized as follows: In Section 2 discusses some of the related work in this area. Section 3 discusses the inefficiency of the main memory partitioned approach when directly used for multiple databases. Section 4 presents our approaches and their performance evaluation. Section 5 has conclusions.

2 Related Work

The partition algorithm for association rules presented in [7] makes at most two passes over the input database to generate the rules. In the first phase of the algorithm the database is divided into non-overlapping partitions and each of the partitions are mined individually to generate the local frequent itemsets. At the end of the first phase the local frequent itemsets are merged to generate the global candidate itemsets. In the second phase of the algorithm a TIDLIST is created and used to calculate the support of all the itemsets in the global candidate itemset to generate the global frequent itemsets. The algorithm could be executed in parallel to utilize the capacity of many processors with each processor generating the rules.

SQL-Based approaches map transactions into relations with (TID, Item) attribute format [3, 13]. The (TID, Itemlist) format was not chosen because the number of items for a particular transaction may exceed the number of attributes a DBMS can support. SQL-92 and SQL-OR approaches [13, 14, 16] were introduced for mining data in relational DBMSs. The SQL-92 based approaches correspond to k-way join and its optimizations such as the Second Pass Optimization (SPO), Reuse of Item Combinations (RIC) and Prune the input table (PI). It has been shown [11, 16] that, of all the SQL-based approaches, K-way join and its optimizations have the best performance (as compared to query-sub query and group by approaches).

Most of the times data is already stored as relations in different databases belonging to the same organization. If one were to mine the data present in multiple databases, there are two options. [2] presents a weighted model for synthesizing high-frequency association rules from different sources. A high-frequency rule is the one that is supported by most of the data sources. The proposed model assigns a high-weight to a data source that supports/votes more high-frequency rules and a lower weight to a data source that supports/votes less high-frequency rules. [1] discusses a new multi-database mining process. The patterns in multi-databases are divided into the three classes as Local patterns, high-vote patterns and Exceptional patterns. The mining strategy identifies two types of patterns, high-vote patterns and exceptional patterns. The discovery of these patterns can capture certain distributions of local patterns and assist global decision-making within a large company.

3 Performance of the Partitioned Approach

The partition algorithm [7] is an efficient main memory algorithm for mining association rules in large data sets that could be partitioned as needed. This approach when used for multiple databases results in poor performance as will be shown in the paper. For its implementation, relational operations (using SQL) were used. There was no change in Phase I of the algorithm. Each database was considered an individual partition and the frequent itemsets were generated for each of the databases. In Phase II of the algorithm the frequent itemsets from each of the partitions were merged to form two sets of itemsets. The first set is the global frequent itemsets, which correspond to itemsets that are large in all the partitions (databases). The second set is the set of global candidate itemsets, which is the union of all the frequent itemsets from each of the partitions (not included in the global frequent itemsets). A TIDLIST is created for the entire database. As the data is assumed to be distributed over different databases, each partition needs to be shipped to a single database to create the TIDLIST. The TIDLIST was used for counting the support of the itemsets in the global candidate itemsets and the itemsets satisfying the user specified support were added to the set of global frequent itemsets.

3.1 Methodology for Experiments

The performance results presented in this paper are based on datasets generated synthetically using IBM's data-generator. The nomenclature of these datasets is of the form "TxxIyyDzzzK", where "xx" denotes the average number of items present per transaction, "yy" denotes the average support of each item in the dataset and "zzzK" denotes the total number of transactions in "K"(1000s). The experiments have been performed on Oracle 8i and IBM DB2 / UDB V7.2 (installed on a machine running Microsoft Windows 2000 Server with 512MB of RAM). Each experiment has been performed 4 times. The values from the first run are ignored so as to avoid the effect of the previous experiments and other database setups. The average of the next 3 runs was used for analysis to avoid any false reporting of time due to system overload or other factors. For the purpose of reporting experimental results in this paper, due to space constraints, we have shown the results only for three datasets – T5I2D100K, T5I2D500K, and T5I2D1000K.

Figure 1. shows the performance of the TIDLIST approach for a T5I2D1000K dataset. The dataset is divided into two equal partitions each of size 500K. The analysis of the time taken for the different phases shows that the Phase II is the most time consuming. In Phase II, the TIDLIST is created for the whole dataset. The TIDLIST creation time increases exponentially as the size of the dataset increases. The partitioned approach although seems to work well for main memory databases, its performance for partitioned databases is not acceptable. The creation of the TIDLIST and the shipping of the partitions to a single database need to be avoided.

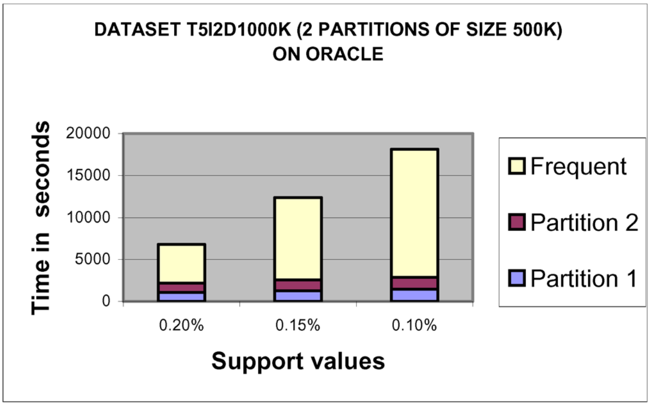


Fig. 1. Performance Of TIDLIST Approach On T5I2D1000K Dataset

4 Proposed Algorithms for Multiple Databases

This section discusses two partitioned approaches – DB-Partition I (or approach I in figures) and DB-partition II (or approach II in figures) – that have been developed and tested for multiple databases. Notations used in the remainder of this paper are shown in Table 1.

Table 1. Notation used for the Partitioned Approach

Notation	Meaning
C^P_K	Set of local candidate k-itemsets in partition P.
F^P_K	Set of local frequent k-itemsets in partition P.
C^G_K	Set of global candidate k-itemsets.
F^G_K	Set of global frequent k-itemsets.
$NBd(F^P_K)$	Set of local non-frequent k-itemsets in partition P.

The negative border of frequent k-itemsets corresponds to those itemsets that did not satisfy the support in pass k. That is, $NBd(F^P_K) = C^P_K - F^P_K$. Given a collection $F \subseteq P(R)$ of sets, closed with respect to set inclusion relation, the NBd(F) of F consists of the minimal itemsets $X \subseteq R$ not in F.

4.1 DB-Partition I

In the TIDLIST approach, the TIDLIST was created as a CLOB. In DB-partition I, the TIDLIST is not at all created and the k-way join approach is used instead. Some of the k-way join optimizations reported in [14, 16] have been used. The two k-way join optimizations used are: Second-pass Optimization (SPO) and Reuse of Item Combinations (RIC). In a multiple database scenario, each of the individual databases is considered as a partition and the merging is done by choosing one of the databases. The changes made to the partition algorithm are described below.

Phase I: In this phase the frequent itemsets F^P_K are generated for each of the partitions. Along with the frequent itemsets in each of the partitions, the negative border of the frequent 2-itemsets $NBd(F^P_2)$ is also retained. These itemsets are used for counting the support in the Phase II of the algorithm. Only the negative border of the 2-itemsets is retained because when the second pass optimization is used, the generation of the 2-itemsets is the first step in each partition. Since the 2-itmeset generation is the first pass, there is no loss of information and the negative border of the 2-itemsets will have all possible 2-itemsets, which did not satisfy the support.

After the frequent itemsets from all the partitions (databases) are generated, the frequent itemsets and the negative border of the frequent 2-itemsets from all the partitions are shipped to one of the databases to do the remaining computation. This step is shown as an edge with label “1” in Figure 2. Merging the frequent itemsets from all the partitions generates the global candidate itemsets C^G_1, C^G_2, …, C^G_K.

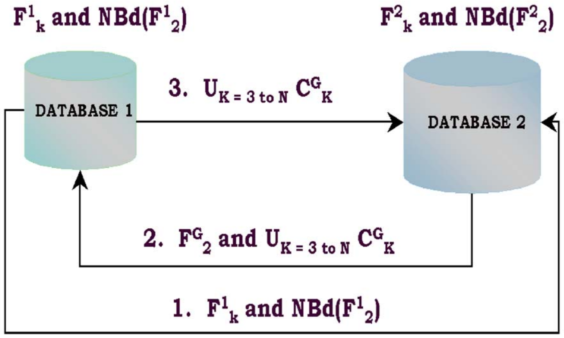

F^1_k and $NBd(F^1_2)$
F^2_k and $NBd(F^2_2)$
DATABASE 1
3. $U_{K=3 \text{ to } N}\ C^G_K$
DATABASE 2
2. F^G_2 and $U_{K=3 \text{ to } N}\ C^G_K$
1. F^1_k and $NBd(F^1_2)$

Fig. 2. Data Transfer Using DB-partition I

Phase II: In this phase, the global frequent itemsets – itemsets that are large in all the partitions – are generated. Merging the count obtained from the negative border and the frequent 2-itemsets from all the partitions generates the count for the remaining 2-itemsets in C^G_2. The itemsets satisfying the support are added to F^G_2. F^G_2 and

$\cup_{k=3 \text{ to } n} C^{G}_{K}$ are shipped to all the databases to generate the counts of the remaining candidate itemsets. This is shown as an edge with label “2” in Figure 2.

Each of the databases generates a materialized table from the global frequent 2-itemsets using the *Reuse of item combination* optimization. The materialized table is used in the successive passes to generate the counts of the itemsets in the global candidate itemsets. Once the counts are generated in all the partitions they are shipped back to one database to do the final counting. This is shown as an edge with label “3” in Figure 2.

Figure 2 shows the data transferred in each of the steps. Database 1 and Database 2 are considered the 2 partitions. Database 2 is chosen for merging the frequent itemsets from all the partitions to global candidate itemset and for generating the final cumulative count of all frequent itemsets obtained from all the partitions in step “3”.

Performance Analysis: Performance experiments were done on datasets of different sizes. Each data set divided into 2 or 3 non-overlapping partitions. Figure 3. shows the performance of a T5I2D1000K dataset divided into 2 equal sized partitions each of size 500K. It can be seen from the graph that the improvement in performance in DB-partition I compared to TID LIST approaches 58% for a support values of 0.20% and the improvement increases to about 78% for a support value of 0.10%. As the support value decreases the percentage improvement in the performance increases.

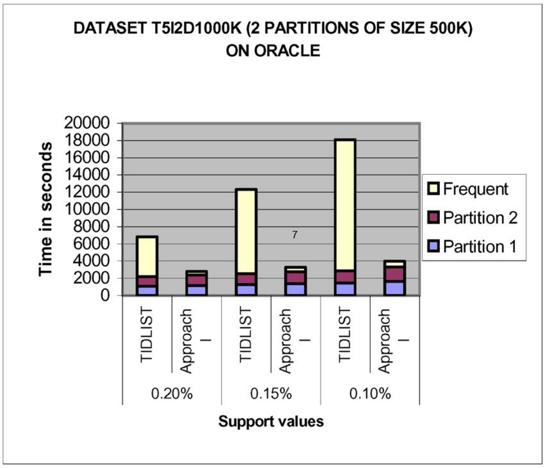


Fig. 3. Performance Comparison Of TIDLIST And DB-partition I

Figure 4. shows the data transfer when there are 3 partitions. At the end of each phase the intermediate results are transferred to one of the partitions to do the remaining computations.

The performance for T5I2D500K is shown in Figure 5. The dataset is divided into 3 partitions of size 200K, 200K and 100K. The performance is shown for Oracle and DB2. For DB2 the percentage improvement in performance decreases from 80% to 53% as the support value decreases from 0.30% to 0.20%. The performance in Oracle shows an increase from 18% to 75% as the support value decreases from 0.20% to 0.10%.

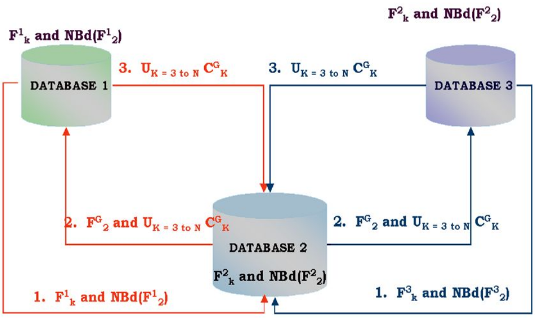

F^1_k and $NBd(F^1_2)$
F^2_k and $NBd(F^2_2)$
DATABASE 1
DATABASE 3
3. $U_{K=3\text{ to }N}\,C^G_K$
3. $U_{K=3\text{ to }N}\,C^G_K$
2. F^G_2 and $U_{K=3\text{ to }N}\,C^G_K$
2. F^G_2 and $U_{K=3\text{ to }N}\,C^G_K$
DATABASE 2
F^2_k and $NBd(F^2_2)$
1. F^1_k and $NBd(F^1_2)$
1. F^3_k and $NBd(F^3_2)$

Fig. 4. Data Transfer Using DB-partition I For 3 Partitions

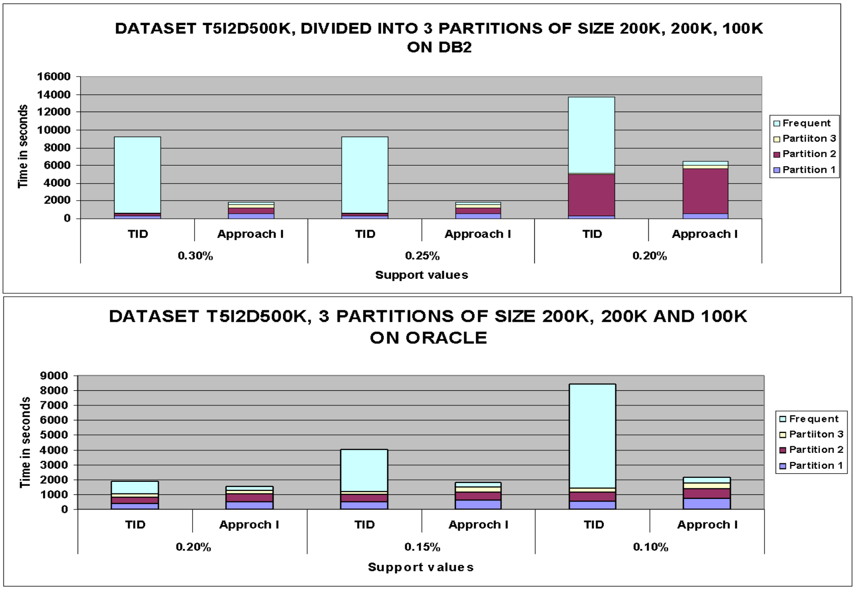


Fig. 5. Performance Comparison Of T5I2D500K For TIDLIST And DB-partition I

Data Transfer: Table 2 shows the number of records transferred between the databases in each step. The input data denotes the transactional data. It is assumed that the dataset is divided into 2 equal sized partitions. The numbers in the Table 2 indicate the number of records transferred. For example, for the T5I2D10K dataset the input data has 27000 records and the total records transferred using DB-partition I between the two databases is 51845. It is observed that transferring the intermediate results is better for the datasets, which have more than 100K transactions (which is typically the case).

In DB-partition I only the negative border of the frequent 2-itemsets was retained in all the partitions. In phase II, a materialized table was created to do the support counting. The time taken to create a materialized table increases as the size of the dataset increases. In this approach the data is transferred 3 times between the partitions. DB-partition II alternative was proposed to overcome the above drawbacks.

Table 2. Data Transferred Using DB_partition I

Dataset	Step 1 $[F^1{}_K + NBd(F^1{}_2)]$	Step 2 $[F^G{}_2 + U_{K=3\ to\ N}\ C^G{}_K]$	Step 3 $U_{K=3\ to\ N}\ C^G{}_K$	Total records
T5I2D10K	50360	1393	92	51845
T5I2D100K	199455	678	101	200234
T5I2D500K	319677	638	76	320391
T5I2D1000K	356696	632	75	357403

4.2 DB-Partition II

During the Phase I of this approach, the negative border of all the frequent itemsets in each of the partitions are retained as compared to the previous approach where only the negative border of the frequent 2-itemsets were retained. When the frequent itemsets are generated the data is transferred to one of the partitions (or databases) to form the global candidate itemsets and the global frequent itemsets. The global frequent k-itemsets are generated by merging the counts of the frequent k-itemsets and the negative border of the frequent k-itemsets. Figure 6 shows the data transfer in DB-partition II.

DB-partition II is different from DB-partition I with regard to the number of times data is transferred between the databases and the itemsets that are retained. In DB-partition I, only the negative border of the frequent 2-itemsets is retained. Since retaining the negative border does not require any additional computation, in DB-partition II, the negative border of all the frequent itemsets are retained for all the databases. In Phase II of DB-partition I, the global frequent 2-itemsets are generated using the local frequent 2-itemsets and their negative border from all the databases. The results have to be transferred to the individual databases to generate the remaining (3 to k) – itemsets, which requires the scanning the input data in each of the databases to generate the counts. But in DB-partition II, all the global frequent itemsets are generated using the local frequent itemsets and their negative border from all the databases. An additional scan of the database is **not** required and the intermediate results are transferred only once as compared to 3 times in DB-partitition I.

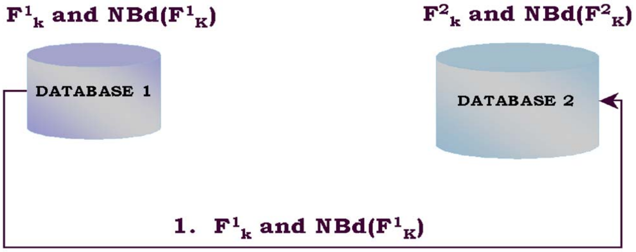

$F^1{}_k$ and $NBd(F^1{}_K)$
$F^2{}_k$ and $NBd(F^2{}_K)$
DATABASE 1
DATABASE 2
1. $F^1{}_k$ and $NBd(F^1{}_K)$

Fig. 6. Data Transfer In DB-partition II

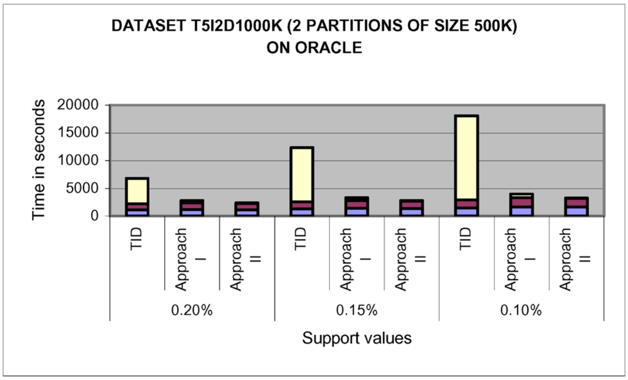


Fig. 7. Performance Comparison Of All The Approaches With 2 Partitions

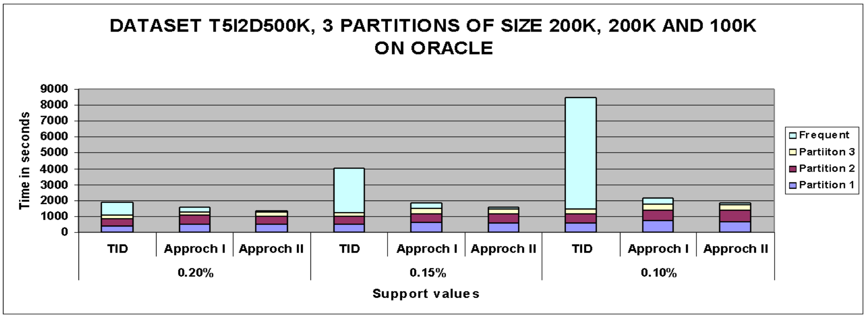


Fig. 8. Performance Comparison Of All The Approaches With 3 Partitions

Figure 7 shows the performance comparison of the TIDLIST, DB-partition I and DB-partition II. A T5I2D1000K dataset was divided into 2 partitions of size 500K each. From the graph it is noted that the performance of DB-partition II improved from 16% to 18% as the support value decreased from 0.20% to 0.10%. Figure 8 shows a similar performance graph for 3 partitions.

We compared the data transfer between DB-partition I and DB-partition II (Table 3) and they were not much different as the size of the data is large for the second pass (2-itemsets). It was noted that DB-partition II performed better than TIDLIST and DB-partition I for almost all the cases. This was because the creation of materialized table was eliminated and retaining the negative border does not require any additional computation. However there is a tradeoff associated with the DB-partition II. This approach may miss out the count of itemsets, which are globally large but locally small in a few partitions. The count of some k-itemsets whose subset did not appear either in the frequent itemsets or its negative border in the earlier passes may be missed. Our intuition was that this was the case only for small data sizes that too when the support is extremely low.

In order to verify this, we conducted experiments using DB-partition II for different data sizes. Figure 9 shows the error observed in the frequent itemsets generated. The frequent itemsets generated using the TIDLIST approach and DB-partition I was compared with the itemsets generated in DB-partition II. This approach showed some error in the number of frequent itemsets generated in each pass. However, it was seen that there was some error only for the smaller datasets with lower support values. No error was noted beyond datasets of size T5I2D100K and above.

Table 3. Data Transfer For DB-partition II

Dataset	**Input data Records**	**Step 1** [F^1_K + NBd (F^1_K)] records	**DB-partition II**
T5I2D10K	27000	50481	50481
T5I2D100K	273000	199521	199521
T5I2D500K	1368500	319740	319740
T5I2D1000K	2736000	356761	356761

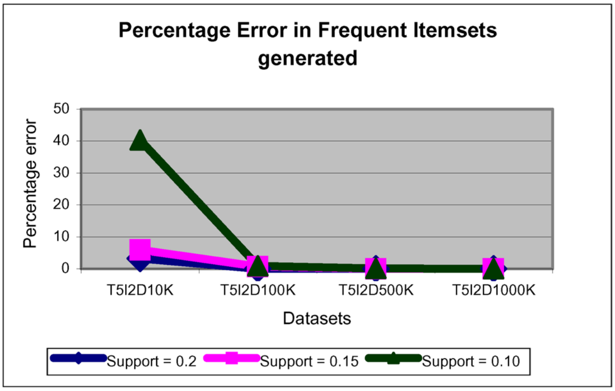


Fig. 9. Error Analysis

5 Conclusions

In this paper, we have focused on the partitioned approach to association rule mining that can be used for multiple databases. The partitioned approach to association rule mining is appropriate for mining data stored in multiple DBMSs. The partition algorithm proposed in this paper provides an efficient way of discovering association rules in large multi-database. This paper presented two approaches – DB-partition I and DB-partition II using the negative border concept which possesses slightly different performance characteristics. Extensive experiments have been performed for the partitioned approach on Oracle 8i and IBM DB2.

References

1. Zhang, S., X. Wu, and C. Zhang, *Multi-Database Mining*. IEEE Computational Intelligence Bulletin, Vol. 2, No. 1, 2003: p. 5-13.
2. Wu, X. and S. Zhang. *Synthesizing High-Frequency Rules from Different Data Sources*. in *IEEE TKDE* 2003.

3. Thomas, S., *Architectures and optimizations for integrating Data Mining algorithms with Database Systems*, Ph.D Thesis, *CISE.* Department 1998, University of Florida, Gainesville.
4. Thuraisingham, B., *A Primer for Understanding and Applying Data Mining.* IEEE, 2000. Vol. 2, No.1: p. 28-31.
5. Agrawal, R., T. Imielinski, and A. Swami. *Mining Association Rules between sets of items in large databases.* in Proc. of *ACM SIGMOD* 1993.
6. Agrawal, R. and R. Srikant. *Fast Algorithms for mining association rules.* in *20th Int'l Conference on Very Large Databases,.* 1994.
7. Savasere, A., E. Omiecinsky, and S. Navathe. *An efficient algorithm for mining association rules in large databases.* in *21st Int'l Cong. on Very Large Databases (VLDB).* 1995. Zurich, Switzerland.
8. Houtsma, M. and A. Swami. *Set-Oriented Mining for Association Rules in Relational Databases.* in Proc. of ICDE , 1995.
9. Meo, R., G. Psaila, and S. Ceri. *A New SQL-like Operator for Mining Association Rules.* in *Proc. of VLDB* 1996. Mumbai, India.
10. Agrawal, R. and K. Shim, *Developing tightly-coupled Data Mining Applications on a Relational Database System.* 1995, IBM Almaden Research Center: San Jose, California.
11. Sarawagi, S., S. Thomas, and R. Agrawal. *Integrating Association Rule Mining with Relational Database System: Alternatives and Implications.* in *ACM SIGMOD* 1998. Seattle, Washington.
12. Hongen, Z., *Mining and Visualization of Association Rules over Relational DBMSs*, MS Thesis *CISE Department* 2000, UFL, Gainesville.
13. Dudgikar, M., *A Layered Optimizer or Mining Association Rules over RDBMS*, MS Thesis, *CISE Dept.* 2000, University of Florida, Gainesville.
14. Mishra, P. and S. Chakravarthy, *Performance Evaluation and Analysis of SQL-92 Approaches for Association Rule Mining* in *Proc.* of *DAWAK* 2003.
15. Toivonen, H. *Sampling Large Databases for Association Rules. In Proc. of VLDB.* 1996.
16. Mishra, P. and S. Chakravarthy, *Evaluation of K-way Join and its variants for Association Rule Mining in Proc. of BNCOD 2002.*
17. Kona, H, *Association Rule Mining Over Multiple Databases: Partitioned and Incremental Approaches*, MS Thesis, UTA, Fall 2003.
http://www.cse.uta.edu/Research/Publications/Downloads/CSE-2003-40.pdf

A Tree Partitioning Method for Memory Management in Association Rule Mining

Shakil Ahmed, Frans Coenen, and Paul Leng

Department of Computer Science, The University of Liverpool
Liverpool L69 3BX, UK
{shakil,frans,phl}@csc.liv.ac.uk

Abstract. All methods of association rule mining require the *frequent sets* of items, that occur together sufficiently often to be the basis of potentially interesting rules, to be first computed. The cost of this increases in proportion to the database size, and also with its density. Densely-populated databases can give rise to very large numbers of candidates that must be counted. Both these factors cause performance problems, especially when the data structures involved become too large for primary memory. In this paper we describe a method of partitioning that organises the data into tree structures that can be processed independently. We present experimental results that show the method scales well for increasing dimensions of data, and performs significantly better than alternatives, especially when dealing with dense data and low support thresholds.

1 Introduction

The most computationally demanding aspect of Association Rule Mining is identifying the *frequent sets* of attribute-values, or *items*, whose *support* (occurrence in the data) exceeds some threshold. The problem arises because the number of possible sets is exponential in the number of items. For this reason, almost all methods attempt to count the support only of c*andidate* itemsets that are identified as possible frequent sets. It is, of course, not possible to completely determine the candidate itemsets in advance, so it will be necessary to consider many itemsets that are not in fact frequent. Most algorithms involve several passes of the source data, in each of which the support for some set of candidate itemsets is counted. The performance of these methods, clearly, depends both on the size of the original database and on the number of candidates being considered. The number of possible candidates increases with increasing density of data (greater number of items present in a record) and with decreasing support thresholds. In applications such as medical epidemiology, where we may be searching for rules that associate rather rare items within quite densely-populated data, the low support-thresholds required may lead to very large candidate sets. These factors motivate a continuing search for efficient algorithms.

Performance will be affected, especially, if the magnitudes involved make it impossible for the algorithm to proceed entirely within primary memory. In these cases, some strategy for *partitioning* the data may be required to enable algorithmic stages to be carried out on primary-memory-resident data. In this paper we examine methods of partitioning to limit the total primary memory requirement, including that required both for the source data and for the candidate sets. We describe a new method of partitioning that exploits tree structures we have previously developed for Association

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 331–340, 2004.

Rule Mining. Experimental results are presented that show this method offers significant performance advantages.

2 Background

Most methods for finding frequent sets are based to a greater or lesser extent on the "Apriori" algorithm [2]. Apriori performs repeated passes of the database, successively computing support-counts for sets of single items, pairs, triplets, and so on. At the end of each pass, sets that fail to reach the required support threshold are eliminated, and candidates for the next pass are constructed as supersets of the remaining (frequent) sets. Since no set can be frequent which has an infrequent subset, this procedure guarantees that all frequent sets will be found.

A problem of Apriori is that it requires the source data to be scanned repeatedly, which is especially costly if this cannot be contained in primary memory. Attempts to reduce this cost include the "Partition" algorithm [12], which partitions the data into a number of equal-sized segments of manageable size, and the strategy introduced by Toivonen [13], which first processes a small random sample of the data to identify candidates for counting across the full database. The drawback of both these approaches highlights the second weakness of Apriori: that the number of candidates whose support must be counted may become very large. Both methods increase the size of the candidate set, and also require all candidates to be retained in primary memory (for efficient processing) during the final database pass. Other methods [3] [4] [1] [14] aim to identify *maximal* frequent sets without first examining all their subsets. These algorithms may cope better with densely-populated databases and long patterns than the others described, but again usually involve multiple database passes. The *DepthProject* [1] algorithm bypasses the problem by explicitly targeting memory-resident data. The method of Zaki et. al. [14] partitions candidate sets into clusters which can be processed independently. The problem with the method is that, especially when dealing with dense data and low support thresholds, expensive preprocessing is required before effective clustering can be identified. The partitioning by *equivalence class*, however, is relevant to the methods we will describe.

Our methods begin by performing a single pass of the database to perform a partial summation of the support totals. These partial counts are stored in a tree structure that we call the *P*-tree [9], which enumerates itemsets counted in lexicographic order. The *P*-tree contains all the sets of items present as distinct records in the database, plus some additional sets that are leading subsets of these. To illustrate, consider a database with items {a,b,c,d,e}, and a set of 20 records: {abcde,abce,abd,abde,abe,acde, ace,ade,b,bcde,bce,bd,bde,be,cd,cde,ce,d,de,e}. (Not necessarily in this order). For convenience, we will use the notation abd, for example, to denote the set of items {a,b,d}. Figure 1 shows the *P*-tree that would be constructed. The counts stored at each node are *incomplete* support-totals, representing support derived from the set and its succeeding supersets in the tree. For example, suppose the record cde is the last to be added. The tree-construction algorithm will traverse the nodes c and cd that are subsets of cde, incrementing the count for each, before appending cde as a child of cd.

We apply to this structure an algorithm, Apriori-TFP, which completes the summation of support counts, storing the results in a second set-enumeration tree (the *T*-tree), which finally contains all frequent sets with their complete support-counts. The algorithm used, essentially a form of Apriori that makes use of the partial counting that has already been done, is described in [6], where we explain the rationale of the approach, and present experimental results that demonstrate performance gains in comparison both with Apriori, and also the *FP-growth* [10] algorithm, which has some similar properties. The *FP*-tree used in [10] is a more pointer-rich structure than the *P*-tree, leading to greater difficulties in dealing with non-memory-resident data, although strategies for this have been proposed, which will be discussed further below. The CATS tree, an extension of the FP-tree proposed in [5], also assumes no limitation on main memory capacity. In this paper we consider implementations of Apriori-TFP when we cannot contain all the data required in main memory, requiring some strategy for partitioning this.

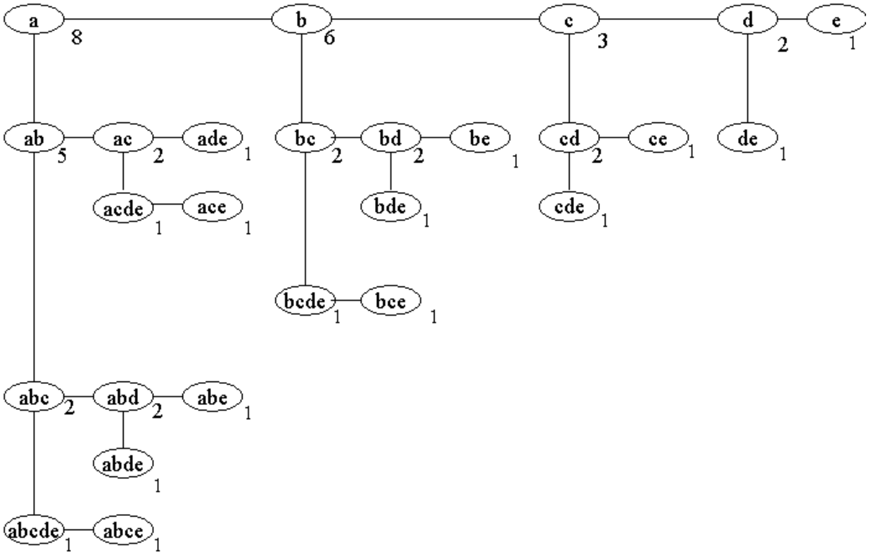


Fig. 1. Example of a P-tree

3 Strategies for Partitioning

The natural implementation of Apriori, when source data cannot be contained in primary memory, requires all the data to be read from secondary memory in each pass. The equivalent for Apriori-TFP, because the first stage of the method involves the construction of a *P*-tree, requires a partitioning of the data into segments of manageable size. We will refer to this form of partitioning, in which each segment contains a number of complete database records, as 'horizontal' partitioning (HP), or *segmentation.* We first take each segment of data separately and create for it a *P*-tree that is then stored in secondary memory. Each pass of Apriori-TFP then requires each of the *P*-trees to be read in turn from secondary memory. The method creates a single final *T*-tree in primary memory, which contains all the frequent sets and their support-counts.

The drawback of this simple approach is that it replicates the two weaknesses of the Apriori methodology: all the data (now in the form of *P*-trees) must be re-read from secondary memory in each pass, and the entire *T*-tree, which contains candidates and, finally, all the frequent sets, must be kept in primary memory while counting proceeds. As we have noted, this tree may be very large. Even when it can be held in primary memory, a large set of candidates leads to slower counting, in Apriori-TFP just as for Apriori. The *P*-tree structure offers another possible form of partitioning, into subtrees that represent equivalence classes of the items represented. However, it is not possible to compute the support for a set by considering only the subtree in which it is located. Although succeeding supersets of a set *S* in the *P*-tree are located in the subtree rooted at *S*, predecessor supersets are scattered throughout the preceding part of the *P*-tree. For example, consider the support for the set bd in the data used for Figure 1. In the subtree rooted at b, we find a partial support total for bd, which includes the total for its superset bde. To complete the support count for bd, however, we must add in the counts recorded for its preceding supersets bcde, abd (incorporating abde) and abcde, the latter two of which are in the subtree headed by a.

The problem can be overcome by a different partitioning of the *P*-tree structure. Our Tree Partitioning (TP) method begins by dividing the ordered set of items into subsequences. For example, for the data used in Figure 1, we might define 3 sequences of items, {a,b}, {c,d} and {e}, labelled 1,2,3 respectively. For each sequence we define a *Partition-P*-tree (*PP-tree),* labelled PP1, PP2 and PP3. The construction of these is a slight modification of the original method. The first, PP1, is a proper *P*-tree that counts the partial support for the power set of {a,b}. PP2, however, counts all those sets that include a member of {c,d} in a tree that includes just these items and their predecessors. The third tree, PP3, will count all sets that include any member of {e}. The three trees obtained, from our example, are illustrated in Figure 2. The *PP*-trees are, in effect, overlapping partitions of the *P*-tree of Figure 1, with some restructuring resulting from the omission of nodes when they are not needed.

The effect of this is that the total support for any set *S* can now be obtained from the *PP*-tree corresponding to the last item within *S*; for example, we now find all the counts contributing to the support of bd are included in PP2. The drawback is that the later trees in the sequence are of increasing size; in particular, PP3 in our example is almost as large as the original *P*-tree. We can overcome this by a suitable reordering of the items. In descending order of their frequency in the data, the items of our example are e,d,b,c,a. Using the same data as for Figures 1 and 2, we will construct *PP*-trees using this ordering, for the sets of items {e,d}, {b,c} and {a} respectively.

The results are shown in Figure 3. Now, because the less frequent items appear later in the sequence, the trees become successively more sparse, so that PP3 now has only 13 nodes, compared with the 23 of PP3 in Figure 2. Our previous work has shown [7] that ordering items in this way leads to a smaller *P*-tree and faster operation of Apriori-TFP. The additional advantage for partitioning is that the *PP*-trees become more compact and more equal in size. The total support-count for bd (now ordered as db) is again to be found within PP2, but now requires the addition of only 2 counts (db+edb).

This form of partitioning offers us a way of dividing the source data into a number of *PP*-trees each of which may then be processed independently. The partitioning, as illustrated in Figure 3, is essentially similar to that obtained by the construction of

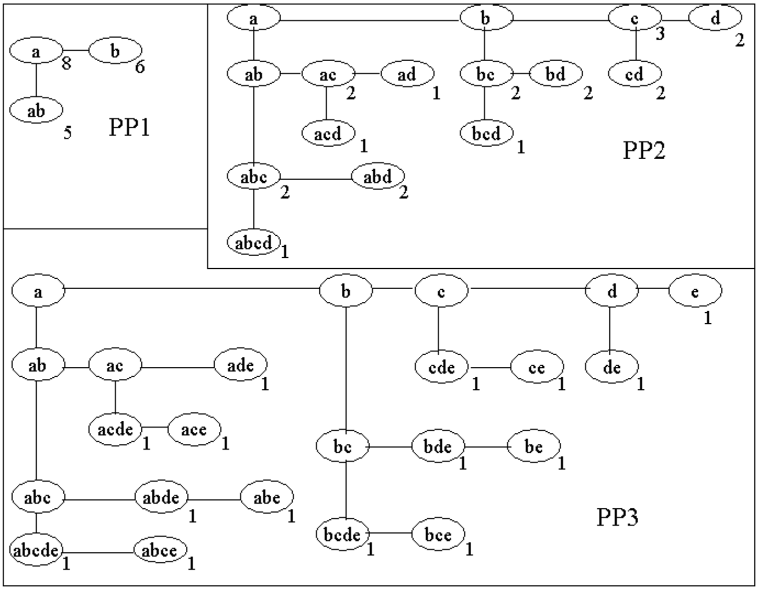


Fig. 2. Partition-P-trees from figure 1

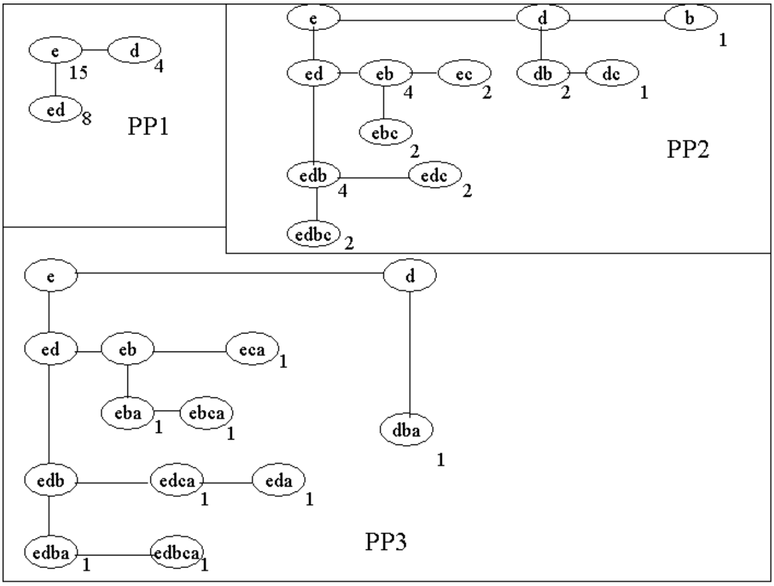


Fig. 3. PP-trees after reordering of items

conditional databases described in [10] and [11], and the COFI-trees proposed in [8]. The latter method also creates subtrees that can be processed independently, but requires an initial construction of an *FP*-tree that must be retained in primary memory for efficient processing. The partitioning strategy proposed in [11] for dealing with an *FP*-tree too large for primary storage, would first construct the *a-conditional* database corresponding to PP3, and after building the *FP*-tree for this, would copy relevant transactions (e.g. edbca), into the next (*c-conditional*) database, as edbc. The method we describe avoids this multiple copying by constructing all the trees in memory in a single pass, during which we also partially count support. With a sufficiently large data set, it will of course still not be possible to construct the *PP*-trees within primary memory. We can, however, combine this approach with a (horizontal) segmentation of the original data into segments small enough to allow the corresponding *PP*-trees to be contained in primary store. This approach is possible in our case because the *P*-tree structure is simpler and less pointer-rich than the corresponding *FP*-tree. Our method also allows us to define *PP*-trees which are, in effect, the union of a sequence of conditional databases, allowing more flexibility of partitioning.

The overall method is as follows:

1. Obtain an (at least approximate) ordering of the frequency of items, and using this ordering, choose an appropriate partitioning of the items into *n* sequences 1, 2, 3,..etc.
2. Divide the source data into *m* segments.
3. For each segment of data, construct *n* *PP*-trees in primary memory, storing finally to disk. This involves just one pass of the source data.
4. For partition 1, read the PP1 trees for all segments into memory, and apply the Apriori-TFP algorithm to build a *T*-tree that finds the final frequent sets in the partition. This stage requires the PP1 trees for each segment of data to be read once only. The *T*-tree remains in memory throughout, finally being stored to disk.
5. Repeat step 4 for partitions 2, 3, ..*n*.

The method offers two advantages. First, we have now effectively reduced the number of disk passes to 2: one (step 3) to construct the *PP*-trees, and a second pass (of the stored trees) to complete the counting (steps 4 and 5). The second advantage is that we are now, at each stage, dealing with smaller tree structures, leading to faster traversal and counting.

4 Results

To investigate the performance of the method, we used synthetic data sets constructed using the QUEST generator described in [2]. The programs were written in standard C++ and run on a 1.3 GHz machine with 256 Kb of cache, and 512 Mb of RAM. The data was stored on an NFS server (1Gb filestore).

We first establish that the method scales acceptably; i.e. the partitioning strategy successfully constrains the maximum requirement for primary memory, without leading to unacceptable execution times. For this purpose we generated data sets with parameters T10.I5.N500, i.e an average record size of 10 items. We divided the data into segments of 50,000 records, and within each segment generated 500 partitions, i.e. a *PP*-tree for each item. In all our experiments, the tree partitioning is naïve: after ordering the items, we partition into sequences of equal length (in this case, 1). In fact, our experiments seem to show that increasing the degree of partitioning always reduces the primary memory requirement (as would be expected), and also almost always reduces execution time. The latter, less obvious result arises because the increased time taken to construct a greater number of *PP*-trees (step 3 above) is usually more than compensated by the faster processing of smaller *T*-trees (steps 4 and 5).

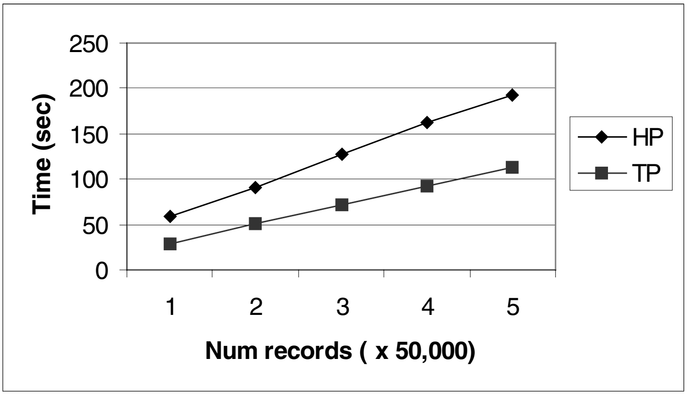


Fig. 4. Execution times for T10.I5.N500 (0.01% support)

Figure 4 shows the overall time to generate frequent sets, with a support threshold of 0.01%, for datasets of increasing size, i.e. 1,2,3,4 and 5 segments. The figure compares the performance of the Tree-partitioning method (TP) with the simple method involving horizontal partitioning (segmentation) only (HP). The times illustrated include both the time to construct the trees and to execute the Apriori-TFP algorithm. As can be seen, TP offers substantially better performance than HP, and its performance scales linearly with the size of the dataset.

Importantly, this performance is achieved within conservative memory requirements. In steps 4 and 5 of the TP method it is necessary to contain in memory all the *PP*-trees for one partition, and the corresponding *T*-tree for that partition. In the experiment of Figure 4, this led to a maximum memory requirement for the TP method that varied from 1.38Mb (1 segment) to 1.6Mb (5 segments). In general, larger data sets, requiring greater horizontal segmentation, lead to some increase in the combined size of the *PP*-trees, but this is relatively slight. By contrast, the HP method requires the *P*-tree for one data segment and the whole of the *T*-tree to be contained in primary store, leading to a maximum memory requirement of between 116 and 128 Mb in the case illustrated.

The combined sizes of the *PP*-trees for any one segment are, of course, greater than the size of a corresponding *P*-tree, which could be a constraint during the construction of the *PP*-trees. If so, the problem can be overcome by imposing a greater degree of horizontal segmentation. We found that the total memory requirement to contain all the *PP*-trees for any one segment decreases from 7.8 Mb, when no segmentation is applied, to a maximum of 0.22 Mb when there are 50 segments, with almost no overall increase in execution time. The method scales well with increasing number of items, also. We found execution time increased slowly and linearly as the number of items increased from 500 to 2500, with no general upward trend in memory requirement. This is because the *P*-tree is a conservative structure, which stores (in general) only those itemsets that are actually needed, so increasing the number of items with no increase in density has little effect. Conversely, the simple HP method has a rapidly increasing memory requirement in this case because of the greater size of the unpartitioned *T*-tree.

The performance of the method when dealing with more dense datasets is shown in figures 5 and 6. In these experiments, we generated databases with D=50000, N=500, varying the T and I parameters, and divided into 5 segments, with 10 items/partition. The small database size was chosen for convenience, but we imposed a requirement that only one partition can be retained in memory. We compared the TP method both with HP and with a method based on the "Negative Border" approach of [13] (labelled NB in the figures). In the latter, *P*-trees are first constructed for all segments, as for the HP method. Then, all the proportionately-frequent sets in the first segment of data are found, using a support threshold reduced to 2/3 of the original, and also retaining the negative border of the sets found, i.e. those sets which, although not themselves frequent, have no infrequent subsets. The frequent sets, with their negative border, are stored in a *T*-tree which is kept in store to complete the counts for these sets for the remaining segments. The reduced support threshold, and the inclusion of the negative border, make it very likely that all the finally frequent sets will be included in this tree, in which case the method requires the disk-stored *P*-trees to be read once only.

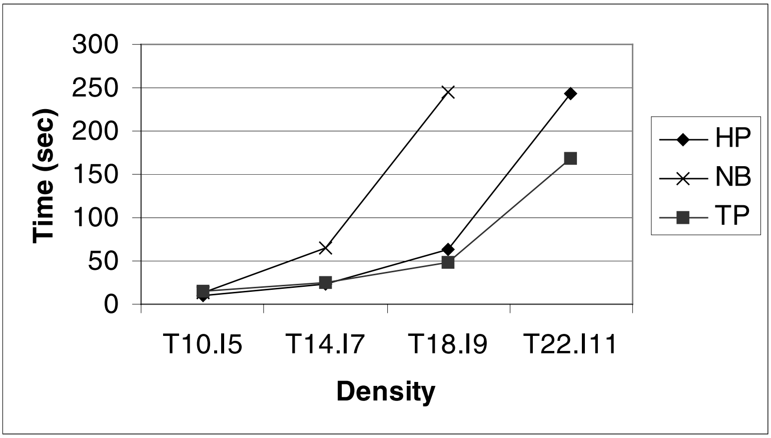


Fig. 5. Execution times for various database characteristics

The figures show results for a support threshold of 0.1%, at which level (higher than in the previous experiments) there is little difference in the total execution times for the T10.I5 data. In this case, the faster computation of the frequent sets by the TP method is offset by the longer time taken to construct the *PP*-trees. With more dense data, however, the latter factor becomes decreasingly significant, and the advantage of the TP method becomes increasingly apparent. This is principally because of the much smaller candidate sets that are involved. This is apparent from the comparison of maximal memory requirements, shown in Figure 6. This reflects the growing size of the candidate sets (and hence the *T*-tree) as the data density increases, leading both to larger memory requirements and to longer times to find candidates. The problem is particularly acute with the 'Negative Border' method. This works well with relatively sparse data, but at high density the reduced support threshold and the inclusion of the negative border lead to very large candidate sets. A similar pattern was observed in experiments varying the support threshold (for the T10.I5 set). At a threshold of 1.0, the overhead of constructing the multiple *PP*-trees for the TP method leads to relatively poor execution times, and in this case, the NB method is fastest. As the support threshold is reduced, however, the increasing cost of servicing a growing candidate set leads to rapidly increasing memory requirements and execution times for the alternative methods, whereas the TP method scales much better, becoming faster than either at a threshold of 0.05, and twice as fast at threshold 0.01.

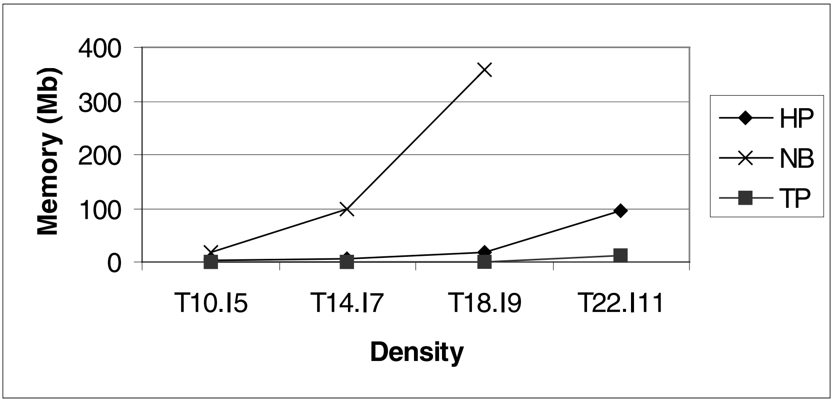


Fig. 6. Memory requirements for various data

5 Conclusions

In this paper we have examined ways of partitioning data for Association Rule Mining. Our aim has been to identify methods that will enable efficient counting of frequent sets in cases where the data is much too large to be contained in primary memory, and also where the density of the data means that the number of candidates to be considered becomes very large. Our starting point was a method which makes use of an initial preprocessing of the data into a tree structure (the *P-tree*) which incorporates a partial counting of support totals. In previous work we have shown this method to offer significant performance advantages. Here, we have investigated ways of applying the approach in cases that require the data to be partitioned for primary memory use. We have, in particular, described a method that involves a partitioning of the tree structures involved to enable separate subtrees to be processed independently. The advantage of this approach is that it allows both the original data to be partitioned into more manageable subsets, and also partitions the candidate sets to be counted. The latter results in both lower memory requirements and also faster counting.

The experimental results we have reported here show that the Tree Partitioning method described is extremely effective in limiting the maximal memory requirements of the algorithm, while its execution time scales only slowly and linearly with increasing data dimensions. Its overall performance, both in execution time and especially in memory requirements, is significantly better than that obtained from either simple data segmentation or a method that aims to complete counting in only two passes of the data. The advantage increases with increasing density of data and with reduced thresholds of support – i.e. for the cases that are in general most challenging for association rule mining. Furthermore, a relatively high proportion of the time required by the method is taken up in the preprocessing stage during which the *PP*-trees are constructed. Because this stage is independent of the later stages, in many applications it could be accepted as a one-off data preparation cost. In this case, the gain over other methods becomes even more marked.

References

1. Agarwal, R., Aggarwal, C. and Prasad, V. Depth First Generation of Long Patterns. In Proc. of the ACM KDD Conference on Management of Data, Boston, pages 108-118, 2000.
2. Agrawal, R. and Srikant, R. Fast Algorithms for Mining Association Rules. In Proc. of the 20th VLDB Conference, Santiago, Santiago, Chile, pages 487-499, September 1994.
3. Bayardo, R.J. Efficiently Mining Long Pattern from Databases. In Proc. of the ACM SIGMOD Conference on Management of Data, pages 85-93, 1998.
4. Bayardo, R.J., Agrawal, R. and Gunopulos, D. Constraint-Based Rule Mining in Large, Dense Databases. In Proc. of the 15th Int'l Conference on Data Engineering, 1999.
5. Cheung, W. and Zaiane, O.R. Incremental Mining of Frequent Patterns without Candidate Generation or Support Constraint. Proc. IDEAS 2003 Symposium 111-116
6. Coenen, F., Goulbourne, G., and Leng, P. Computing Association Rules using Partial Totals. PKDD 2001, pages 54-66, 2001.
7. Coenen, F. and Leng, P. Optimising Association rule Algorithms using Itemset Ordering. In 'Research and Development in Intelligent Systems XVIII (Proc ES2001 Conference, Cambridge), eds M Bramer, F Coenen and A Preece, Springer-Verlag, London, 2002, 53-66
8. El-Hajj, M. and Zaiane, O.R. Non Recursive Generation of Frequent K-itemsets from Frequent Pattern Tree Representations. Proc DAWAK 2003, pp 371-380
9. Goulbourne, G., Coenen, F. and Leng, P. Algorithms for Computing Association Rules Using a Partial-Support Tree. J. Knowledge-Based System 13 (2000), pages 141-149. (also Proc ES'99.)
10. Han, J., Pei, J. and Yin, Y. Mining Frequent Patterns without Candidate Generation. In Proc. of the ACM SIGMOD Conference on Management of Data, Dallas, pages 1-12, 2000.
11. Pei, J., Han, J. and Mao, R. CLOSET: an efficient algorithm for mining frequent closed itemsets. Proc ACM SIGMOD Workshop on Data Mining and Knowledge Discovery, 2000, 11-20
12. Savasere, A., Omiecinski, E. and Navathe, S. An Efficient Algorithm for Mining Association Rules in Large Databases. In Proc. of the 21th VLDB Conference, Zurich, Swizerland, pages 432-444, 1995.
13. Toivonen, H. Sampling Large Databases for Association Rules. In Proc. of the 22th VLDB Conference, Mumbai, India, pages 1-12, 1996.
14. Zaki, M.J. Parthasarathy, S. Ogihara, M. and Li, W. New Algorithms for fast discovery of association rules. Technical report 651, University of Rochester, Computer Science Department, New York. July 1997.

Mining Interesting Association Rules for Prediction in the Software Project Management Area

María N. Moreno García, Francisco J. García Peñalvo, and M. José Polo Martín

Dept. Informática y Automática, University of Salamanca, Salamanca, Spain
mmg@usal.es

Abstract. Association and classification are two data mining techniques traditionally used for solving different kind of problems. Association has been applied in knowledge discovery and classification in predictive tasks. Recent studies have shown that knowledge discovery algorithms can be successfully used for prediction in classification problems. The improvement of association rules algorithms is the subject of many works in the literature, whereas little research has been done concerning their classification aspect. On the other hand, methods for solving the problems of the association rules must be tailored to the particularities of the application domain. This work deals with the predictive use of association rules and addresses the problem of reducing the number of rules generated in the software project management field. We propose an algorithm for refining association rules based on incremental knowledge discovery. It provides managers with strong rules for decision making without need of domain knowledge.

1 Introduction

Association and classification techniques have been recognized as useful tools in making decisions in many application domains. Association is considered an unsupervised data-mining task, so it is mainly used in the area of knowledge discovery in databases, while classification problems are treated by means of supervised learning algorithms. This is a predictive modeling task, where the goal is to build a classification model (classifier) that relates the value of one attribute called the class label and the values of other attributes. The model is used later to make predictions about new, unlabeled data.

Traditionally, both kinds of techniques have been used to solve different classes of problems. However, recent studies show that knowledge discovery algorithms can be successfully used for predictive tasks in classification problems [12] [11] [22]. The improvement of the knowledge discovery algorithms, especially association rules, is the subject of many works in the literature, whereas little research has been done concerning their classification aspect. Association rule algorithms discover patterns in the form “IF X THEN Y”. If the consequence part (Y) of the rule is a class, these patterns can be used to predict the class of unclassified records.

The main application area of the association rules is market management. Deriving association rules from a sales transaction database provides managers with information about products that customers tend to buy together. This information can be used to drive promotional campaigns and to lead to more effective marketing. The purpose is to find items that imply the presence of other items in the same transaction. A pur-

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 341–350, 2004.

chase consisting of several articles (items) can be considered as a transaction. The process of deriving rules is simple [8], but it often generates a large set of rules, most of which are not useful. So, it is necessary to evaluate their validity. Most of the existing methods consider two factors, *support* and *confidence*, which capture the statistical strength of a pattern.

The problem of finding interesting rules has been widely studied [5] [18] [19]. The goal of these proposals is to obtain a reduced number of rules with high support and confidence. Subjective measures such as *unexpectedness* and *actionability* can be also used for determining the interestingness of a rule [18] [19] [14]. Unexpectedness indicates that rules are interesting if they are unknown to the users or contradict existing knowledge. Actionability means that users can take advantage of the rules [14]. Unfortunately these subjective factors cannot be easily obtained in some application areas such as project management, especially when a large number of quantitative attributes are involved and, so, it is very difficult to acquire domain knowledge. These applications present additional problems such as the discretization of continuous (quantitative) attributes that can take a wide range of values. In order to reduce the number of rules generated it is necessary to split the range of values into a manageable number of intervals.

On the other hand, a rule that is actionable in one application may not be in another. Therefore, methods for solving the problems of the association rules must be tailored to the particularities of each case.

This work deals with the predictive use of association rules and addresses the problem of reducing the number of rules generated in the project management field. We propose an iterative process to refine association rules, based on the generation of unexpected patterns. The principal feature of our approach is the incremental knowledge discovery from a gradual generation of the unexpected patterns by taking a single attribute in each iteration. We take advantage of knowledge of good attributes for classification and use them progressively, beginning with the best. This simplifies the selection of patterns and the refinement process. Besides, the algorithm not only provides a reduced number of rules but also selects the most useful rules for classification purposes. Another important aspect of the method is that the generation of unexpected patterns does not require domain knowledge. In our study context, it is very difficult to acquire experience due to the use of quantitative attributes. For this reason, the refinement method is very suitable for our purposes. We intend to predict the software size expressed in lines of code (LOC) in initial stages of a software project by using attributes that can be obtained early in the life cycle. This is necessary for estimating the project effort. Accurate estimations are critical in software project management because they lead to a good planning and reduce project costs.

The rest of the paper is organized as follows. Section 2 summarizes existing related research. In Section 3 the proposed method is presented. The experimental study is showed in section 4 and conclusion are exposed in section 5.

2 Related Work

Since Agrawal and col. introduced the concept of association between items [2] [1] and proposed the Apriori algorithm [3], association rules have been the focus of intensive research. Most of the efforts have been oriented to simplify the rule set and improve the algorithm performance.

The first pass of the Apriori algorithm consists of counting the number of occurrences of each item to determine the large itemsets, whose supports are equal or greater than the minimum support specified by the user. There are algorithms that generate association rules without generating frequent itemsets [12]. Some of them simplifying the rule set by mining a constraint rule set, that is a rule set containing rules with fixed items as consequences [5] [6]. The procedure causes loss of information, but it does not represent a drawback for predictive purposes when a single item, the class, constitutes the consequence part of the rules. Many other algorithms for obtaining a reduced number of interesting rules have been proposed, however this work is focused in the use of the rules in classification problems. The number of papers in the literature that consider this aspect is lesser. A proposal of this category is the CBA (Classification Based on Association) algorithm [13] that consists of two parts, a rule generator for finding association rules and a classifier builder based on the discovered rules. In [22] the weight of the evidence and association detection are combined for flexible prediction with partial information. The main contribution of this method is the possibility of making prediction on any attribute in the database. Moreover, new incomplete observations can be classified. The algorithm uses the weight of the evidence of the attribute association in the new observation in favor of a particular value of the attribute to be predicted. This approach uses all attributes in the observation, however in many domains some attributes have a minimal influence in the classification, so the process can be unnecessary complicated if they are taken in consideration. Our proposal evaluates the importance of the attributes on the classification.

Most of the commented methods consider two factors, *support* and *confidence*, which capture the statistical strength of a pattern, in order to select the best rules and reduce the generated rule set. However, these factors are useful neither for informing about rules convenience nor for detecting conflicts between rules. It is necessary to consider other factors in order to obtain consistent and interesting patterns. This is the motivation for rules refinement. The topic of knowledge refinement has been treated in the literature, but in the area of association rules little research has been done. In [18] and [19] the concept of unexpectedness is introduced in an iterative process for refining association rules. The authors have proposed methods to discover unexpected pattern in data. They use prior domain knowledge to reconcile unexpected patterns and to obtain stronger association rules. Domain knowledge is fed with the experience of the managers. This is a drawback for the use of the method in the project management environment because the rules are numeric correlations between project attributes and they are influenced by many factors (size and complexity of the software, programming language, etc.). It is very difficult to acquire experience in this class of problem. For this reason, in this work we present a refinement method more suitable for our purposes that does not need use managerial experience. It is also based on the discovery of unexpected patterns, but it uses "the best attributes for classification" in a progressive process for rules refinement. The best attributes are obtained by the technique of "importance of columns" [15] based on the amount of information (entropy) that the attributes provide in discriminating the classes (intervals of values of the label attribute LOC).

Our aim is not to improve the performance of the existing algorithms but propose an effective procedure for classification problems that is very suitable for applications that manage quantitative attributes where domain knowledge cannot be easily obtained. The aim is to provide managers with a convenient number of good association rules for prediction, which help them to make right decisions about the software project.

3 Association Based Classifier

In this work, an algorithm for generating and refining association rules is proposed in order to solve a classification problem. We fix the class label as the only item of the consequent part of the rules but we generate incrementally the rules by adding items progressively to the antecedent part. We use information about good attributes for classification and take them progressively, beginning with the best.

3.1 Finding the Best Attributes for Classification

The *label* attribute is the target of prediction in classification problems. A model that relates the label and the other attributes can be used for making predictions about new, unlabeled data. Importance of columns is a technique that determines the degree of utility of the descriptive attributes (columns) in discriminating the different values of the label attribute. The *purity* measure (a number from 0 to 100) informs about how well the columns discriminate the classes (different values of the label attribute). It is based on the amount of information (entropy) that the column (attribute) provides [15]. The expression for that information (I) is:

$$\mathrm{I}\,(P(c_1), ..., P(c_n)) = \sum_{i=1}^{n} - P(c_i)\log_{\mathrm{n}} P(c_i)$$

where $P(c_i)$ is the probability of the class i and n is the number of classes.

The information is 1 when the classes have the same probabilities.

The purity is defined as 1 – I. Every set in the partition has its own purity measure, and the purity of the partition is a combination of these individual measures. For a given set in the partition, the purity is 0 if each class has equal weight and 100 if every record belongs to the same class.

3.2 Association Rules and Unexpectedness

Mining association rules from data is very useful in many applications domains, mostly for finding purchase patterns in commercial environments. Algorithms for discovering rules, such as “Apriori” [2], are not complex; however, they usually generate a large number of rules representing obvious or irrelevant patterns, even restricting them with high values of support and confidence. In this work, we present a refinement algorithm that generates the association rules in an incremental way and produces the best rules considering their predictive purpose. The process is based on the concept of unexpectedness. This concept was introduced by Padmanabhan and Tuzhilin in order to generate useful patterns by using manager domain knowledge. They generate *unexpected patterns* versus the managerial experience and use them to refine rules representing beliefs [18]. We also use that concept but without needing domain knowledge . The foundations of association rules and unexpectedness [19] are introduced below.

A set of discrete attributes At={a_1 ,a_2 ,..... ..,a_m } is considered. Let D={T_1 ,T_2 ,..... ..,T_N } be a relation consisting on N transactions T_1 ,..... ..,T_N over the relation schema {a_1 ,a_2 ,..... ..,a_m }. Also, let an atomic condition be a proposition of the form $value_1 \leq attribute \leq value_2$ for ordered attributes and *attribute* = *value* for unordered attributes, where *value*, $value_1$ and $value_2$ belong to the set of distinct values taken by attribute in D. Finally, an itemset is a conjunction of atomic conditions. In [19] rules and beliefs are defined as extended association rules of the form X → Y, where X is the conjunction of atomic conditions (an itemset) and Y is an atomic condition. The strength of the association rule is quantified by the following factors:

Confidence or *predictability*. A rule has confidence *c* if *c%* of the transactions in D that contain X also contain Y. A rule is said to hold on a dataset D if the confidence of the rule is greater than a user-specified threshold. *Support* or *prevalence*. The rule has support *s* in D if *s%* of the transactions in D contain both X and Y. *Expected predictability*. This is the frequency of occurrence of the item Y. So the difference between expected predictability and predictability (confidence) is a measure of the change in predictive power due to the presence of X [15].

In [18] unexpectedness is defined by starting with a set of beliefs that represent knowledge about the domain. A rule A → B is defined to be unexpected with respect to the belief X → Y on the database D if the following conditions hold:

- B and Y logically contradict each other (B AND Y |= FALSE);
- A AND X holds on a ''large'' subset of tuples in D;
- The rule A, X → B holds.

The discovered unexpected patterns are used in the refinement the initial rules or beliefs.

3.3 Refining Association Rules

The proposed refinement algorithm uses the best attributes for classification in order to find the more suitable rules for prediction.

The algorithm requires generating initial beliefs and unexpected patterns. In [18] the beliefs can either be obtained from the manager domain knowledge or induced from the data using machine learning methods. In our case, those beliefs were generated from the available data of projects by an association rule algorithm [15]. We found in an earlier work that the best attributes for classification give good results in the refinement of association rules in the area of projects management [17]. Taking as reference these previous results, we propose now a complete procedure for solving classification problems. We start with a set of beliefs which relate a single attribute with the class label. Then we search for unexpected patterns that could help us to increase the confidence or to solve ambiguities or inconsistencies between the rules representing the beliefs.

The steps to be taken in the refinement process [17] used for software size estimation are described below:

1. Obtain the best attributes for classification and create the sequence: seqA = $< \mathrm{A}_k >$, $k = 1 \ldots t$ (t: number of attributes). The attributes in the sequence are ordered from greater to lesser purity.
2. The values of the attribute A_k are represented as $\{\mathrm{V}_{k,l}\}$, $l = 1 \ldots m$ (m: number of different values).
3. Set $k = 1$ and establish the minimal *confidence* c_{min} and minimal *support* s_{min}.
4. Generate initial beliefs with confidence $c \geq c_{min}$ and support $s \geq s_{min}$.
5. Select beliefs with *confidence* near c_{min} or with conflicts between each other:

   Let $\mathrm{X}_i \rightarrow \mathrm{Y}_i$ and $\mathrm{X}_j \rightarrow \mathrm{Y}_j$ be two beliefs, R_i and R_j respectively. There is a conflict between R_i and R_j if $\mathrm{X}_i = \mathrm{X}_j$ and $\mathrm{Y}_i \neg= \mathrm{Y}_j$.
6. With the selected beliefs create the rule set setR = $\{\mathrm{R}_i\}$, $i = 1 \ldots n$ (n: number of selected beliefs)
7. For all beliefs $\mathrm{R}_i \in$ setR do:

   7.1. Use the values $\{\mathrm{V}_{k,l}\}$ of the attribute A_k for generating unexpected pattern fulfilling conditions of unexpectedness and *confidence* $\geq c_{min}$. The form of the patterns is: $\mathrm{V}_{k,l} \rightarrow \mathrm{B}$.

   7.2. Refine the beliefs by searching for rules R' like:

   $\mathrm{X}_i, \mathrm{V}_{k,l} \rightarrow \mathrm{B}$

   $\mathrm{X}_i, \neg\mathrm{V}_{k,l} \rightarrow \mathrm{Y}_i$

   7.3. Let setR' be the set of refined rules, then the beliefs refined in step 7.2 should be added to it:

   setR' = setR' $\cup$ $\{\mathrm{R'}_u\}$, $u = 1 \ldots f$ (f: number of refined rules obtained in the iteration i).
8. Set $k = k + 1$ and setR = setR'.
9. Repeat steps 7 and 8 until no more unexpected patterns can be found.

A significant advantage of the algorithm is the incremental knowledge discovery by means of the gradual generation of the refined rules. Good attributes in discriminating the classes are taken one by one in the iterative process, beginning with the best. This simplifies the selection of patterns, the refinement process and generates the best rules for class prediction. In other approaches a great set of rules is generated, which is pruned later without considering classification criteria.

3.4 Building the Classifier

The associative model obtained in the previous step is very suitable for classification purposes. Thus, it is used for making predictions. This model is composed by rules at different levels of refinement. The most refined rules are used in the first instance. If an observation to be classified has attributes coincident with the antecedent part of the rule, the label class assigned to the observation is the consequent part. If there are not rules that match the observation, previous level of refinement is used for searching suitable patterns.

The problem of finding more than one rule matching the observation is solved taking the more confident rule.

The simplicity of the models obtained with the refined rules lead to their efficient application for prediction. Another benefit is the lesser number of descriptive attributes needed with respect to traditional classification methods.

4 Experimental Study

Software size estimation is a critical issue in the project management area. Good early estimates are essential for a reliable prediction of project effort and cost as well as for an efficient planning and scheduling. Most of the software size estimation methods provide functional measures of the software size such as functions points [4], functions blocks [10] and object points [7]. Those functional variables should have a correspondence with the final product size expressed, for example, in lines of code. We have previously applied [16] several supervised data mining techniques in software size estimation taking as reference a component-based method [21] and a global method (Mark II) [20]. A correlation between the lines of code (LOC) and other attributes was found in the study. In this work we attempt to optimize the use of these atributes in the software size prediction.

4.1 Data Description

The data set used in the empirical study was obtained from experiments carried out by Dolado [9]. The data originate from academic projects in which students developed accounting information systems that have the characteristics of commercial products.

The information available was divided in two groups. One group containing global data from each project (42 records) and other group containing attributes from each component of the projects (1537 records). The code of the applications was classified into three types of components according to Verner and Tate [21].

The component attributes are: TYPECOMP (type of component, 1: menu, 2: input, 3: report/query), OPTMENU (number of choices, for menus), DATAELEMENT (number of data elements, only for inputs and reports/queries), RELATION (number of relations, only for inputs and reports/queries) and LOC (lines of code).

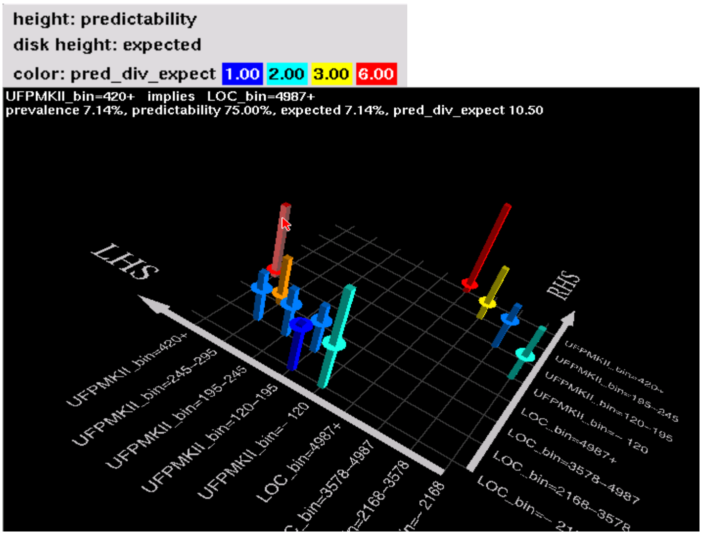
The project attributes described below are those used in Mark II method [20]: LOC (lines of code), NOC (number of components), NTRNSMKII (number of transactions MKII), INPTMKII (number of total inputs), ENTMKII (number of referenced entities), OUTMKII (number of total ouputs), UFPMKII (number of unadjusted functions points MKII).

New project attributes calculated from the module attributes of each project are: NOC-MENU (total number of menu components), NOC-INPUT (total number of input components), NOC-RQ (total number of report/query components), OPTMENU (total number of menu choices), DATAELEMENT (total number of data elements), RELATION (total number of relations).

4.2. Appliying the Refinement Algorithm

We aim to build a model that relates some project attributes that can be obtained early in the life cycle with final software size. Thus, LOC-bin (the intervals of lines of code) can be considered as the class label. The first step is to search the best attributes for discriminating the LOC-bin label by calculating their cumulative purity. The attributes found by means of the *Mineset* tool [15] were RELATION, NTRNSMKII, OPT_MENU and OUTMKII. These attributes were used in the refinement of the association rules.

We started with a set of beliefs which relate lines of code with UFPMKII due to MKII function points (UFPMKII) are good software size predictors [16]. The initial beliefs were generated by using the available data from 42 software projects. Figure 1 is a *Mineset* graphical representation of these rules on a grid landscape with left-hand side (LHS) items on one axis, and right-hand side (RHS) items on the other. Attributes of a rule (LHS → RHS) are displayed at the junction of its LHS and RHS item. The display includes bars, disk and colors whose meaning is given in the graph.


height: predictability
disk height: expected
color: pred_div_expect 1.00 2.00 3.00 6.00

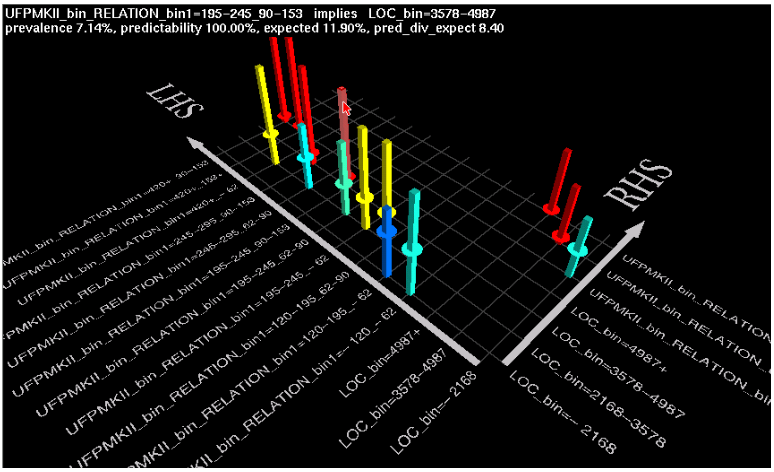
UFPMKII_bin=420+ implies LOC_bin=4987+
prevalence 7.14%, predictability 75.00%, expected 7.14%, pred_div_expect 10.50

Fig. 1. Rules representing the initial beliefs

Some of the initial rules presented conflicts between them and showed software size intervals with a weak correlation between lines of code and Mark II function points. Then we searched for unexpected patterns and applied the refinement process in order to solve these conflicts and obtain stronger rules that reinforce the relation between LOC and MKII FP by using the refered good attributes for classification that can be obtained early in the project life cycle. Figure 2 shows the refined rules obtained in the first iteration. It reveals that the refined rules have greater confidence than the initial ones. Most of them have 100% of confidence. After the entire process of incremental refinement we obtained a set of strong rules very useful for software size estimation.

5 Conclusions

In this work a classifier based on the predictive use of association rules is presented. We propose an algorithm for refining association rules based on incremental knowl-



UFPMKII_bin_RELATION_bin1=195-245_90-153 implies LOC_bin=3578-4987
prevalence 7.14%, predictability 100.00%, expected 11.90%, pred_div_expect 8.40

LHS
RHS
MKII_bin_RELATION_bin1=420+_90-153
UFPMKII_bin_RELATION_bin1=420+_153+
UFPMKII_bin_RELATION_bin1=420+_-62
PMKII_bin_RELATION_bin1=245-295_90-153
UFPMKII_bin_RELATION_bin1=245-295_62-90
UFPMKII_bin_RELATION_bin1=195-245_90-153
UFPMKII_bin_RELATION_bin1=195-245_62-90
UFPMKII_bin_RELATION_bin1=195-245_-62
UFPMKII_bin_RELATION_bin1=120-195_62-90
PMKII_bin_RELATION_bin1=120-195_-62
KII_bin_RELATION_bin1=-120_-62
LOC_bin=4987+
LOC_bin=3578-4987
LOC_bin=-2168
UFPMKII_bin_RELATION
UFPMKII_bin_RELATION_
UFPMKII_bin_RELATION_bi
LOC_bin=4987+
LOC_bin=3578-4987
LOC_bin=2168-3578
LOC_bin=-2168

Fig. 2. Refined beliefs in the first iteration

edge discovery that addresses the problem of producing a reduced number of useful rules in the software project management field. It provides managers with high confident rules for decision making without need of domain knowledge. The identification of weak rules representing beliefs and conflicts between them is the starting point to the iterative refinement process.

Our proposal evaluates the importance of the attributes on the classification. The algorithm produces the most useful rules for prediction, due to the fact that it uses the most important attributes in discriminating the different values of the class attribute, which is the target of the prediction. Other rule refinement methods do not consider this issue, thus they reduce the number of rules, but they do not obtain the most suitable rules for the desired objectives. With respect to the advantages of using association instead of classification, the first is the lower number of attributes required and the second, the greater efficiency of the association algorithms.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Database Mining: A performance Perspective. IEEE Trans. Knowledge and Data Engineering, vol. 5, 6 (1993b) 914-925.
2. Agrawal, R., Imielinski, T., Swami, A.: Mining associations between sets of items in large databases. Proc. of ACM SIGMOD Int. Conference on Management of Data, Washinton, D.C. (1993a) 207-216.
3. Agrawal, R., Srikant, R.: Fast Algorithms for mining association rules in large databases. Proc. of 20th Int. Conference on Very Large Databases, Santiago de Chile (1994) 487-489.
4. Albrecht, A.J. Measuring Application Development: Proc. IBM Applications Development Joint SHARE/GUIDE Symposium, Monterey, CA, (1979) 83-92.
5. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense database, Proc. 15th Int. Conference on Data Engineering (1999) 188-197.

6. Bayardo, R., Agrawal, R.: Constraint-based rule mining in large, dense database. Proc. Mining the most interesting rules. Proc. ACM SIGKDD Int. Conf. Knowledge Discovery in Databases, ACM Press, NY (1999) 145-154.
7. Boehm, B.W., Clark; B. Horowitz E. et al., Cost Models for Future Life Cycle Processes: COCOMO 2.0: Annals *Software Engineering* 1, No. 1 (1995) 1-24.
8. Cabena, P., Hadjinian, P., Stadler, R. Verhees, J. and Zanasi, A.: Discovering Data Mining. from concept to implementation, (Prentice Hall, 1998).
9. Dolado, J.J. A Validation of the Ccomponent-Based Method for Software Size Estimation: IEEE Transactions on Software Engineering 26, 10 (2000) 1006-1021.
10. Hall, B., Orr, G., Reeves, T.E. A Technique for Function Block Counting: The Journal of System and Software, 57 (2001), 217-220.
11. Hu, Y.C., Chen, R.S., Tzeng, G.H.: Mining fuzzy associative rules for classifications problems. Computers and Industrial Engineering,
12. Li, J., Shen, H., Topor, R.: Mining the smallest association rule set for predictions. Proc. IEEE International Conference on Data Mining (ICDM'01) (2001).
13. Liu, B., Hsu, W. Ma, Y.: Integration Classification and Association Rule Mining. Proc. 4th Int. Conference on Knowledge Discovery and Data Mining, (1998) 80-86.
14. Liu, B., Hsu, W., Chen, S., Ma, Y.: Analyzing the subjective Interestingness of Association Rules. IEEE Intelligent Systems, september/October (2000) 47-55.
15. Mineset user's guide, v. 007-3214-004, 5/98. (Silicon Graphics, 1998).
16. Moreno, M.N., Miguel, L.A., García, F.J., Polo, M.J.: Data mining approaches for early software size estimation. Proc. 3rd ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'02), 361-368, Madrid, Spain (2002).
17. Moreno, M.N., Miguel, L.A., García, F.J., Polo, M.J.: Building knowledge discovery-driven models for decision support in project management. Dec. Support Syst., (in press).
18. Padmanabhan, B., Tuzhilin, A.: Knowledge refinement based on the discovery of unexpected patterns in data mining. Decision Support Systems 27 (1999) 303– 318.
19. Padmanabhan, B., Tuzhilin, A.: Unexpectedness as a measure of interestingness in knowledge discovery. Decision Support Systems 33 (2002) 309– 321.
20. Symons, C.R. Software Sizing and Estimating MKII FPA (John Wiley and Sons, 1991).
21. Verner, J. and Tate, G. A Software Size Model: IEEE Transaction of Software Engineering, 18 No. 4 (1992): 265-278.
22. Wang, Y., Wong, A.K.C.: From Association to Classification: Inference Using Weight of Evidence. IEEE Transactions on Knowledge and Data Engineering, 15 (2003) 764-767.

PROWL: An Efficient Frequent Continuity Mining Algorithm on Event Sequences

Kuo-Yu Huang, Chia-Hui Chang, and Kuo-Zui Lin

Department of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320
{want,kuozui}@db.csie.ncu.edu.tw, chia@csie.ncu.edu.tw

Abstract. Mining association rule in event sequences is an important data mining problem with many applications. Most of previous studies on association rules are on mining intra-transaction association, which consider only relationship among the item in the same transaction. However, intra-transaction association rules are not a suitable for trend prediction. Therefore, inter-transaction association is introduced, which consider the relationship among itemset of multiple time instants. In this paper, we present PROWL, an efficient algorithm for mining inter-transaction rules. By using projected window method and depth first enumeration approach, we can discover all frequent patterns quickly. Finally, an extensive experimental evaluation on a number of real and synthetic database shows that PROWL significantly outperforms previous method.

1 Introduction

Mining frequent patterns in event sequences is a fundamental problem in data mining areas. There are many emerging applications in pattern mining, including stock market price movement, telecommunication network fault analysis and web usage recommendation. Therefore, there are various directions in pattern mining, such as frequent itemset, sequential pattern, frequent episode [2], periodic pattern [1] etc. Some of them are on mining intra-transaction association rules like “The price of 3Com(COMS) and Cisco(CSCO) always go up together on the same day with 70% confidence”. Such information can be used for sale purposes such as decision making because the co-occurrence patterns of records can be used to infer another record. Despite the above rule reflects some relationship among the prices, it is not a suitable and definite rule for trend prediction. The investors may be more interested in a rule like “When the price of COMS goes up, the price of CSCO will go up with 60% probability three days later.” Therefore, we need a pattern that shows the relationships between items in different transaction records. In order to classify these two rules explicitly, we call the former rule as **intra-transaction associations**, the latter rule as **inter-transaction associations**.

In addition to inter-transaction association rules, there are two other kinds of association mining from different transaction records, frequent episodes and periodic patterns. An episode is defined to be a collection of events in a specific

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 351–360, 2004.

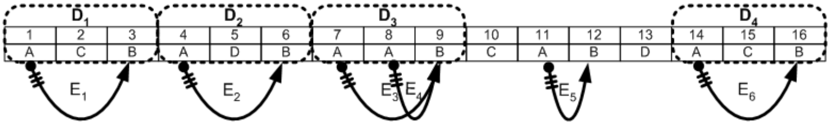


Fig. 1. Event Sequence S.

window interval that occur relatively close to each other in a given partial order [2]. Take Figure 1 as an example, there are six matches of episode $\{A, B\}$, from E_1 to E_6, in event sequence S. Mannila et al. proposed an algorithm, WINEPI [2], for finding all episodes that are frequent enough. This algorithm was an Apriori-like algorithm based on the "anti-monotone" property of episodes. They also presented MINEPI, an alternative approach, to the discovery of frequent episodes based on minimal occurrences of episodes. Note that an episode considers only the partial order relation, instead of the actual positions, of events in a window.

Unlike episodes, a periodic pattern [1] considered not only the order of events but also the exact positions of events. To form periodicity, a list of k disjoint matches is required to form a contiguous subsequence with k satisfying some predefined min_rep threshold. As illustrated in Figure 1, pattern $\{A, *, B\}$ is a periodic pattern that matches D_1, D_2, and D_3, three continuous while disjoint matches, where A (resp. B) occurs at the first (resp. *third*) position of each match. The symbol "*" (the "don't care" position in a pattern) is introduced to allow partial periodicity. Sometimes, we use a 4-tuple (P, l, *rep*, *start*) to denote a segment of pattern P with period l starting from position *start* for *rep* times. In this case, the segment can be represented by ($\{A, *, B\}$, 3, 3, 1).

The concept and definition of inter-transaction association rules are first introduced in [4] by Tung et al. A frequent pattern in inter-transaction association rule is similar to a periodic pattern, but without the constraint on the contiguous and disjoint matches. To distinguish the frequent patterns in inter-transaction association rules from the frequent patterns in intra-transaction association rules, we use the term "**frequent continuities**" for the frequent patterns in inter-transaction association rules. Let us return to our previous example in Figure 1. $\{A, *, B\}$ is a continuity with four matches D_1, D_2, D_3, and D_4.

Inter-transaction association rules can be mined by the FITI algorithm proposed in [3]. FITI consists of two phases: intra-transaction and inter-transaction itemsets mining. Similar to Apriori-like algorithms, FITI could generate a huge number of candidates and require several scans over the whole data sequence to check which candidates are frequent. To illustrate, if there are I frequent 1-patterns, the FITI algorithm will generate approximately I^W 2-pattern candidates for window size W. Experimentally, the bottleneck of the FITI method comes from its step-wise candidate pattern generation and test. Therefore, it is desirable to develop an algorithm which avoid candidate generation and reduce the search space. With this motivation, we devised a two-phase algorithm, PROWL, for mining frequent continuities from event sequences. We introduce a projected window mechanism and employ depth first enumeration approach to discover all frequent continuities.

The remaining parts of the paper are organized as follows. We define the problem of frequent continuity mining from event sequence in Section 2. Section 3 presents our algorithm for mining frequent continuity in event sequence. Experiments on both synthetic and real world data are reported in Section 4. Finally, conclusion is made in Section 5.

2 Problem Definition

In this section, we define the problem of frequent continuity mining. The problem definition is similar to [3] but is restricted to a sequence of events. Let $E = (e_1, e_2, \ldots, e_n)$ be a set of literals, called events. A event sequence S is a set of time records where each time record is a tuple (tid, x_i) for time instant tid and event x_i ($x_i \in E$). A sequence stored in form of (tid, x_i) is called horizontal format (e.g. Figure 1).

Definition 1. *A continuity pattern (or a continuity in short) with window W is a nonempty sequence $P = (p_1, p_2, \ldots, p_W)$ where p_1 is an event and others are either an event or *, i.e. $p_j \in E$ or $\{*\}$ for $2 \leq j \leq W$.*

The symbol "*" is introduced to allow mismatching (the "don't care" position in a pattern). Since a continuity pattern can start anywhere in a sequence, we only need to consider patterns that start with a non-"*" symbol. A continuity P is called an i-continuity or has length i if exactly i positions in P contain event. For example, $\{A, *, *\}$ is a 1-continuity; $\{A, *, C\}$ is a 2-continuity which has length 2.

Definition 2. *Given a continuity pattern $P = (p_1, p_2, \ldots, p_W)$ and a subsequence of W slots $D = (d_1, d_2, \ldots, d_W)$ in S, we say that P* **matches** *D (or D supports P) if and only if, for each position j ($1 \leq j \leq l$), either p_j = * or $p_j = d_j$ is true. D is also called a match of P.*

In general, given a sequence of events and a pattern P, multiple matches of P may exist. In Figure 1, $D_1, D_2, \ldots, D_4$ are four matches of pattern $\{A, *, B\}$.

Definition 3. *An* **inter-transaction association rule** *is an implication of the form $X \Rightarrow Y$, where*

1. *X, Y are continuity patterns with window w_1 and w_2, respectively.*
2. *The concatenation $X \cdot Y$ [1] is a continuity pattern with window $w_1 + w_2$.*

Similar to the studies in mining intra-transaction rules, we also introduce two measures of inter-transaction association rules: support and confidence.

[1] The **concatenation** of two continuity patterns $P = (p_1, \ldots, p_{w_1})$ and $Q = (q_1, \ldots, q_{w_2})$ is defined as $P \cdot Q = (p_1, \ldots, p_{w_1}, q_1, \ldots, q_{w_2})$.

Definition 4. *Let $|S|$ be the number of transactions in the event sequence S. Let $Sup(X \cdot Y)$ be the number of matches with respect to continuity $X \cdot Y$ and $Sup(X)$ be the number of matches with respect to continuity X. Then, the support and confidence of an inter-transaction association rule $X \Rightarrow Y$ are defined as*

$$support = \frac{Sup(X \cdot Y)}{|S|},\ confidence = \frac{Sup(X \cdot Y)}{Sup(X)}. \tag{1}$$

The problem is formulated as follows: given a minimum support level $minsup$ and a minimum confidence level $minconf$, our task is to mine the complete set of inter-transaction association rules from an event sequence with support greater than $minsup$ and confidence greater than $minconf$. We illustrate the concepts with an example. Let $minsup$ and $minconf$ be 25% and 60% respectively. An example of an inter-transaction association rule from the event sequence in Figure 1 will be: $\{A, *\} \Rightarrow \{B\}$. This rule (Event B occurs two slots later after event A.) holds in the sequence S with support 25% and confidence 67%.

3 The PROWL Algorithm

In this section, we explore methods for mining frequent continuities in an event sequence. Our algorithm, PROWL (PROjected Window Lists), uses a recursive depth first enumeration strategy to discover all frequent continuities from an event sequence. To avoid candidate generation, we use a vertical data format together with a horizontal format for efficient continuity generation. Table 1 shows the vertical format for the event sequence S in Figure 1, where a time list is maintained for each event. A time list of an event records the time slots where the event occurs in the sequence.

Definition 5. *Given a sequence of events S and a continuity P with window W, let I_i denotes a subsequence of W time slots $I_i = (S[s_i], S[s_i{+}1], \ldots, S[s_i{+}W{-}1]$ in S that supports P. Suppose there are k matches of P in S. The time list of P is defined as $P.list = \{s_1 + W - 1, s_2 + W - 1, \ldots, s_k + W - 1\}$.*

By definition, each event is itself a continuity with window 1. The time list for a 1-continuity pattern is consistent with the time list for an event. Now, we define the projected window list of a continuity from its time list as follows.

Definition 6. *Projected Window List (PWL): Given a time list of a continuity P, $P.list = \{o_1, o_2, \ldots, o_k\}$ in the event sequence S, the projected window list of P is defined as $P.PWL = \{w_1, w_2, \ldots, w_k\}$, $w_i = o_i + 1$ for $1 \leq i \leq k$. Note that a time slot w_i is removed from the projected list if w_i is greater than $|S|$, i.e. $w_i \leq |S|$ for all i. If an event X is frequent in the projected window list of pattern P, we refer to the concatenation $P \cdot X$ as an extension of P.*

The PROWL algorithm mines frequent continuities by the following phases.

- Initial phase: The sequence S is first read into memory and scanned once. For each event, a time list is maintained to record its occurring time slot. The number of occurrences is also accumulated for frequent event filtering.

Table 1. Vertical format of the event sequence S in Figure 1.

Event	Time List	Projected Window List
A	1, 4, 7, 8, 11, 14	2, 5, 8, 9, 12, 15
B	3, 6, 9, 12, 16	4, 7, 10, 13
C	2, 10, 15	3, 11, 16
D	5, 13	6, 14

- Recursive phase: For each frequent 1-continuity, we calculate a projected window list (PWL) from the pattern's time list and find frequent events in its PWL. We then output the frequent continuity formed by current pattern and the frequent events in the PWL. For each extension pattern, the process is applied recursively to find all frequent continuities until the projected window list becomes empty or the window of a continuity is greater than the maximum window.

3.1 An Example

The PROWL algorithm can be best understood by an illustrative example described below and its corresponding flowchart is depicted in Figure 2.

Example 1. Given $Sup = 3$ and $maxwin = 4$, the frequent events for Table 1 include A, B and C. For frequent 1-continuity $\{A\}$, the projected window list is $P_A.PWL = \{2, 5, 8, 9, 12, 15\}$. Note that $P_A.PWL$ is also the time list of continuity $\{A, *\}$. By examining the time slots of $P_A.PWL$ in Figure 1, all the continuities with window 2 having prefix $\{A\}$ can be generated by concatenating $\{A\}$ with a frequent event in $P_A.PWL$ or the don't care symbol. For instance, the corresponding events for the time slots in $P_A.PWL$ are C, D, A, B, B, C, respectively. For each event E_i, a time list is constructed accordingly. Since D is not frequent in S, it is simply ignored. Furthermore, as there are no frequent events in $P_A.PWL$, the only frequent continuity generated from $\{A\}$ is $\{A, *\}$ (with 6 matches).

Recursively, we apply the above process to continuity $\{A, *\}$. The projected window list of $\{A, *\}$ is $P_{\{A,*\}}.PWL = \{3, 6, 9, 10, 13, 16\}$. In this layer, we find a frequent event B in time records of $P_{\{A,*\}}.PWL$. Thus, two continuities: $\{A, *, B\}$ and $\{A, *, *\}$ are generated with time list $P_{\{A,*,B\}}.list = \{3, 6, 9, 16\}$ and $P_{\{A,*,*\}}.list = \{3, 6, 9, 10, 13, 16\}$, respectively. The extensions of the continuities can be mined by applying the above process respectively to each continuity as shown in Figure 2. Note that the projected window list of $\{A, *, B\}$ is $\{4, 7, 10\}$, because time record 17(16+1) is greater than sequence length 16. Similarly, we can find all frequent continuities having prefix $\{B\}$, respectively, by constructing $P_B.PWL$ and mining them respectively. The set of frequent continuities is the collection of patterns found in the above recursive mining process.

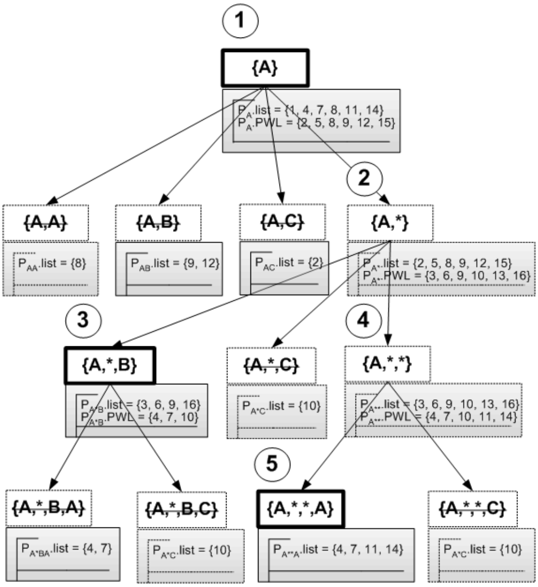

1
{A}
P_{A}.list = {1, 4, 7, 8, 11, 14}
P_{A}.PWL = {2, 5, 8, 9, 12, 15}
2
~~{A,A}~~
P_{AA}.list = {8}
~~{A,B}~~
P_{AB}.list = {9, 12}
~~{A,C}~~
P_{AC}.list = {2}
{A,*}
P_{A*}.list = {2, 5, 8, 9, 12, 15}
P_{A*}.PWL = {3, 6, 9, 10, 13, 16}
3
{A,*,B}
P_{A*B}.list = {3, 6, 9, 16}
P_{A*B}.PWL = {4, 7, 10}
~~{A,*,C}~~
P_{A*C}.list = {10}
4
{A,*,*}
P_{A**}.list = {3, 6, 9, 10, 13, 16}
P_{A**}.PWL = {4, 7, 10, 11, 14}
5
~~{A,*,B,A}~~
P_{A*BA}.list = {4, 7}
~~{A,*,B,C}~~
P_{A*C}.list = {10}
{A,*,*,A}
P_{A**A}.list = {4, 7, 11, 14}
~~{A,*,*,C}~~
P_{A*C}.list = {10}

Fig. 2. The overall process of PROWL for Figure 1.

3.2 The PROWL Algorithm

The main idea of PROWL is to utilize the memory for both the event sequence and the indices in the mining process. The event sequence S is read once from disk and each event is associated with a time list containing indices to the time slots where it occurs. Based on the concepts of projected window list and "anti-monotone" property, we can generate all frequent continuities by depth-first enumeration. PROWL discovers all frequent continuities recursively by searching the time slots in the projected window lists immediately. Figure 3 outlines the proposed PROWL algorithm.

In the first phase, we scan event sequence S once and transform S into vertical format (Step 1$\sim$2 in the *Prowl*). To reduce the number of events for enumeration, the number of occurrences for each event is accumulated during sequence reading and non-frequent events are masked (Step 3$\sim$5 of *Prowl*). Therefore, non-frequent events will not be enumerated in the recursive phase. In the second phase, the procedure *Project* is applied recursively to enumerate all continuities with known frequent continuities as their prefixes. For each frequent continuity P, we transform its time list into a projected window list and examing the time slots of it's projected window in S (Step 4$\sim$8 in the *Project*). Note that the horizontal format of sequence S is maintained in main memory for fast access to the event at each time slot $o_i \in P.PWL$. The function $TempEvent(e).insert(o_i)$ insert time slot o_i into the time list of event e. If an event E_i is frequent, the continuity $P \cdot E_i$ is output (Step 10$\sim$13 of *Project*). The recursive call stops when the layer is greater than *maxwin* (Step 1 of *Project*).

```
Given event sequence $S$, $Sup$, $maxwin$;

Procedure of **Prowl()**
for $i = 1$ to $|S|$ do
    $EventSet[S[i]].insert(i)$;
for each event $E_i \in EventSet$ do
    if ($EventSet[E_i].size < Sup$) then
        for each time instant $o_i \in EventSet[E_i]$ do
            $S[o_i] = *$;
for each event $E_i \in EventSet$ do
    if ($EventSet[E_i].size >= Sup$) then
        Pattern[0] = $E_i$;
        for $j = 1$ to $maxwin - 1$ do
            Pattern[j] = *;
        Project($EventSet[E_i], Pattern, 1$);
end

Subprocedure of **Project($TimeList$, $Pattern$, $Layer$)**
begin if ($Layer <= maxwin$) then
    $PWL = NULL$;
    $TempEvent = NULL$;
    for each time instant $T_i \in TimeList$ do
        if ($T_i < |S|$) then
            $PWL.insert(T_i + 1)$;
    for each time instant $o_i \in PWL$ do
        if $S[o_i] \neq *$ then
            $TempEvent[S[o_i]].insert(o_i)$;
    begin for each event $E_i \in TempEvent$ do
        if ($TempEvent[E_i].size >= Sup$) then
            Pattern[Layer] = $E_i$;
            Project($TempEvent[E_i], Pattern, Layer + 1$);
            Output Pattern;
        Pattern[Layer] = *;
        Project($TempEvent[E_i], Pattern, Layer + 1$);
    end
end
```

Fig. 3. PROWL: Frequent Continuity mining algorithm.

In contrast with Apriori-like algorithms, we only generate longer pattern by shorter ones. It does not generate any candidate patterns for checking. Besides, the projected window lists are actually smaller than the original ones. The situation will be illustrated in the scale-up experiments discussed later.

4 Experimental Result

In this section, we report a performance study of the algorithm proposed in this paper and applications of frequent continuity in real world data. We first investigate the performance of PROWL and compare the result with the FITI algorithm proposed in [3] using synthetic data. The PROWL algorithm is then applied to real world data for frequent continuity mining.

4.1 Synthetic Data

To obtain reliable experimental results, the method to generate synthetic data we employed in this study is similar to the ones used in prior works [1]. We use a synthetically generated event sequence consisting of $|N|$ distinct symbols and $|S|$ events. A set of candidate continuities C is generated as follows. First, we decide the window length of a continuity from a geometrical distribution with mean W. Then L $(1 < L < W)$ positions are chosen randomly for non-empty events. The number of occurrences of a continuity follows a normal distribution with mean *Avg_Sup*. The continuity is then inserted into the sequence *Avg_Sup* times with gap between W to $2W$. A total of $|C|$ complex patterns are generated. After all candidate patterns are generated, events are picked at random from the symbol set N for empty time slots. The default parameter is S20K-N1K-C10-L4-W10-Avg_Sup=0.5. The experiments are conducted on a computer with a CPU clock rate of 1G MHz and 1.5G MB of main memory, the program is written by visual C++ in windows 2000 platform.

4.2 Comparison of PROWL with FITI

We reported experimental results on the default data set. For comparison with PROWL, we implement FITI algorithm which is proposed in [3]. Both PROWL and FITI algorithms show linear scalability with the size of a sequence from 10K to 80K as shown in Figure 4(a). However, PROWL is much more scalable than FITI. As the size of a sequence grows up, the difference between the two methods becomes larger and larger.

Figure 4(b) shows that the running time of PROWL and FITI with variable number of candidates. It shows that the running time of PROWL increases smoothly while FITI increases exponentially as the number of candidates increases. To test the scalability with the pattern length, we also execute an experiment with varying pattern length. The results are presented in Figure 4(c). Overall, PROWL is about an order of magnitude faster than FITI. This is because the large number of candidates that need to be generated and tested in FITI as the pattern length grows. Figure 4(d) shows the scalability of the algorithm over the varying window size. Note that the parameter *maxwin* is set to the window size W of dataset. As the window size becomes larger, more candidates need to be generated in FITI. On the contrary, PROWL doesn't generate any candidate for checking. Thus, for a fixed pattern length, the running time of PROWL is smooth with the varying window size.

4.3 Real World Data

We also run PROWL on a variety of different real world data sets to get a better view of the usefulness of frequent continuities in event sequences. The data sets

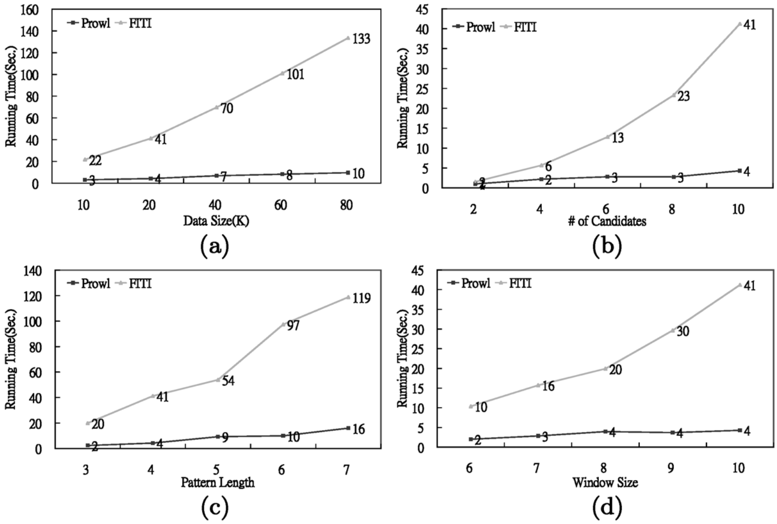


Fig. 4. Scale-up performances with (a) sequence size (b) number of candidates (c) pattern length (d) window size.

Table 2. Dataset Characteristics.

Data Set	Events	Event Types	Support	Maximum Window	# of continuity
UNIX User0	8974	197	100	10	341
UNIX User1	19881	288	200	10	711
UNIX User2	18738	310	200	10	2183
Protein Sequence	296	22	10	12	10165

are taken from the UNIX user usage logs in UCI Machine Learning Database Repository. The UNIX user usage logs contains 9 subsets of sanitized user data drawn from the command histories of 8 UNIX computer users at Purdue over the course of up to 2 years. We show only the data of User0, User1 and User2 due to space limitation. The description of the data used, the parameters, and the number of frequent continuities discovered are presented in Table 2.

Finally, we apply PROWL to protein sequences to discover tandem repeats, which is an important problem in bioinformatics. We used data in the PROSITE database of the ExPASy Molecular Biology Server (http://www.expasy.org/). We selected a protein sequence P13813 (110K_PLAKN) with a known tandem repeats "{E,E,T,Q,K,T,V,E,P,E,Q,T}". As expected, several continuities which are related to the known tandem repeat are discovered. It is indicated that our algorithm can be used in protein sequence mining.

5 Conclusion

In this paper, we proposed an algorithm, PROWL, for mining frequent continuities in event sequence. The idea is to utilize memory for storing event sequence in both horizontal and vertical data format. PROWL generates frequent continuities by applying a projected window method recursively. The experiments show that the method is efficient and flexible. We compared PROWL with previous research, and the result reported that Prowl outperformed than FITI. We have applied the method in the analysis of the UNIX user usage log and mining tandem repeats in protein sequences. There are several directions for future work. The first direction is to develop techniques for mining inter-transaction association rules from a sequence of eventsets, or a transaction database. The second direction is to develop techniques for managing the large number of frequent continuities discovered, by introducing the concept of maximal or closed continuities. More research will be reported in the near future.

Acknowledgements

This work is sponsored by Ministry of Economic Affairs, Taiwan under grant 93-EC-17-A-02-S1-029.

References

1. K.Y. Huang and C.H. Chang. Asynchronous periodic patterns mining in temporal databases. In *Proc. of the IASTED International Conference on Databases and Applications (DBA)*, pages 43–48, 2004.
2. Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In *Proc. of the First International Conference on Knowledge Discovery and Data Mining*, pages 210–215, 1995.
3. A. K. H. Tung, J. Han H. Lu, and L. Feng. Breaking the barrier of transactions: Mining inter-transaction association rules. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 297–301, 1999.
4. A. K. H. Tung, J. Han H. Lu, and L. Feng. Efficient mining of intertransaction association rules. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):43–56, 2003.

Algorithms for Discovery of Frequent Superset, Rather than Frequent Subset

Zhung-Xun Liao and Man-Kwan Shan

Department of Computer Science, National Cheng Chi University, Taipei, Taiwan, ROC
{g9113,mkshan}@cs.nccu.edu.tw

Abstract. In this paper, we propose a novel mining task: mining frequent superset from the database of itemsets that is useful in bioinformatics, e-learning systems, jobshop scheduling, and so on. A frequent superset means that it contains more transactions than minimum support threshold. Intuitively, according to the Apriori algorithm, the level-wise discovering starts from 1-itemset, 2-itemset, and so forth. However, such steps cannot utilize the property of Apriori to reduce search space, because if an itemset is not frequent, its superset maybe frequent. In order to solve this problem, we propose three methods. The first is the Apriori-based approach, called Apriori-C. The second is Eclat-based approach, called Eclat-C, which is depth-first approach. The last is the proposed data complement technique (DCT) that we utilize original frequent itemset mining approach to mine frequent superset. The experiment study compares the performance of the proposed three methods by considering the effect of the number of transactions, the average length of transactions, the number of different items, and minimum support.

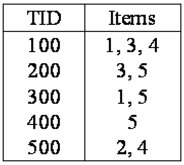
1 Introduction

Frequent itemset mining algorithms have been investigated for a long time, such as Apriori[1], FP-growth[2], Eclat[3], Tree-Projection[4], H-Mine[5], DHP[6], and so on. The essence of frequent itemset mining is to discover frequent subsets from a set of itemset. In our study, we propose a novel mining task to discover frequent superset. We want to find patterns that are superset of a certain number of transactions. For the clarity of description, we use the term "frequent subset" to denote the traditional frequent itemset, and "frequent superset" to this new problem.

Definition 1. Let $D=\{T_1, T_2, \ldots, T_k\}$ be a transaction database with k transactions, X be an itemset, and minimum support is h. If $T_{i_j} \subseteq X$, where $1 \le i_j \le k$, for each j=1, 2, …, m, m is referred to as the support of itemset X. If $m \ge h$, we say that X is a frequent superset.

Example 1. In Fig. 1, the support of itemset {1, 3, 5} is 3, because there are three transactions, {3, 5}, {1, 5}, and {5} which are subsets of {1, 3, 5}. Given the value of minimum support 3, the frequent supersets are {1, 2, 3, 4, 5}, {1, 2, 3, 5}, {1, 2, 4, 5}, {1, 3, 4, 5}, {2, 3, 4, 5}, and {1, 3, 5}.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 361–370, 2004.



TID	Items
100	1, 3, 4
200	3, 5
300	1, 5
400	5
500	2, 4

Fig. 1. An Example

Intuitively, the basic approach to solve the problem of frequent superset mining is just a little modification of the original Apriori-based algorithm for frequent subset mining. The original Apriori property states that all nonempty subsets of a frequent subset must also be frequent. In other words, if an itemset is not frequent, then its superset is not a frequent itemset. Therefore, in the Apriori-based algorithm, the level-wise generation of frequent itemsets is employed.

However, the Apriori property cannot be directly applied to the problem of frequent superset mining. Because if an itemset *X* is not a frequent superset, it is possible that the itemset which is a superset of *X* is frequent. Therefore, we can not use k-itemset to prune and explore (k+1)-itemset in the same way as the original Apriori-based algorithm does.

Certainly, we can utilize the property that if an itemset *X* is not a frequent superset, then the itemset which is a subset of *X* is not frequent either. Based on this property, we can develop the algorithm that (k+1)-itemset are used to prune and explore k-itemset. However, this algorithm starting the exploration from a long pattern is inefficient. This method is used as the baseline approach for performance comparison.

In order to solve this problem, we design three algorithms to discover such frequent supersets. These are Apriori-based, Eclat-based, and data complement technique (DCT), presented in section 2.1, 2.2, and 2.3, respectively.

In our experiments, we assess the performance of these algorithms based on the following four parameters, minimum support, number of transactions, number of items, and the average size of transactions.

2 Discovery of Frequent Superset

In this study, we design three algorithms, Apriori-C, Eclat-C, and data complement technique (DCT), which are all based on the set complement view of point. We do not find frequent supersets directly, but the complement of frequent supersets, and then obtain frequent supersets by taking the complement of those patterns. First of all, we give some related definitions.

Definition 2. Let $I=\{i_1, i_2, \ldots, i_n\}$ denote the set of all items, $X=\{j_1, j_2, \ldots, j_m\}$ be an itemset, where $j_p \in I$, for all $p=1\sim m$. An itemset $X'=\{i_q \mid \forall\, i_q \notin X,\ q=1\sim n\}$ is defined as the complement of *X*, denoted as *X'*. Moreover, given a database $D=\{T_1, T_2, \ldots, T_k\}$, the complement of *D* is defined as $D'=\{\ T_1', T_2', \ldots, T_k'\ \}$.

Example 2. Let I={1, 2, 3}. Suppose we have a database D={{1, 3}, {2, 3}}, and the complement of D, D'={{2}, {1}}.

2.1 Algorithm Apriori-C

This algorithm is an Apriori-based method, which is the level-wise discovery process starting from 1-itemset, 2-itemset, and so on. Each level is also a join-and-prune process. We first define some terms and properties before describing the algorithm.

Definition 3. Consider two itemsets X_1 and X_2, if $\forall\ Y \in \mathrm{P}(X_1)$, the powerset of itemset X_1, $Y \not\subset X_2$, $Y \neq \phi$, we say that X_1 **complement-contains** X_2, denote as $X_1 \supseteq_c X_2$.

Example 3. Let X_1={1, 3} and X_2={2, 3}. Because {1}, {3}, and {1, 3} are not contained in X_2, we say that X_1 complement-contains X_2, represented as $X_1 \supseteq_c X_2$.

Definition 4. Let $D=\{T_1, T_2, \ldots, T_k\}$ be a transaction database with k transactions, and h be the minimum support. If an itemset X, $X \supseteq_c T_{i_j}$, where $1 \leq i_j \leq k$, for each j=1, 2, …, m, we say that m is the **complement-support** of X. If $m \geq h$, we define X to be a **complement-frequent superset** of D.

Example 4. Consider the database in Example 1, suppose that the minimum support is 3, the complement-frequent supersets are {1}, {2}, {3}, {4}, and {2, 4}, and their complement-support are 3, 4, 3, 3, and 3, respectively.

Lemma 1. Consider two itemsets X_1 and X_2. If $X_1 \supseteq_c X_2$, then X_1' is a superset of X_2. That is, $X_1' \supseteq X_2$.

Proof. Suppose $X_1=\{a_1, a_2, \ldots, a_i\}$, $X_2=\{b_1, b_2, \ldots, b_j\}$, and $X_1 \cap X_2 = \{c_1, c_2, \ldots, c_k\}$, where $c_n \in X_1$ and $c_n \in X_2$, for all n=1~k. Because $X_1 \supseteq_c X_2$, we can get $\forall\ Y \in \mathrm{P}(X_1)$, $Y \not\subset X_2$. But the itemset that containing c_n , n=1~k, must be a subset of X_2 and must belong to $\mathrm{P}(X_1)$. It is known that any c_n, n=1~k does not exist, that is $X_1 \cap X_2 = \phi$, I-$X_1 \supseteq X_2$, and $X_1' \supseteq X_2$.

Example 5. Suppose we have a transaction T={1, 5} and two itemsets X_1={2, 3} and X_2={1, 3}. Let I={1, 2, 3, 4, 5}, we can obtain the powerset of X_1, $\mathrm{P}(X_1)$={{2}, {3}, {2, 3}}, and X_2, $\mathrm{P}(X_2)$={{1}, {3}, {1, 3}}. Because {1} $\subseteq$ {1, 5}, X_2'={2, 4, 5} is not a frequent superset of T, while X_1'={1, 4, 5} is a frequent superset of T.

There are two steps consisted in each passes of proposed algorithm Apriori-C. This algorithm adopts the Apriori property based on the following observation: if an itemset X is not a complement-frequent superset, the superset of X is not a complement-frequent superset either. At the kth pass, first, the candidates with size of k are generated from complement-frequent superset with size of k-1 using the original aprior-gen methods. Next, scan database once and count the complement-support of candidates to determine the complement-frequent supersets. Fig. 2 gives the algorithm of Apriori-C. In order to count complement-support efficiently, we build a prefix trie, which is modified from [7] to record the candidates. Fig. 3 shows the algorithm for counting complement support using the prefix trie for each transaction. After the execution of this algorithm, we have obtained the complement-frequent superset, so we must take the complement of the complement-frequent superset to get the frequent superset.

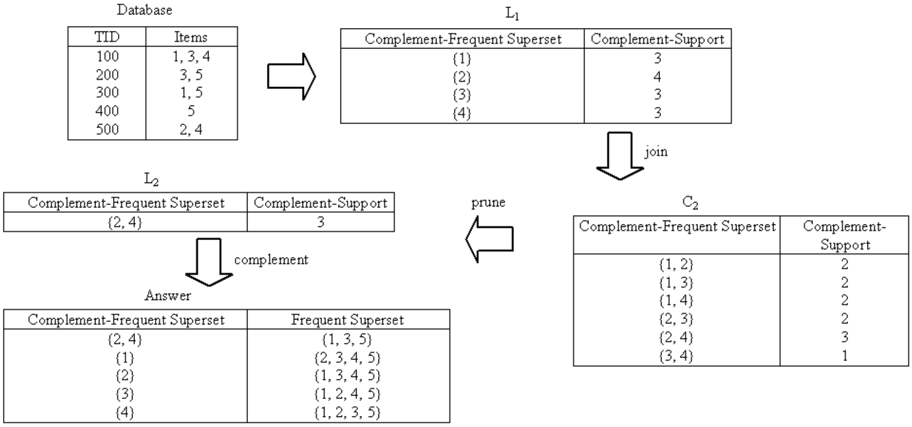
```
$L_1$ = {frequent superset with size of 1}
$L' = I - L_1$;
for ( $k = 2$ ; $L \neq \phi$ ; $k$++) do begin
  $C_k$ = apriori-gen($L'$);
  forall transactions $t \in D$ do
    compSupp(root,$t$,$0$); //count complement support
  $L_k = \{c \in C_k | c$.count $\geq$ minsup$\}$
  $L' = C_k - L_k$
end
Answer = $\bigcup_k L_k$ ;
```

Fig. 2. Apriori-C algorithm

```
Procedure compSupp (Node:$n$,$Transaction$:$t$,$Int$:$i$)
 if($n$ is internal node) then begin
  forall children $c$ of $n$ do begin
   if($c < t[i]$) then begin
    call compSupp($i$,$c$);
   else if($c > t[i]$) then
    if(exist $t[i+1]$) then
     $i$++;
    else
     forall leaf node $f$ large then $c$ do
      $f$.counter++;
   else
    if( exist $t[i+1]$) then
     $i$++;
   end
  end
else
 $n$.counter++;
end
```

Fig. 3. An algorithm for counting complement support

Example 6. Consider the database in Fig. 4 and assume that minimum support is 3 transactions. At the first pass, we determine the complement-frequent superset with size of 1 showed in L_1. At the second pass, there are 6 complement-supersets generated, but only {2, 4} is frequent. There are three transactions, which are complement-contain {2, 4}, {3, 5}, {1, 5}, and {5}. At last, we obtain the frequent superset by taking the complement of each frequent-complement superset, showed in the table Answer.



Database

TID	Items
100	1, 3, 4
200	3, 5
300	1, 5
400	5
500	2, 4

L_1

Complement-Frequent Superset	Complement-Support
{1}	3
{2}	4
{3}	3
{4}	3

C_2

Complement-Frequent Superset	Complement-Support
{1, 2}	2
{1, 3}	2
{1, 4}	2
{2, 3}	2
{2, 4}	3
{3, 4}	1

L_2

Complement-Frequent Superset	Complement-Support
{2, 4}	3

Answer

Complement-Frequent Superset	Frequent Superset
{2, 4}	{1, 3, 5}
{1}	{2, 3, 4, 5}
{2}	{1, 3, 4, 5}
{3}	{1, 2, 4, 5}
{4}	{1, 2, 3, 5}

Fig. 4. An example for Apriori-C

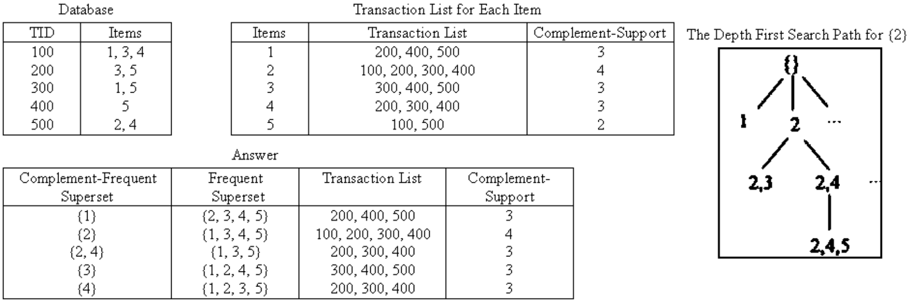
2.2 Algorithm Eclat-C

According to the characteristics of complement-frequent superset, the shorter the frequent supersets are, the longer the complement-frequent supersets are. To find shorter frequent supersets, we have to go more passes to obtain longer complement-frequent supersets. Eclat is an algorithm which is beneficial for mining frequent subset when the patterns are long[8], owing to its depth-first and transaction-list intersection mechanisms[3][9].

Eclat maintains the transaction list for each frequent itemset. Each transaction list records the set of transaction ID corresponding to the itemset. When generating 2-itemset, Eclat intersects two transaction lists to obtain the transaction list of each 2-itemset. In our study, we design an Eclat-based algorithm, named Eclat-C, which finds complement-frequent superset, and then obtain the frequent superset.

To find complement-frequent superset, we maintain transaction list to record the TID of transactions that are complement-contained in the candidates.

Example 7. Consider the database in Fig. 5, and let the minimum support be 3 transactions. The transaction list records the transaction ID that is complement-contained in the items. For example, the itemset {2} complement-contains the transactions, 100, 200, 300, and 400. The complement-support of {2} is 4, which is more than the minimum support. Because {2} is complement-frequent, the complement of the itemset {2}, that is {1, 3, 4, 5}, is a frequent superset. The right of Figure 5 shows the search path for generation of itemsets with the prefix 2. The final results are shown in the second column of the table Answer.



Database

TID	Items
100	1, 3, 4
200	3, 5
300	1, 5
400	5
500	2, 4

Transaction List for Each Item

Items	Transaction List	Complement-Support
1	200, 400, 500	3
2	100, 200, 300, 400	4
3	300, 400, 500	3
4	200, 300, 400	3
5	100, 500	2

Answer

Complement-Frequent Superset	Frequent Superset	Transaction List	Complement-Support
{1}	{2, 3, 4, 5}	200, 400, 500	3
{2}	{1, 3, 4, 5}	100, 200, 300, 400	4
{2, 4}	{1, 3, 5}	200, 300, 400	3
{3}	{1, 2, 4, 5}	300, 400, 500	3
{4}	{1, 2, 3, 5}	200, 300, 400	3

Fig. 5. An example for Eclat-C

2.3 Data Complement Technique (DCT)

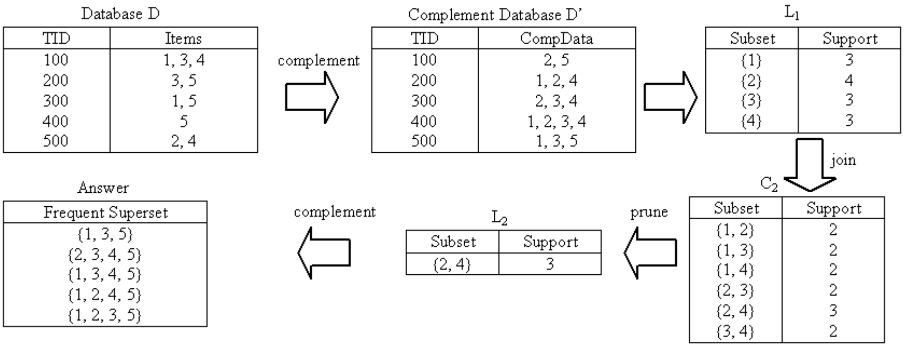
The other proposed method is so called the data complement technique (DCT). DCT utilizes the original frequent itemset mining algorithms. All of Apriori, FP-growth, Eclat, Tree-Projection, H-Mine, DHP, or other frequent subset mining algorithms can be adopted as the black box of our algorithm without any modification. We do not need to develop a new data mining system, but adopt the well-developed one. In our experiments, we choose Apriori and Eclat as instances of black boxes to compare the performance with Apriori-C and Eclat-C.

Lemma 2. If X is a frequent subset of database D with minimum support h, X' must be a frequent superset of database D' with minimum support h.

Proof. If X is a frequent subset of database D with minimum support h, there are k transactions, T_1, T_2, …, T_k, $k \geq h$, that $T_i \supseteq s$, where $1 \leq i \leq k$. We can get $I\text{-}T_i \subseteq I\text{-}X$, that is $T_i' \subseteq X'$, $1 \leq i \leq k$, and $k \geq h$. This means X' is a superset of each T_i, where $1 \leq i \leq k$, and $k \geq h$. In other words, X' is a frequent superset of D'.

According to this lemma, we can first transfer the database D into its complement, D', and then explore the frequent subsets of D'. Finally, the frequent supersets are obtained by taking the complement of those explored frequent subsets. The following gives an example.

Example 8. Fig. 6 gives an example for illustration of DCT-Apriori. Consider the database D and assume the minimum support is 3 transactions, where CompData is the complement of database. L_1 and L_2 are frequent itemset of CompData, and also frequent complement-superset of original database (See the previous example of Apriori-C). The frequent supersets are shown in the table Answer.



Database D

TID	Items
100	1, 3, 4
200	3, 5
300	1, 5
400	5
500	2, 4

Complement Database D'

TID	CompData
100	2, 5
200	1, 2, 4
300	2, 3, 4
400	1, 2, 3, 4
500	1, 3, 5

L_1

Subset	Support
{1}	3
{2}	4
{3}	3
{4}	3

C_2

Subset	Support
{1, 2}	2
{1, 3}	2
{1, 4}	2
{2, 3}	2
{2, 4}	3
{3, 4}	2

L_2

Subset	Support
{2, 4}	3

Answer

Frequent Superset
{1, 3, 5}
{2, 3, 4, 5}
{1, 3, 4, 5}
{1, 2, 4, 5}
{1, 2, 3, 5}

Fig. 6. An example for DCT-Apriori

3 Experiment Results

We evaluate the time efficiency of the proposed algorithms. Four parameters may affect the performance of the algorithms for discovering frequent superset. The first is the effect of minimum support. The second is the number of transactions. The third is the number of items, while the fourth is the average length of transactions. We choose Apriori and Eclat as the black box for DCT algorithms to compare with Apriori-C and Eclat-C. To inspect the performance of these five algorithms, baseline, Apriori-C, Eclat-C, DCT-Apriori, and DCT-Eclat, under the four parameters, we performed several experiments to measure them on an Intel Xeon 2.4GHz computer with 2GB main memory, and running FreeBSD 5.2-CURRENT. We first describe the generation of the synthetic datasets used in the assessment. Then we show the performance results of the five frequent superset discovery methods.

3.1 Synthetic Data Generation

The synthetic data was generated using the dataset generator from IBM Almaden Quest research group[10]. To create a dataset, we take the parameters for generation program shown in Table 1. Table 2 gives the descriptions for the abbreviations of values of parameters.

Table 1. Parameters for dataset generator

Parameters	Description
|D|	The number of transactions
|T|	Average size of the transactions
N	Number of items

Table 2. Descriptions for the abbreviations of values of parameters

Name	|D|	|T|	N	Size in Megabytes
D100kT25N50	100k	25	50	7.3
D10kT25N50	10k	25	50	0.75
D10kT5N40	10k	5	40	0.12
D10kT5N10	10k	5	10	0.07
D10kT25N40	10k	25	40	0.6

3.2 Performance Analysis

Table 3 depicts the execution times in seconds for the five algorithms at different support from 10% to 90%, using the D10kT10N20 dataset. According to this figure, the Eclat-based methods are much faster than all Apriori-based algorithms, especially in lower minimum support. It is because the lower the minimum support is, the more the shorter the minimum frequent superset is, and in turn, the longer the complement-frequent superset is. Besides, the Eclat-C is a little faster than DCT-Eclat.

Table 3. Execution times in seconds for the five algorithms in different minimum support, the dataset is D10kT10N20

Minimum Suppport (%)	10	20	30	40	50	60	70	80	90
Baseline	150.11	163.37	168.49	171.11	172.32	173.75	173.93	174.08	174.05
Apriori-C	1.66	0.4	0.21	0.07	0.06	0.05	0.05	0.05	0.02
DCT-Apriori	1.72	0.46	0.25	0.1	0.08	0.08	0.08	0.07	0.02
Eclat-C	0.26	0.12	0.08	0.06	0.05	0.04	0.03	0.04	0.03
DCT-Eclat	0.27	0.12	0.09	0.07	0.05	0.06	0.05	0.04	0.05

The second experiment is the evaluation of influence of the number of transactions. Table 4 shows the execution times for different number of transactions from 10k to 100k. The test dataset is T5N15 and the minimum support is set to 60%. We can see that the execution times are increasing linearly with the number of transactions.

Table 4. Execution times in second for five algorithms in different number of transactions from 10k to 100k and the datasets are T5N15

Number of Transactions (k)	10	20	30	40	50	60	70	80	90	100
Baseline	2.8	3.04	3.27	3.32	3.42	3.55	3.65	3.68	3.73	3.8
Apriori-C	0.02	0.05	0.08	0.11	0.13	0.16	0.19	0.22	0.25	0.28
DCT-Apriori	0.06	0.11	0.18	0.24	0.29	0.35	0.42	0.48	0.54	0.6
Eclat-C	0.02	0.04	0.07	0.09	0.13	0.14	0.16	0.2	0.23	0.26
DCT-Eclat	0.03	0.08	0.12	0.17	0.21	0.26	0.31	0.36	0.4	0.46

The third parameter is the total number of items in the transaction database. Basically, when the number of items increases, the size of the complement of a itemset increases, too. Table 5 shows the execution times for different number of items from 10 to 40, and use D10kT5 datasets. When the number of items is more than 25, the baseline algorithm falls into memory exhausting.

Table 5. Execution times in second for five algorithms in different number of itmes from 10 to 40 and the datasets are D10kT5

Number of Items	10	15	20	25	30	35	40
Baseline	0.08	2.78	226.2	NA	NA	NA	NA
Apriori-C	0.02	0.02	0.03	0.06	0.44	7.61	55.53
DCT-Apriori	0.02	0.05	0.09	0.22	0.99	6.88	62.75
Eclat-C	0.02	0.01	0.03	0.06	0.1	0.28	0.94
DCT-Eclat	0.01	0.04	0.54	0.08	0.14	0.33	0.99

The last is the average size of transactions. When the average size of transactions increases, the size of the complement of an itemset decreases. The execution becomes slow, especially when |T| are small. Table 6 shows the result, and we can see that the depth first approach, Eclat-based algorithm, is more efficient when the average size of transactions is small.

Table 6. Execution times in second for five algorithms in different average size of transactions from 10 to 30 and the datasets are D10kN50

Average Size of Transactions	10	15	20	25	30
Baseline	NA	NA	NA	NA	NA
Apriori-C	473.37	26.42	1.08	0.24	0.17
DCT-Apriori	518.99	19.25	1.71	0.37	0.17
Eclat-C	4.88	0.53	0.22	0.12	0.11
DCT-Eclat	4.92	0.57	0.23	0.13	0.09

4 Conclusion and Discussion

In this paper, we proposed three methods, Apriori-C, Eclat-C, and data complement technique (DCT) for discovering frequent superset, which are all based on the set complement view of point. We do not find frequent supersets directly, but the complement of frequent supersets. We performed several experiments to evaluate our three algorithms in different dataset and diverse support. In DCT, we choose Apriori and Eclat to compare with Apriori-C and Eclat-C. We modify the original Apriori algorithm as a baseline in addition. The experiment results show that all of these three algorithms are more time efficient than the baseline, and the Eclat-C is most time efficient. When the number of transactions increase, the execution time of the three algorithms increase linearly. Due to the Eclat-based algorithm is depth first algorithm, so the transaction size does not influence the execution time of Eclat-based approach too much.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. Proceedings of the 20th VLDB Conference Santiago, Chile (1994)
2. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidate Generation. ACM SIGMOD (2000)
3. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New Algorithms for Fast Discovery of Association Rules. ACM SIGKDD (1997)
4. Agarwal, R.C., Aggarwal, C.C., Prasad, V.V.V.: A Tree Projection Algorithm For Generation of Frequent Itemsets. Journal on Parallel and Distributed Computing (Special Issue on High Performance Data Mining), Vol 61, Issue 3 (2000) 350-371
5. Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., Yang, D.: H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Database. International Conference on Data Mining, ICDM (2001)
6. Park, J.S., Chen, M.-S., Yu, P.S.: An Effective Hash-Based Algorithm for Mining Association Rules. In Proc. of ACM SIGMOD (1995) 175-186
7. Bodon, F.: A Fast APRIORI implementation. Workshop on Frequent Itemset Mining Implementions, FIMI'03 (2003)
8. Agarwal, R. C., Aggarwal, C.C., Prasad, V.V.V.: Depth First Generation of Long Patterns. In Proc. of ACM SIGKDD (2000) 108-118
9. Hipp, J., Guntzer, U., Nakhaeizadeh, G.: Algorithms for Association Rule Mining – A General Survey and Comparison. ACM SIGKDD Explorations, Vol. 2, Issue 1 (2000) 58-64
10. IBM Almaden Research Center, Intelligent Information System, http://www.almaden.ibm.com/software/quest/Resources/index.shtml

Improving Direct Counting for Frequent Itemset Mining

Adriana Prado[⋆], Cristiane Targa[∗], and Alexandre Plastino[⋆⋆]

Department of Computer Science, Universidade Federal Fluminense
Rua Passo da Pátria, 156 - Bloco E - 3^o andar - Boa Viagem
24210-240, Niterói, RJ, Brazil
{aprado,ctarga,plastino}@ic.uff.br
http://www.ic.uff.br

Abstract. During the last ten years, many algorithms have been proposed to mine frequent itemsets. In order to fairly evaluate their behavior, the *IEEE/ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)* has been recently organized. According to its analysis, kDCI++ is a state-of-the-art algorithm. However, it can be observed from the *FIMI'03* experiments that its efficient behavior does not occur for low minimum supports, specially on sparse databases. Aiming at improving kDCI++ and making it even more competitive, we present the kDCI-3 algorithm. This proposal directly accesses candidates not only in the first iterations but specially in the third one, which represents, in general, the highest computational cost of kDCI++ for low minimum supports. Results have shown that kDCI-3 outperforms kDCI++ in the conducted experiments. When compared to other important algorithms, kDCI-3 enlarged the number of times kDCI++ presented the best behavior.

1 Introduction

Association rules represent an important type of information extracted from data mining processes that describe interesting relationships among data items of a specific knowledge domain. *Market basket analysis* is the typical application of *association rule mining* (*ARM*) and consists in identifying relationships among products that significantly occur in customers buys. For instance, the rule "a customer who buys bean, in general buys rice", represented by $\{bean\} \Rightarrow \{rice\}$, would certainly be extracted from a market database in Brazil. Formally, an association rule, defined over a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$, is an implication of the form $X \Rightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, $X \;/= \emptyset Y \;/= \emptyset$, and$X \cap Y = \emptyset$. We say that X is the antecedent and Y is the consequent of the rule.

Let $\mathcal{D}$ be a set of transactions (a transactional database) defined over $\mathcal{I}$, where each transaction t is a subset of $\mathcal{I}$ ($t \subseteq \mathcal{I}$). Then, the rule $X \Rightarrow Y$ holds in $\mathcal{D}$ with support s and confidence c if, respectively, s% of the transactions in $\mathcal{D}$ contain $X \cup Y$, and c% of the transactions in $\mathcal{D}$ that contain X also contain Y.

[⋆] Work sponsored by CAPES Master scholarship.

[⋆⋆] Work sponsored by CNPq research grant 300879/00-8.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 371–380, 2004.

The *ARM* problem is commonly broken into two phases. Let *minsup* and *minconf* be, respectively, the user specified minimum support and confidence. The first phase, the *frequent itemset mining* (*FIM*) phase, consists in identifying all *frequent itemsets* (sets of items) that occur in at least *minsup*% of the transactions. The second phase outputs, for each identified frequent itemset Z, all association rules $A \Rightarrow B$ with confidence greater than or equal to *minconf*, such that $A \subset Z$, $B \subset Z$, and $A \cup B = Z$. The *FIM* phase demands more computational effort than the second one and has been intensively addressed [3].

1.1 Previous Work

During the last ten years, many algorithms have been proposed to efficiently mine frequent itemsets. Most of them are improved variations of Apriori [1]. In this strategy, the set F_1 containing all frequent itemsets of length 1 (1-itemsets) is initially identified. Then, at each iteration $k \geq 2$: (a) the set C_k of candidates of length k (k-candidates) is generated by combining all pairs included in F_{k-1} (frequent $(k-1)$-itemsets) that share a common $(k-2)$-prefix; (b) C_k is pruned in order to eliminate candidates that have at least one $(k-1)$-subset that is not frequent (all subsets of a frequent itemset are also frequent); (c) then, in the *counting phase*, the database $\mathcal{D}$ is read and, for each transaction t of $\mathcal{D}$, the support of all k-candidates contained in t is incremented. After reading the whole database, F_k is identified. If this set is empty, the termination condition is reached. In the Apriori algorithm, the candidates are stored in a hash-tree.

Subsequent proposed algorithms have improved Apriori by reducing: the number of database scans [11, 12], the computational cost of the counting phase [2, 7–9], and the number and size of transactions to be scanned [7–9].

Another class of *FIM* algorithms mines frequent itemsets by adopting a depth-first approach. The Eclat algorithm [14] uses an in-memory vertical layout of the database and an intersection-based approach to determine the supports of the candidates. The FP-growth algorithm [5] does not generate candidates. It builds a compact in-memory representation of the database, called FP-tree (*frequent pattern tree*), from which the support of all frequent itemsets are derived. According to experiments presented in [5], FP-growth is very efficient. However, it did not show good performance on sparse databases [15].

OpportuneProject [6], PatriciaMine [10] and FPgrowth* [4] are recent *FIM* algorithms that have improved the ideas adopted by FP-growth. OpportuneProject is able to choose between two different data structures according to the database features. PatriciaMine uses only one data structure (*Patricia trie*) to represent both dense and sparse databases together with optimizations that reduce the cost of tree traversal and that of physical database projections. FPgrowth* introduces an array-based technique also aiming at reducing FP-trees traversals.

DCI (*Direct Count & Intersect*) [8] and its recent version kDCI++ [7] are apriori-like algorithms. During their first iterations, they exploit database pruning techniques, inspired in [9], and efficient data structures to store candidates. When the vertical layout of the pruned database fits into main memory, the supports of the candidates are obtained by an intersection-based technique. Both of them can adapt their behavior according to the database features. Moreover, kDCI++ uses a counting inference strategy based on that presented in [2].

A recent and efficient proposed method called LCM-freq [13] mines all frequent itemsets from frequent closed itemsets (an itemset is a closed itemset if none of its supersets have the same support).

1.2 Motivation

As pointed out in [3], every new proposed *FIM* algorithm is many times evaluated by limited experimental tests. In order to fairly evaluate the behavior of these algorithms, the *IEEE/ICDM Workshop on Frequent Itemset Mining Implementation (FIMI'03)* has been recently organized. All the accepted *FIM* implementations can now be found in the *FIMI repository* (available at http://fimi.cs.helsinki.fi/) together with an extensive performance evaluation.

According to the *FIMI'03 Workshop* experiments, kDCI++ is a state-of-the-art algorithm. However, it can be observed from its analysis [3] that its highly efficient behavior is not true for low values of support, specially on sparse databases.

Aiming at evaluating kDCI++ under these specific circumstances, we performed experiments on different combinations of databases and minimum supports also used in [2–8, 10, 13, 15]. We observed that the third iteration presented a very high computational cost compared to the other iterations.

Table 1 presents, for different databases and low minimum supports, the total execution times (in seconds) of kDCI++ and its third iteration execution times (in seconds). The databases are described in Section 4. The last column represents the ratio between the third iteration execution time and the total execution time. We observed that, on average, the third iteration represented 65% (ranged from 34% to 83%) of the total execution time of kDCI++ runs. This is due to the huge number of 3-candidates that must be evaluated. Indeed, as already observed in [7, 8], the third iteration may represent a bottleneck in apriori-like algorithms.

In this work, in order to improve the kDCI++ algorithm, we propose a strategy, called kDCI-3, that enables the direct counting at the third iteration, one of its most time consuming iterations for low minimum supports, specially on sparse databases. This proposal is based on a directly accessible data structure to store candidates, which allows a more efficient 3-candidate counting.

The paper is organized as follows. In Section 2, we review the kDCI++ algorithm. In Section 3, we present the kDCI-3 algorithm. The experimental results are reported and discussed in Section 4. Finally, in Section 5, some concluding remarks are made and future work is pointed out.

2 The kDCI++ Algorithm

The kDCI++ algorithm [7] is an apriori-like strategy. During its first iterations, it exploits directly accessible data structures together with database pruning techniques that reduce the number and size of transactions to be scanned.

Table 1. Execution times of kDCI++

Database (*minsup* - %)	Total time	3^{rd} iteration time	(%)
T20I10N1KP5C0.25D200K (0.1)	544	353	65
T20I10N1KP5C0.25D200K (0.3)	51	34	67
T10I5N1KP5C0.25D200K (0.01)	298	220	74
T10I5N1KP5C0.25D200K (0.03)	104	70	67
T30I15N1KP5C0.25D200K (0.25)	655	541	83
T30I15N1KP5C0.25D200K (0.5)	139	116	83
T25I10D10K (0.1)	26	21	81
T25I10D10K (0.2)	4	1.9	47
T40I10D100K (0.5)	386	271	70
T40I10D100K (0.75)	133	101	76
T10I4D100K (0.01)	307	103	34
T10I4D100K (0.03)	36	28	78
T30I16D400K (0.4)	775	487	63
T30I16D400K (0.6)	234	161	69
BMS-POS (0.1)	422	156	37
BMS-POS (0.3)	47	22	47

For $k = 2$, kDCI++ builds a *prefix table* P_2 of $\binom{|F_1|}{2}$ entries. Each entry of P_2 is a counter that represents a 2-candidate and accumulates its support.

To directly access the entry (counter) associated with a generic candidate $c{=}(c_1{,}c_2)$, where $c_1 < c_2$, kDCI++ maps c into a pair $\{x_1{,}x_2\}$ where $x_1{=}\mathcal{T}(c_1)$, $x_2{=}\mathcal{T}(c_2)$, and $\mathcal{T}$ is a strictly monotonous increasing function defined by $\mathcal{T} : F_1 \rightarrow \{1, \ldots, |F_1|\}$. Equation 1 is thus adopted by kDCI++ in order to find the entry of P_2 that represents c=$\{c_1, c_2\}$, called here $EP_2(c_1, c_2)$. This equation is derived considering that the counters associated with pairs $\{1{,}x_2\}$, $2 \leq x_2 \leq |F_1|$ are stored in the first ($|F_1|$-1) positions of P_2, the counters associated with pairs $\{2{,}x_2\}$, $3 \leq x_2 \leq |F_1|$, are stored in the next ($|F_1|$-2) positions, and so on.

$$EP_2(c_1, c_2) = \sum_{i=1}^{x_1-1} (|F_1| - i) + (x_2 - x_1) = |F_1|(x_1 - 1) - \frac{x_1(x_1 - 1)}{2} + x_2 - x_1 \ . \quad (1)$$

At the end of the counting phase, if the corresponding entry of c is greater than or equal to *minsup*, c is included in F_2.

When $k > 2$, another data structure was proposed. A *prefix table* P_k of $\binom{|M_k|}{2}$ entries is built, where M_k is the set of items at iteration k that were not pruned during execution progress. Each entry of P_k contains a pointer to the beginning of a memory section that stores the ordered k-candidates having the same 2-prefix. The entry of P_k that represents the prefix $\{c_1, c_2\}$ of a candidate $c{=}\{c_1, c_2, \ldots, c_k\}$ is obtained similarly to the second iteration.

To obtain the support of each k-candidate, for each transaction t, kDCI++ determines all possible 2-prefixes of all k-itemsets in t. Then, for each 2-prefix, the entry i of P_k is obtained and the section of ordered candidates that must be evaluated will be delimited by $P_k[i]$ and $P_k[i + 1]$.

At each iteration $k \geq 2$, kDCI++ checks whether the vertical layout of the pruned database fits into main memory. The vertical layout can be seen as a set of $|M_k|$ bit vectors of size $|T_k|$, where T_k is the set of not pruned transactions at that iteration. If the bit vector associated with an item i has its j^{th} bit equal to 1, item i is present in the j^{th} transaction. If the database is small enough, its vertical representation is built and the supports of the $(k+1)$-candidates is thus obtained by the intersections of the k bit vectors associated with their items.

3 The kDCI-3 Algorithm

Aiming at improving the performance of kDCI++, in this section, we propose the kDCI-3 algorithm, which uses an efficient direct counting technique to determine the support of all 3-candidates.

The data structure used by kDCI-3 during the third iteration is based on the *prefix table* P_2 and on a new *array* C which represents all 3-candidates.

In the kDCI-3 algorithm, an entry of P_2 related to the frequent 2-itemset $\{c_1,c_2\}$ stores two values. The first one, represented by $Pt(c_1, c_2)$, is a pointer to the first entry of the contiguous section in C associated with all 3-candidates (in lexicographic order) having the same 2-prefix $\{c_1,c_2\}$. This value is null if there is no 3-candidate with that 2-prefix. The second value is the order, represented by $Ord(c_1, c_2)$, in which the frequent 2-itemset $\{c_1,c_2\}$ is represented in P_2. $Ord(c_1, c_2)$ is null if $\{c_1,c_2\}$ is not a frequent 2-itemset. To access P_2, kDCI-3 utilizes Equation 1, similarly to kDCI++.

The entry of C which corresponds to a generic 3-candidate $c=\{c_1,c_2,c_3\}$, called here $EC(c_1, c_2, c_3)$ and defined by Equation 2, can be found from the pointer $Pt(c_1, c_2)$ together with an offset value. This offset is defined by the number of frequent 2-itemsets represented between $\{c_1,c_2\}$ and $\{c_1,c_3\}$ in P_2.

$$EC(c_1, c_2, c_3) = Pt(c_1, c_2) + ((Ord(c_1, c_3) - Ord(c_1, c_2)) - 1) \ . \tag{2}$$

Figure 1 illustrates how kDCI-3 identifies the entry associated with a given 3-candidate. In this example, F_1={0,1,3,7,9} and F_2={{0,1},{0,3},{0,7},{0,9}, {1,3},{1,9},{3,9}}. The set of 3-candidates (C_3), represented by C, is generated in lexicographic order by the combination of the frequent 2-itemsets that share a common 1-prefix. For instance, the frequent 2-itemset {0,1} will be combined with all subsequent frequent 2-itemsets in P_2 that have 0 as their first item, in order to generate all 3-candidates with {0,1} as their 2-prefix. Since the frequent 2-itemset {0,9} is the third one after {0,1}, it is easy to conclude that the candidate {0,1,9} is the third candidate in C having {0,1} as their 2-prefix. Then, the 3-candidate {0,1,9}, generated by {0,1} and {0,9}, is located in C two entries after the first 3-candidate sharing the prefix {0,1}. Indeed, this two entries are given by $((Ord(0, 9) - Ord(0, 1)) - 1)$ and the first 3-candidate sharing the prefix {0,1} is identified by $Pt(0, 1)$.

To obtain the support of each 3-candidate, for each transaction t, kDCI-3 determines all the possible 3-itemsets included in t. After that, for each 3-itemset $\{c_1,c_2,c_3\}$ in t, it identifies the entries of P_2 that represents the 2-itemsets

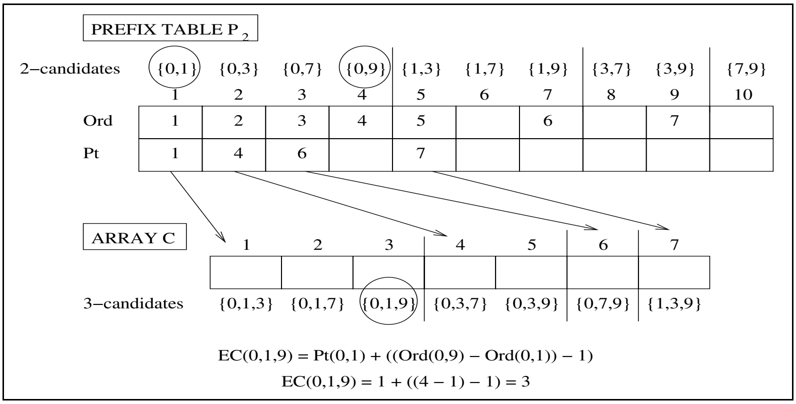


Fig. 1. Data structure used by kDCI-3 to access 3-candidates

Table 2. Databases used in the experiments and its corresponding characteristics

Database	Items	Transactions	Avg. Length	References
T20I10N1KP5C0.25D200K	1000	197,437	20.1	[3]
T10I5N1KP5C0.25D200K	1000	192,889	10.3	[3]
T30I15N1KP5C0.25D200K	1000	199,093	29.6	[3]
T25I10D10K	1001	9,219	27.7	[2, 5, 7, 8]
T40I10D100K	1000	100,000	39.6	[4, 10, 13]
T10I4D100K	1000	100,000	10.1	[10, 13]
T30I16D400K	1000	397,487	29.7	[7, 8, 10]
BMS-POS	1657	515,597	6.5	[3, 6, 10, 13, 15]

$\{c_1,c_2\}$ and $\{c_1,c_3\}$. If the identified entries represent frequent 2-itemsets, the entry associated with $\{c_1,c_2,c_3\}$, in C, is found from Equation 2 and is then incremented. At the end of the candidate counting, if an entry in C is greater than or equal to *minsup*, its corresponding candidate is included in F_3.

4 Performance Evaluation

The computational experiments reported in this work have been carried out on a 600 MHz Pentium III PC with 256 MB of RAM memory under the RedHat Linux 9.0 (kernel version 2.4.20) operating system.

The databases used in the experiments are described in Table 2. The first three databases were generated by the IBM dataset generator [1]. The databases T40I10100K, T10I4D100K and BMS-POS (real database) were downloaded from the *FIMI repository*, and finally, T25I10D10K and T30I16D400K were downloaded from the DCI site (*http://miles.cnuce.cnr.it/~palmeri/datam/DCI*).

This section is organized as follows. In Subsection 4.1, we compare the effectiveness of the new technique used by kDCI-3 with that adopted by kDCI++ during the third iteration. In Subsection 4.2, we compare the performance of kDCI-3 with recent *FIM* algorithms evaluated in the *FIMI'03 Workshop*.

4.1 Effectiveness of the New Direct Counting Technique

For kDCI++, we used the source code available at the DCI site, now also available at the *FIMI repository*. For kDCI-3, we adapted the source code of kDCI++ in order to implement kDCI-3 features and to allow a fair comparison between them.

Table 3 shows, for the same combinations of databases and low minimum supports presented in Table 1, the total execution times (in seconds) and the third iteration execution times (in seconds) of kDCI++ and kDCI-3, respectively. The fourth column represents the ratio between the third iteration execution time of kDCI-3 and that of kDCI++. The seventh column represents the ratio between the total execution time of kDCI-3 and that of kDCI++.

Table 3. Execution times of kDCI++ and kDCI-3

	3rd iteration times			Total times		
Database (*minsup* - %)	kDCI++	kDCI-3	(%)	kDCI++	kDCI-3	(%)
T20I10N1KP5C0.25D200K (0.1)	353	40	11	544	191	35
T20I10N1KP5C0.25D200K (0.3)	34	10	29	51	27	53
T10I5N1KP5C0.25D200K (0.01)	220	12	5	298	90	30
T10I5N1KP5C0.25D200K (0.03)	70	7	10	104	48	46
T30I15N1KP5C0.25D200K (0.25)	541	86	16	655	209	32
T30I15N1KP5C0.25D200K (0.5)	116	31	27	139	60	43
T25I10D10K (0.1)	21	4	19	26	9	35
T25I10D10K (0.2)	1.9	1.3	68	4	3.7	93
T40I10D100K (0.5)	271	69	25	386	208	54
T40I10D100K (0.75)	101	39	39	133	72	54
T10I4D100K (0.01)	103	6	6	307	202	66
T10I4D100K (0.03)	28	3	11	36	16	44
T30I16D400K (0.4)	487	132	27	755	478	63
T30I16D400K (0.6)	161	67	42	234	154	66
BMS-POS (0.1)	156	9	6	422	216	51
BMS-POS (0.3)	22	5	23	47	25	53

Due to the large number of 3-candidates generated in these tests (up to 11,051,970 on T10I4D100K (0.01%)), kDCI++ presented a poor performance in the third iteration. We can note that kDCI-3 efficiently reduced the execution time of this time consuming phase. We can observe, for instance, that in the first line of Table 3, while kDCI++ executed the third iteration in 353 seconds, kDCI-3 executed it in 40 seconds. This represents a reduction of 89%. Since the third iteration represents in this run a great part of the total execution time (as observed in Subsection 1.2), kDCI-3 total execution time was 35% of that of kDCI++ (a reduction of 65%).

The elapsed times of kDCI-3 were, for the third iteration, on average, 23% (from 5% to 68%) of that of kDCI++ (a reduction, on average, of 77%). We can also observe the impact produced by the proposed direct counting technique during the third iteration in the total execution times. The relative total execution times of kDCI-3 ranged from 30% to 66% (with a discrepant value of 93%) representing, on average, 51% of the total execution times of kDCI++.

4.2 Performance Comparison

This section reports a performance comparison among kDCI-3 and four preeminent algorithms: kDCI++, PatriciaMine, FPgrowth*, and LCM-freq. According to [3], kDCI++ and PatriciaMine are considered state-of-the-art algorithms. We also selected FPgrowth* and LCM-freq since they showed a good behavior in the *FIMI'03* experiments, specially for low supports. For PatriciaMine, FPgrowth* and LCM-freq, we used the source codes available at the *FIMI repository*.

Figure 2 shows the relative execution times of kDCI-3 and the other algorithms. In each picture, the value 1 (in y-axis) represents the relative execution time of the worst algorithm for the correspondent minimum support. The value in brackets beside the minimum support (in x-axis) is the absolute execution time of the worst algorithm (in seconds).

We observe that kDCI-3 was better than kDCI++ for almost all combinations of databases and minimum supports. In graph (e), for the highest support (0.9%), and in graph (f) for the three highest supports (0.5%, 0.4%, and 0.3%), kDCI++ was slightly better than kDCI-3 due to a not relevant number of 3-candidates.

For lower values of minimum support in graphs (e) and (f), when the number of 3-candidates increases, kDCI-3 performs much better than kDCI++. In some cases, as pointed out in [7], the bad behavior of kDCI++ can be justified by the size of C_3, which can lead to “a lot of useless work to determine the support of many candidate itemsets which are not frequent”.

In graphs (g) and (h), kDCI++ was always the worst algorithm. For all values of support, kDCI-3 was better than kDCI++, making it more competitive.

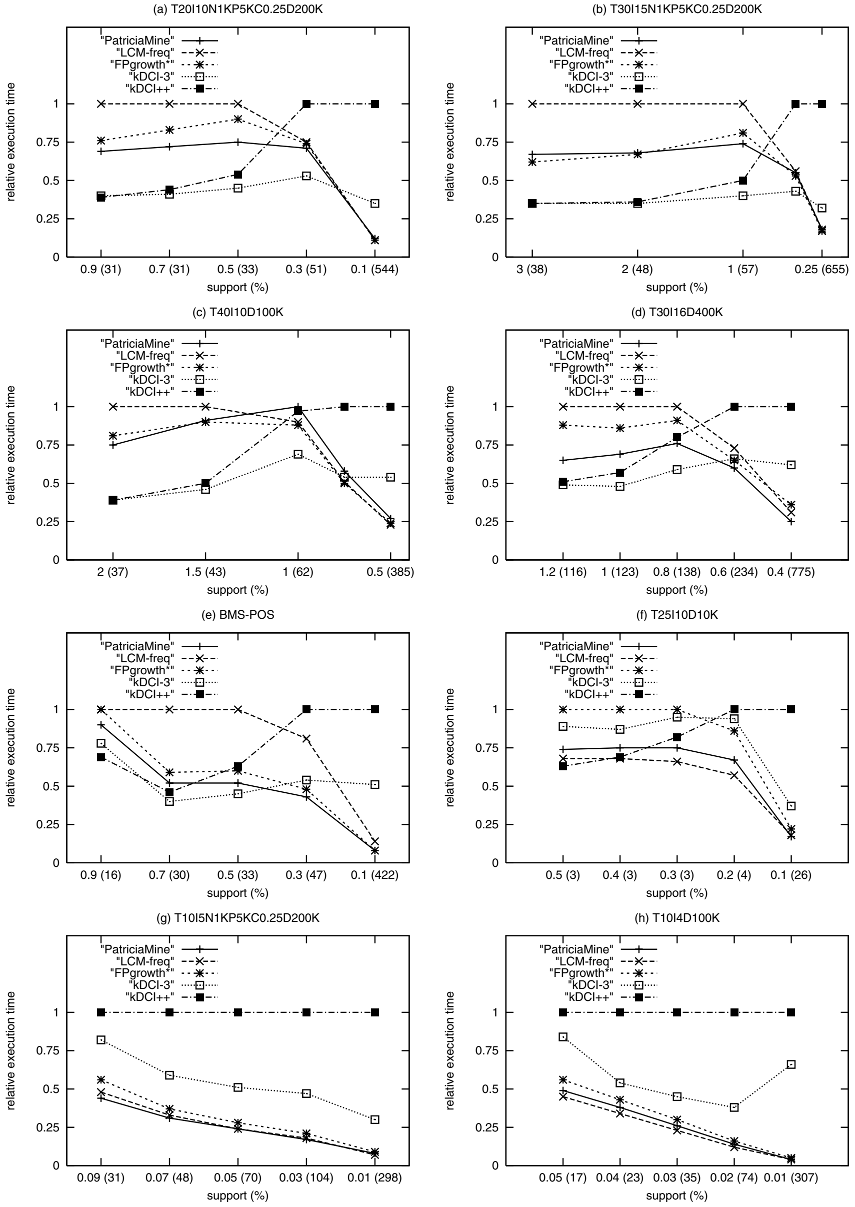
In graphs (a)-(d), we observe that for higher minimum supports, kDCI-3 runs faster than kDCI++, which was the best algorithm in these situations. For lower minimum supports, however, kDCI++ performance decreases, but kDCI-3 is still better than kDCI++, also making it more competitive. In these graphs (and also in graph(e)), for some values of support, kDCI++ was the worst algorithm (or almost the worst), while the improvements of kDCI-3 made it the best one.

5 Conclusions

In this work we proposed kDCI-3, an extension of the kDCI++ algorithm – a recent and important method for the *FIM* problem. kDCI-3 provided an efficient direct counting of candidates when dealing with a huge number of 3-candidates, a consequence of low minimum support values, specially in sparse databases.

From the conducted computational experiments, we observed that kDCI-3 presented a very efficient behavior. kDCI-3 reduced the execution times of the third iteration of kDCI++ and, consequently, its total elapsed times in almost all executed experiments. We believe that these results represent an important contribution that improves a state-of-the-art algorithm.

Based on the encouraging observed results, as future work, we intend to investigate the use of the proposed direct counting of candidates in other iterations.


(a) T20I10N1KP5KC0.25D200K

(b) T30I15N1KP5KC0.25D200K

(c) T40I10D100K

(d) T30I16D400K

(e) BMS-POS

(f) T25I10D10K

(g) T10I5N1KP5KC0.25D200K

(h) T10I4D100K

Fig. 2. Relative execution times of kDCI-3 and other algorithms

References

1. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *20th VLDB Conference*, 1994.
2. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L.Lakhal. Mining Frequent Patterns with Counting Inference. In *ACM SIGKDD Explorations*, v.2, n.2, 2000.

3. B. Goethals and M. J. Zaki. Advances in Frequent Itemset Mining Implementations: Introduction to FIMI'03. In *IEEE ICDM FIMI Workshop*, 2003.
4. G. Grahne, J. Zhu. Efficiently Using Prefix Trees in Mining Frequent Itemsets. In *IEEE ICDM FIMI Workshop*, 2003.
5. J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *ACM SIGMOD Conference*, 2000.
6. J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. In *8th ACM SIGKDD Conference*, 2002.
7. S. Orlando, P. Palmerimi, R. Perego, C. Lucchese, and F. Silvestri. kDCI++: A Multi–Strategy Algorithm for Discovering Frequent Sets in Large Databases. In *IEEE ICDM FIMI Workshop*, 2003.
8. S. Orlando, P. Palmerimi, and R. Perego. Adaptive and Resource–Aware Mining of Frequent Sets. In *IEEE ICDM Conference*, 2002.
9. J. S. Park, M. Chen, and P. S. Yu. An Effective Hash-Based Algorithm for Mining Association Rules. In *ACM SIGMOD Conference*, 1995.
10. A. Pietracaprina and D. Zandolin. Mining Frequent Itemsets using Patricia Tries. In *IEEE ICDM FIMI Workshop*, 2003.
11. A. Savasere, E. Omiecinski, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *21th VLDB Conference*, 1995.
12. H. Toivonen. Sampling Large Databases for Association Rules. In *22th VLDB Conference*, 1996.
13. T. Uno, T. Asai, Y. Uchida, and H. Arimura. LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets. In *IEEE ICDM FIMI Workshop*, 2003.
14. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. In *3rd ACM SIGKDD Conference*, 1997.
15. Z. Zheng, R. Kohavi, and L. Mason. Real World Performance of Association Rule Algorithms. In *7th ACM SIGKDD Conference*, 2001.

Mining Sequential Patterns with Item Constraints

Show-Jane Yen and Yue-Shi Lee

Department of Computer Science & Information Engineering, Ming Chuan University
5 The-Ming Rd., Gwei Shan District, Taoyuan County 333, Taiwan
{sjyen,leeys}@mcu.edu.tw

Abstract. *Mining sequential patterns* is to discover sequential purchasing behaviors for most customers from a large amount of customer transactions. Past transaction data can be analyzed to discover customer purchasing behaviors. However, the size of the transaction database can be very large. It is very time consuming to find all the sequential patterns from a large database, and users may be only interested in some items. Moreover, the criteria of the discovered sequential patterns for the user requirements may not be the same. Many uninteresting sequential patterns for the user requirements can be generated when traditional mining methods are applied. Hence, a data mining language needs to be provided such that users can query only interesting knowledge to them from a large database of customer transactions. In this paper, a data mining language is presented. From the data mining language, users can specify the interested items and the criteria of the sequential patterns to be discovered. Also, an efficient data mining technique is proposed to extract the sequential patterns according to the users' requests.

1 Introduction

The definitions about mining sequential patterns are presented as follows [2, 3, 4, 5]: A *customer sequence* is the list of all the transactions of a customer, which is ordered by increasing transaction-time. A customer sequence *c supports* a sequence *s* if *s* is contained in *c*. The *support* for a sequence *s* (or an itemset *i*) is defined as the ratio of the number of customer sequences that supports *s* (or *i*) to the total number of customer sequences. If the support for a sequence *s* (or an itemset *i*) satisfies the user-specified minimum support threshold, then *s* (or *i*) is called *frequent sequence* (or *frequent itemset*). The *length* of an itemset *X* is the number of items in the itemset *X*, and the *length* of a sequence *s* is the number of itemsets in the sequence. An itemset of length *k* is called a *k-itemset*, and a frequent itemset of length *k* a *frequent k-itemset*. Also, a sequence of length *k* is called a *k-sequence*, and a frequent sequence of length *k* a *frequent k-sequence*. A *sequential pattern* is a frequent sequence that is not contained in any other frequent sequence.

For designing a data mining language [6], two important issues need to be considered: the easy-to-use user interface and the efficient data mining language processing. This paper is concerned with the two issues. We present a data mining language, from which users only need to specify the criteria and the interested items for discovering the sequential patterns.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 381–390, 2004.

2 Data Mining Language and Database Transformation

In this section, we propose a data mining query language and transform the original transaction data into another type to improve the efficiency of query processing.

2.1 Data Mining Language

In this section, we present a data mining language. Users can query sequential patterns by specifying the related parameters in the data mining language. The data mining language is defined as follows:

Mining <*Sequential Patterns*>
From <*CSD*>
With <*{D_1},{D_2}, …,{D_m}*>
Support <*s%*>

1. In the **Mining** clause, <*Sequential Patterns*> is specified because the discovered knowledge is sequential patterns.
2. In the **From** clause, <*CSD*> is used to specify the database name to which users query the sequential patterns.
3. In the **With** clause, <*{D_1},{D_2}, …,{D_m}*> *are* user-specified items which ordered by increasing purchasing time, Besides, the notation "*" can be in the itemsets D_i, which denotes any itemsets and *{D_i}* can be the notation "*", which represents any sequence.
4. **Support** clause is followed by the user-specified minimum support *s%*.

2.2 Database Transformation

In order to find the interesting sequential patterns efficiently, we need to transform the original transaction data into another type. Each item in each customer sequence is transformed into a bit string. The length of a bit string is the number of the transactions in the customer sequence. If the *i*th transaction of the customer sequence contains an item, then the *i*th bit in the bit string for this item is set to 1. Otherwise, the *i*th bit is set to 0.

For example, in Table 1, the customer sequence in CID 1 contains items A, C and E. Because item A is contained in the second and the third transactions in this customer sequence, the second and the third bits in the bit string for item A in this customer sequence is 1s and the other bits in the bit string is 0s. The bit string for item A in CID 1 is 011. Hence, we can transform the customer sequence database (Table 1) into the *bit-string database* (Table 2).

Table 1. customer sequence database (CSD)

CID	Customer sequence
1	{C}{AC}{ACE}
2	{AE}{A}{ACE}{CE}
3	{C}{E}{E}{CE}
4	{BD}{AE}{BC}{AE}{ABE}{F}
5	{D}{DEF}{CEF}{AD}{BD}{DF}

Table 2. bit-string database

CID	Items	Bit string for each item
1	A, C, E	011,111,001
2	A, C, E	1110,0011,1011
3	C, E	1001,0111
4	A, B, C, D, E, F	010110,101010,001000,100000,010110,000001
5	A, B, C, D, E, F	000100,000010,001000,110111,011000,011001

2.3 Sequential Bit-String Operation

Suppose a customer sequence contains the two sequences S_1 and S_2. We present an operation called *sequential bit-string operation* to check if the sequence S_1S_2 is also contained in this customer sequence. The process of the sequential bit-string operation is described as follows: Let the bit string for sequence S_1 in customer sequence c is B_1, and for sequence S_2 is B_2. Bit string B_1 is scanned from left to right until a bit value 1 is visited. We set this bit and all bits on the left hand side of this bit to 0 and set all bits on the right hand side of this bit to 1, and assign the resultant bit string to a template T_b. Then, the bit string for sequence S_1S_2 in c can be obtained by performing logical AND operation on bit strings T_b and B_2. If the number of 1's in the bit string for sequence S_1S_2 is not zero, then S_1S_2 is contained in customer sequence c. Otherwise, the customer sequence c does not contain S_1S_2.

For example, consider Table 1. We want to check if sequence {A}{C} is contained in customer sequence CID 1. From Table 2, we can see that items A and C are contained in customer sequence CID 1, and the bit string of items A and C in CID 1 are B_A=011 and B_C=111, respectively. We scan the bit string B_A from left to right, and generate the template bit string T_b=001. By performing logical AND operation on T_b and B_C, we can obtain that the bit string for sequence {A}{C} in customer sequence CID 1 is 001, in which the number of 1's is not zero. Hence, the sequence {A}{C} is contained in the customer sequence CID 1, and the resultant bit string 001 is the bit string for the sequence {A}{C} in the customer sequence CID 1.

3 Mining Interesting Sequential Patterns

In this section, we describe how to process user's query and find the interesting sequential patterns. For a user's query, if there is no notation "*" specified in the **With** clause, then this query is to check if the sequence followed by the **With** clause is a frequent sequence. We call the type of user's queries the Type I query. If the user would like to extract the sequential patterns which contain other sequences except the sequences specified in the **With** clause, then the notation "*"s have to be specified in the **With** clause. We call this type of user's queries the Type II query. In the following, we discuss the query processing for the two types of user queries.

3.1 Query Processing for Type I Query

Suppose the specified sequence $S=\{D_1\}\{D_2\}...\{D_m\}$ in the **With** clause, where D_i is an itemset. The method to check if sequence S is a frequent sequence is described as follows:

Step 1. Scan the bit-string database and find the number of customer sequences which contain the sequence S.

For each record in the bit-string database, if the customer sequence contains all items in sequence S, then scan sequence S from left to right. For each itemset D_i ($1 \leq i \leq m$) in sequence S, perform the logical AND operation on the bit strings for all items in D_i, and the resultant bit string is the bit string for itemset D_i, If the bit string for itemset D_i is not zero, then perform the sequential bit-string operation on the bit strings for D_i and D_{i+1}. The resultant bit string is the bit-string for sequence $\{D_i\}\{D_{i+1}\}$. Then, perform the sequential bit-string operation on the bit strings for sequence $\{D_i\}\{D_{i+1}\}$ and itemset D_{i+2}, and so on. During performing those operations, if the resultant bit string is zero, then we do not need to continue the process, because we can sure that the customer sequence does not contain sequence S. If the final resultant bit string is not zero, then we increase the number of customer sequences which contain the sequence S.

Step 2. Determine if the sequence S is a frequent sequence

The support for the sequence S can be obtained by dividing the number of total customer sequences from the number of customer sequences which contain the sequence S. If the support of the sequence S is no less than the minimum support threshold, then sequence S is a frequent sequence.

3.2 Query Processing for Type II Query

For Type II query, there is the notation "*" specified in the **With** clause. For example, in Query 1, the user would like to find all the sequential patterns which contain the sequence {E}{A}{B} from the customer sequence database (Table 1) and the minimum support threshold is set to 40%.

Query 1:
Mining <*Sequential Patterns*>
From <*CSD*>
With <*,{E},*,{A},*,{B},*>
support <*40%*>

Suppose the user specifies a sequence which contains m itemsets D_1, D_2, …and D_m in the **With** clause and S = $\{D_1\}\{D_2\}\ldots\{D_m\}$. We divide the algorithm for this type of queries into two steps: the first step is to find (m+1)-frequent sequences which contains sequence S, and the second step is to find all the q-frequent sequences .$q \geq m+2$. which contains sequence S. In the following, we describe the two steps:

Step 1. Find all the frequent (m+1)-sequences.

Step 1.1. Scan the bit-string database, if all items in S are contained in a record, then output the items in this record and the bit string for each item into *1-itemset database*. If S is a frequent sequence, then find all 1-frequent itemsets. The frequent itemsets are found in each iteration. For the *k*th iteration (k≥1), the candidate (k+1)-itemsets are generated, and scan the *(k+1)-itemset database* to find (k+1)-frequent itemsets.

The method to generate the candidate (k+1)-itemsets is described as follows [1]: For every two k-frequent itemsets A=$\{a_1, \ldots, a_{k-1}, r\}$ and B= $\{a_1, \ldots, a_{k-1}, t\}$, the candidate ($k$+1)-itemset $\{a_1, \ldots, a_{k-1}, r, t\}$ can be generated. For each record in the *k-itemset database*, we use the *k*-frequent itemsets in this record and apply the above method to generate candidate (k+1)-itemsets. Suppose the two frequent *k*-itemsets X and Y in a record generate candidate (k+1)-itemset Z. We perform AND operation on the two bit strings for the two frequent k-itemsets X and Y, and the resultant bit string is the bit string for the candidate (k+1)-itemset Z. If this bit string is not zero, then output the candidate (k+1)-itemset Z and its bit string into (k+1)-itemset database. Besides, we also output the frequent *k*-itemsets and its bit string in each record into the *frequent itemset database*.

Table 3. 1-itemset database

CID	Items	Bit string for each item
4	A, B, C, D, E, F	010110,101010,001000,100000,010110,000001
5	A, B, C, D, E, F	000100,000010,001000,110111,010000,010001

Table 4. 2-itemset database

CID	2-itemsets	Bit string for each 2-itemset
4	{AB},{AE},{BC},{BD},{BE}	000010,010110,001000100000,000010
5	{AD},{BD},{DE},{DF},{EF}	000100,000010,010000,010001,010000

For example, in Table 2, the records which contain the sequence {E}{A}{B} in the **With** clause in Query 1 are CID 4 and CID 5, Hence, the 1-itemset database can be generated, which is shown in Table 3. Then, the 1-itemset database is scanned to generate frequent 2-itemsets and 2-itemset database. The 2-itemset database is shown in Table 4. Finally, we can generate the frequent itemsets {A}, {B}, {C}, {D}, {E}, {F} and {B, D}, and the frequent database which is shown in Table 5.

Table 5. Frequent itemset database

CID	Frequent itemsets	Bit string for each frequent itemset

4	{A},{B},{C},{D}
	{E},{F},{BD}
		010110,101010,001000,100000,
		010110,000001,100000

5	{A},{B},{C},{D}
	{E},{F},{BD}
		000100,000010,001000,110111,
		010000,010001,000010

Step 1.2. Each frequent itemset (i.e., frequent 1-sequence) is given a unique number, and replace the frequent itemsets in the frequent itemset database with their numbers to form a *1-sequence database*.

For example, the numbers for the frequent itemsets {A}, {B}, {C}, {D}, {E}, {F} and {B,D} are 1, 2, 3, 4, 5, 6, and 7, respectively, and the 1-sequence database is shown in Table 6, which is generated from Table 5.

Table 6. 1-sequence database

CID	1-sequence	Bit string for each 1-sequence
4	1, 2, 3, 4, 5, 6, 7	010110, 101010, 001000, 100000, 010110, 000001, 100000
5	1, 2, 3, 4, 5, 6, 7	000100, 000010, 001000, 110111, 010000,010001,000010

Step 1.3. Generate candidate 2-sequences, and scan 1-sequence database to generate *2-sequence database* and find all the frequent 2-sequences.

The candidate 2-itemsets are generated as follows: For each frequent 1-sequence f except D_1, the itemset D_1 is combined with the frequent 1-sequence to generate a candidate 2-sequence. If there is a notation "*" appears before the itemset D_1 in the **With** clause, then the candidate 2-sequence $\{f\}\{D_1\}$ is generated. If the notation "*" appears after the itemset D_1, then the candidate 2-sequence $\{D_1\}\{f\}$ is generated. If the reverse order of a candidate 2-sequence is contained in the specified sequence S, then this candidate 2-sequence can be pruned.

For each record in the 1-sequence database, we use the frequent 1-sequences in the record and apply the above method to generate candidate 2-sequences. Suppose that the two frequent 1-sequences X and Y in a record generate candidate 2-sequence Z. We perform the sequential bit-string operation on the two bit strings for the two frequent 1-sequences X and Y, and the resultant bit string is the bit string for the candidate 2-sequence Z. If this bit string is not zero, then output the candidate 2-sequence Z and its bit string into 2-sequence database. After scanning 1-sequence database, the 2-sequence database can be generated and the candidate 2-sequences can be counted. If the support for a candidate 2-sequence is no less than the minimum support threshold, then the candidate 2-sequence is a frequent 2-sequence.

For example, in Query 1, the first itemset specified in the **With** clause is {E} whose number is 5, and there are notation "*"s which appear before and after the itemset {E}. Hence, the generated candidate 2-sequences are {1}{5}, {5}{1}, {2}{5}, {5}{2}, {3}{5}, {5}{3}, {4}{5}, {5}{4}, {6}{5}, {5}{6}, {7}{5}, and {5}{7}. From these candidate 2-sequences, {1}{5} and {2}{5} can be pruned, because the reverse order of the two sequences are contained in the specified sequence {5}{1}{2}. After scanning 1-sequence database (Table 6), the generated 2-sequence database is shown in Table 7, and the frequent 2-sequences are {5}{1},{5}{2}, {4}{5}, {5}{3} and {5}{6}.

Table 7. 2-sequence database

CID	2-sequence	Bit string for each 2-sequence
4	{3}{5},{4}{5},{7}{5},{5}{1},{5}{2}, {5}{3},{5}{6}	000110, 010110, 010110, 000110, 001010, 001000, 000001
5	{4}{5},{5}{1},{5}{2},{5}{3},{5}{4}, {5}{6},{5}{7}	010000, 000100, 000010, 001000, 000111, 000001, 000010

Step 1.4. Generate candidate 3-sequences, and scan 2-sequence database to generate 3-sequence database and find all the frequent 2-sequences.

The method to generate candidate 3-sequences is described as follows: For every two frequent 2-sequences S_1={D_1}{r} which is a sub-sequence of S and S_2={D_1}{t} (or S_1={D_1}{r} and S_2={t}{D_1}), we can generate the candidate 3-sequences {D_1}{r}{t} and {D_1}{t}{r}.or {t}{D_1}{r}.

For the above example, the generated candidate 3-sequences are {5}{1}{2}, {4}{5}{1}, {4}{5}{2}, {5}{3}{1}, {5}{1}{3}, {5}{3}{2}, {5}{2}{3}, {5}{6}{1}, {5}{1}{6}, {5}{6}{2} and {5}{2}{6}. After scanning each record in 2-sequence database (Table 7), the candidate 3-sequences in each record can be generated, and the candidate 3-sequences can be counted. Finally, the generated frequent 3-sequences are {5}{1}{2}, {4}{5}{1}, {4}{5}{2}, {5}{3}{1}, {5}{3}{2}, {5}{1}{6} and {5}{2}{6}.

Step 1.5. Frequent (h+1)-sequences (3≤h≤m) are generated in each iteration. For the (h-2)th iteration, we use frequent h-sequences to generate candidate (h+1)-sequence, and scan h-sequence database to generate (h+1)-sequence database, and find all the frequent (h+1)-sequences.

We use the following method to generate candidate (h+1)-sequences: For any two frequent h-sequence S_1={s_1}{s_2}…{s_{h-1}}{r} and S_2={s_1}{s_2}…{s_{h-1}}{t}, in which {s_1}{s_2}…{s_{h-1}} is a sub-sequence of S or {r} and {t} are contained in S, the candidate (h+1)-sequences {s_1}{s_2}…{s_{h-1}}{r}{t} and {s_1}{s_2}…{s_{h-1}}{t}{r} can be generated. If a generated candidate {h+1}-sequence contains more than one itemsets which are not contained in S, then the candidate {h+1}-sequence can be pruned.

For each record in the h-sequence database, we use the frequent h-sequences in this record and the above method to generate candidate (h+1)-sequences, and perform the sequential bit-string operation on the two bit strings for the two frequent h-sequences which generate the candidate (h+1)-sequence. The resultant bit string is the bit string for the candidate (h+1)-sequence. If the resultant bit string is not zero, then output the candidate (h+1)-sequence and its bit string into (h+1)-sequence database, and count the support for the candidate (h+1)-sequence. After scanning the h-sequence database, the (h+1)-sequence database can be generated and the supports for the candidate (h+1)-sequences can be computed. If the support for a candidate (h+1)-sequence is no less than the minimum support, then the candidate (h+1)-sequence is a frequent (h+1)-sequences. If there are frequent (m+1)-sequences generated, then step 2 need to be performed. Otherwise, step 3 is performed directly.

For the above example, according to step 1.5, the generated candidate 4-sequences are {4}{5}{1}{2}, {5}{3}{1}{2}, {5}{1}{6}{2} and {5}{1}{2}{6}. After scanning 3-sequence database, the generated frequent 4-sequences are {4}{5}{1}{2}, {5}{3}{1}{2} and {5}{1}{2}{6}. Because there are frequent 4-sequences generated, we need to perform the next step.

Step 2. The frequent (m+n+1)-sequences (n≥1) which contain the specified sequence S are generated in each iteration. For the nth iteration, we use the frequent (m+n)-sequences to generate candidate (m+n+1)-sequences and scan the (m+n)-sequence database and 1-sequence database to generate (m+n+1)-sequence database in which the candidate (m+n+1)-sequences are contained in each record but the bit string are not, and find the frequent (m+n+1)-sequences.

The method to generate candidate (m+n+1)-sequences is as follows: For every two frequent (m+n)-sequences $S_1 = \{s_1\}\{s_2\}\ldots\{s_i\}\{r\}\{s_{i+1}\}\ldots\{s_{m+n-1}\}$ and $S_2 = \{s_1\}\{s_2\}\ldots\{s_j\}\{t\}\{s_{j+1}\}\ldots\{s_{m+n-1}\}$ ($i \le j$), in which {r} is not contained in S_2 and {t} is not contained in S_1, a candidate (m+n+1)-sequence $\{s_1\}\{s_2\}\ldots\{r\}\ldots\{t\}\ldots\{s_{m+n-1}\}$ can be generated. For each record in (m+n)-sequence database, we also use every two frequent (m+n)-sequences in this record and apply the above method to generate a candidate (m+n+1)-sequence, and perform the sequential bit-string operations on the bit strings for the itemsets in the candidate (m+n+1)-sequence by scanning the 1-sequence database. If the resultant bit string is not zero, then output the candidate (m+n+1)-sequence into the (m+n+1)-sequence database, and count the support for the candidate (m+n+1)-sequence. After scanning (m+n)-sequence database, the (m+n+1)-sequence database can be generated and the frequent (m+n+1)-sequences can be found.

For the above example, according to step 2, the generated candidate 5-sequences are {4}{5}{3}{1}{2}, {4}{5}{1}{2}{6} and {5}{3}{1}{2}{6}. For each record in the 4-sequence database, we use the frequent 4-sequences in this record to generate candidate 5-sequences for this record, and perform the sequential bit-string operations on the bit strings for the itemsets in the candidate 5-sequence to generate the bit string for the candidate 5-sequence. If the resultant bit string is not zero, then count the support for the candidate 5-sequence. After scanning the 4-sequence database, the generated frequent 5-sequences are {4}{5}{3}{1}{2}, {4}{5}{1}{2}{6} and

{5}{3}{1}{2}{6}. These frequent 5-sequences can further generate candidate 6-sequence {4}{5}{3}{1}{2}{6}. After scanning 5-sequence database, the generated frequent 6-sequence is also {4}{5}{3}{1}{2}{6}, and there is no candidate 7-sequence generated. Hence, the algorithm for mining frequent sequences terminates.

Step 3. For each frequent sequence, the code for each itemset in the frequent sequence is replaced with the itemset itself. If a frequent sequence is not contained in another frequent sequences, then this frequent sequence is a sequential pattern.

For the above example, the frequent sequences which satisfy the user requirement in Query 1 are {E}{A}{B}, {D}{E}{A}{B}, {E}{C}{A}{B}, {E}{A}{B}{F}, {D}{E}{C}{A}{B}, {D}{E}{A}{B}{F}, {E}{C}{A}{B}{F} and {D}{E}{C}{A}{B}{F}, and the sequential pattern is {D}{E}{C}{A}{B}{F}.

4 Experimental Result

The synthetic database of sales transactions is generated to evaluate the performance of our algorithm. The method to generate synthetic transactions is similar to the one used in [2]. The parameters used in our experiments are shown in Table 8.

First, we generate four synthetic transaction databases C5-T10-I10-R10, C10-T10-I10-R10, C20-T10-I10-R10 and C10-T10-I10-R20, in which C is the number of customers (in 000's), D is the number of transactions (in 000's), |T| is the average number of the transactions for each customer, |I| is the average size of the transactions, and R is the percentage of the repeated items between every two neighbor transactions. We also generated 3 general cases of TypeII queries. Figure 1 shows the relative execution time for PrefixSpan [4] and our algorithm for a generated Type II query. Figure 2 shows the relative execution time for the generated three queries in the database C20-T10-I10-R10 for different minimum support thresholds.

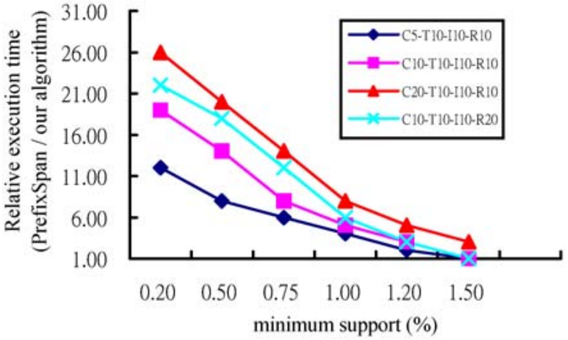


Fig. 1. Relative execution time

The experimental results show that our algorithm outperforms PrefixSpan algorithm, and the performance gap increases as the minimum support threshold decreases because when the minimum support decreases, the number of the frequent sequences increases, the number of the projected databases increases and the size of each projected database also increases, such that the performance is degraded for PrefixSpan algorithm. Besides, PrefixSpan algorithm needs to take extra time to pick the frequent itemsets from the large amount of frequent itemsets to match the user queries.

However, for our algorithm, we only focus on the items specified in user queries, that is, there is no redundant frequent sequence can be generated. Hence, our algorithm can significantly outperform PrefixSpan algorithm.

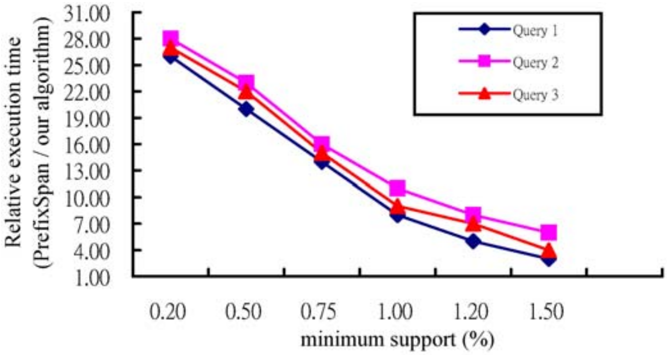


Fig. 2. Relative execution time for database C20-T10-I10-R10

Achknowledgement

Research on this paper was partially supported by National Science Council grant NSC92-2213-E-030-019 and NSC92-2213-E-130-007.

References

1. Agrawal, R. and *et a*l.: Fast Algorithm for Mining Association Rules. Proceedings of International Conference on Very Large Data Bases (VLDB), (1994) 487-499
2. Agrawal, R. and *et al.*: Mining Sequential Patterns. Proceedings of the International Conference on Data Engineering (ICDE), (1995) 3-14
3. R. Agrawal, R. Srikant: Mining Sequential Patterns: Generalizations and Performance Improvements", International Conference on Extending Database Technology (EDBT), (1996) 3-17
4. Pei, J., Han, J. and *et al.*: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. Proceedings of the International Conference on Data Engineering (ICDE), Heidelberg, Germany, (2001) April
5. Yen, S.J. and Lee, Y.S.: An Efficient Data Mining Technique for Discovering Interesting Sequential Patterns. Proceedings of the International Conference on Data Mining (ICDM), (2001) 663-664
6. Yen, S.J. and Lee, Y.S.: Mining Interesting Association Rules: A Data Mining Language, Lecture Notes in Artificial Intelligence, Vol. 2336, (2002) 172-176

Mining Borders of the Difference of Two Datacubes

Alain Casali

Laboratoire d'Informatique Fondamentale de Marseille (LIF)
CNRS UMR 6166, Université de la Méditerranée
Case 901, 163 Avenue de Luminy, 13288 Marseille Cedex 9, France
casali@lif.univ-mrs.fr

Abstract. In this paper we use the novel concept of minimal cube transversals on the cube lattice of a categorical database relation for mining the borders of the difference of two datacubes. The problem of finding cube transversals is a sub-problem of hypergraph transversal discovery since there exists an order-embedding from the cube lattice to the power set lattice of binary attributes. Based on this result, we propose a levelwise algorithm and an optimization which uses the frequency of the disjunction for mining minimal cube transversals. Using cube transversals, we introduce a new OLAP functionality: discovering the difference of two uni-compatible datacubes or the most frequent elements in the difference. Finally we propose a merging algorithm for mining the boundary sets of the difference without computing the two related datacubes. Provided with such a difference of two datacubes capturing similar informations but computed at different dates, a user can focus on what is new or more generally on how evolve the previously observed trends.

1 Introduction and Motivation

Hypergraph transversals [8, 10] have various applications in binary data mining and various kinds of knowledge can be discovered: minimal keys and minimal functional dependencies [13], connection between positive and negative borders of theories [14]. When mining minimal transversals, we show in [4] that the power set lattice is not really suitable when addressing multidimensional data mining problems, and suggest, as an alternative, an algebraic structure which is called cube lattice of a categorical database relation. Cube lattice is a set of tuples representing multidimensional patterns, provided with a generalization order between tuples. A similar lattice has been independently proposed by Lakshmanan *et al.* [12]. Based on this structure, the authors define the quotient cube lattice, a succinct summary of the datacube, preserving the Rollup/Drilldown semantics of cube.

Cube lattice provides a sound basis and a graded search space for extracting semantics from the datacube such as Roll-Up dependencies [3], multidimensional associations [5], iceberg cubes [2, 11], concise representation of hight frequency multidimensional patterns [6] and reduced cubes [12, 5]. In this paper, following

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 391–400, 2004.

from this semantics trend, we use the concept of cube transversals [5] in order to compute the borders of the difference of two uni-compatible datacubes. More precisely, we make the following contributions.

- We propose an optimization for mining minimal cube transversals based on the Bonferroni inequalities and a levelwise algorithm which enforces the optimization.
- We formally define borders of the difference. Computing the difference of two uni-compatible datacubes is closely related to the discovery of emerging patterns originally proposed by [7] in the power set lattice framework. Our characterization and algorithm differ from the approach of emerging patterns since they are based on minimal cube transversals and fit in the cube lattice framework.

Because data warehouses are regularly refreshed, it is specially interesting to observe how evolves the mined knowledge: are great trends (for instance in consumer behavior) still increased or are they decreasing? Provided with two versions of a datacube computed at different dates, a user can answer these question very easily.

The remainder of the paper is organized as follows. Cube lattice framework [4] is described in section 2 as a graded search space for multidimensional data mining. In section 3, we recall the concept of cube transversal and a merging algorithm for mining minimal relation transversals from categorical database relations. In section 4, we propose an application using the minimal cube transversals: the "difference of two uni-compatible datacubes" and characterize its borders.

2 Background: The Cube Lattice Framework [4]

Throughout the paper, we make the following assumptions and use the introduced notations. Let r be a relation over the schema $\mathcal{R}$. Attributes of $\mathcal{R}$ are divided in two sets (i) $\mathcal{D}$ the set of dimensions, also called categorical or nominal attributes, which correspond to analysis criteria for OLAP, classification or concept learning and (ii) $\mathcal{M}$ the set of measures (for OLAP) or class attributes. Moreover, the set of attributes called $\mathcal{D}$ is totally ordered (the underlying order is denoted by $<_{\mathcal{D}}$) and for each attribut $A \in \mathcal{D}, Dim(A)$ stands for the projection of r over A.

The multidimensional space of the categorical database relation r groups all the valid combinations built up by considering the value sets of attributes in $\mathcal{D}$, which are enriched with the symbolic value ALL. The latter, introduced in [9] when defining the Cube-By operator, is a generalization of all the possible values of any dimension ($\forall A \in \mathcal{D}, \forall a \in Dim(A), \{a\} \subset \text{ALL}$).

The multidimensional space of r is noted and defined as follows: $Space(r) = \{\times_{A \in \mathcal{D}}(Dim(A) \cup \text{ALL})\} \cup \{<\emptyset, \ldots, \emptyset>\}$ where × symbolizes the Cartesian product, and $<\emptyset, \ldots, \emptyset>$ stands for the combination of empty values. Any combination belonging to the multidimensional space is a tuple and represents a multidimensional pattern.

Example 1. - Table 1 presents the categorical database relation used all along the paper to illustrate the introduced concepts. In this relation, Sky, AirTemp and Humidity are dimensions and EnjoySport is a class. The following tuples $t_1 = <S, W, \text{ALL}>$, $t_2 = <S, W, N>$, $t_3 = <S, C, N>$, $t_4 = <S, \text{ALL}, N>$, $t_5 = <S, \text{ALL}, \text{ALL}>$, $t_6 = <\text{ALL}, W, \text{ALL}>$ and $t_7 = <\text{ALL}, W, H>$ are elements of $Space(r)$.

Table 1. Relation example r

Sky	AirTemp	Humidity	EnjoySport
S	W	N	Yes
S	W	H	Yes
R	C	H	No

The multidimensional space of r is structured by the generalization order between tuples which is denoted by $\geq_g$. If $u \geq_g v$, we say that u is more general than v in $Space(r)$. In the multidimensional space of our relation example, we have: $t_5 \geq_g t_2$, i.e. t_5 is more general than t_2 and t_2 is more specific than t_5. Moreover any tuple generalizes the tuple $<\emptyset, \emptyset, \emptyset>$ and specializes the tuple $<\text{ALL}, \text{ALL}, \text{ALL}>$. When applied to a set of tuples, the operators *min* and *max* yield the tuples which are the most general ones in the set or the most specific ones respectively.

The two basic operators provided for tuple construction are: Sum (denoted by $+$) and Product (noted $\bullet$). The Semi-Product operator (noted $\odot$) is a constrained product used for the candidate generation in levelwise algorithms.

- The Sum of two tuples yields the most specific tuple which generalizes the two operands. If $t = u + v$, then we say that t is the Sum of the tuples u and v. In our example of $Space(r)$, we have $t_2 + t_3 = t_4$. This means that t_4 is built up from (or aggregates data of) the tuples t_2 and t_3.
- The Product of two tuples yields the most general tuple which specializes the two operands. If it exists, for these two tuples, a dimension A having distinct and real world values (i.e. existing in the original relation), then the only tuple specializing them is the tuple $<\emptyset, \ldots, \emptyset>$ (apart from it, the tuple sets which can be used to construct them are disjoined). If $t = u \bullet v$, then we say that t is the Product of the tuples u and v. In our example of $Space(r)$, we have $t_1 \bullet\ t_4 = t_2$. This means that t_1 and t_4 generalize t_2 and t_2 participates to the construction of t_1 and t_4 (directly or not). The tuples t_1 and t_3 have no common point apart from the tuple of empty values.

In some cases, it is interesting to know the attributes for which the values associated with a tuple t are different from the value ALL. This is why the function *Attribute* is introduced. In our example, we have $Attribute(t_1) = \{\text{Sky}, \text{AirTemp}\}$.

The Semi-Product operator is a constrained product operator useful for candidate generation in a levelwise approach [14]. Provided with multidimensional patterns at the level i, we only generate candidates of level $i+1$, if they exist (else the constrained product yields $<\emptyset, \dots, \emptyset>$). Moreover, each tuple is generated only once. In our example, we have $t_5 \odot t_6 = t_1$ and $t_5 \odot t_7 = <\emptyset, \dots, \emptyset>$.

By providing the multidimensional space of r with the generalization order between tuples and using the above-defined operators Sum and Product, we define an algebraic structure which is called cube lattice. Such a structure provides a sound foundation for multidimensional data mining issues.

Theorem 1. *[4] - Let r be a categorical database relation over $\mathcal{D} \cup \mathcal{M}$. The ordered set $CL(r) = \langle Space(r), \geq_g \rangle$ is a complete, graded, atomistic and coatomistic lattice, called cube lattice in which Meet ($\bigwedge$) and Join ($\bigvee$) operators are given by:*

1. $\forall\ T \subseteq CL(r),\ \ \bigwedge T = +_{t \in T}\ t$
2. $\forall\ T \subseteq CL(r),\ \ \bigvee T = \bullet_{t \in T}\ t$

Through the following proposition, we characterize the order-embedding from the cube lattice to the powerset lattice of the whole set of attribute values. For avoiding ambiguities, we choose to prefix each value by the name of the concerned attribute.

Proposition 1 *- Let $\mathcal{L}(r)$ be the powerset lattice of the attribute value set, i.e. the lattice $\langle \mathcal{P}(\bigcup_{A \in \mathcal{D}} A.a, \forall a \in Dim(A)), \subseteq \rangle$ [1]. Then it exists an order-embedding:*

$$\Phi : CL(r) \rightarrow \mathcal{L}(r)$$

$$t \mapsto \begin{cases} \bigcup_{A \in \mathcal{D}} A.a, \forall a \in Dim(A) \ \textit{if}\ t = <\emptyset, \dots, \emptyset> \\ \{A.t[A] \mid \forall A \in Attribute(t)\} \ \textit{elsewhere.} \end{cases}$$

Consequently, we can associate to each categorical relation r a binary relation $\Phi(r) = \{\Phi(t), \forall\ t \in r\}$. The latter can be used to apply binary data mining technics. Let us underline that Φ is not the single order-embedding Φ. We can construct other order-embedding by associating to each value a_i in the dimension of the attribute A a single binary attribute in the power set lattice of the attribute value set.

Finally, we have to highlight two important elements in the cube lattice: its atoms and coatoms. An atom of the cube lattice is a concept similar to the one-itemsets in the power set lattice (one-itemset are the atoms of the power set lattice). We denote by $\mathcal{A}t(CL(r))$ the atoms of the cube lattice (i.e. $\{t \in CL(r) : |\Phi(t)| = 1\}$). The atoms of a tuple t, denoted by $\mathcal{A}t(t)$, constitute the set of the atoms of the cube lattice which are more general than t (i.e. $\{t' \in \mathcal{A}t(CL(r)) : t' \geq_g t\}$). The coatoms of the cube lattice, denoted by $\mathcal{CA}t(CL(r))$, represent the concept dual to the one of atoms of the cube lattice (i.e. $\{t \in CL(r) : |\Phi(t)| = |\mathcal{D}|\}$).

[1] $\mathcal{P}(X)$ is the powerset of X.

3 Cube Transversals

When considering the power set lattice $\mathcal{L}(r)$ as the search space, a binary pattern X is a transversal of $\Phi(r)$ if and only if each transaction t of $\Phi(r)$ contains at least one value of X (X is a transversal of $\overline{\Phi(r)}$ if X is not a subset of any transactions t of $\Phi(r)$ respectively). For example, using our relation example and the order-embedding Φ, the binary pattern $\{Sky.S, AirTemp.W\}$ is a transversal of $\Phi(r)$. Moreover, this pattern is a minimal transversal of $\Phi(r)$ because none of its subsets is transversal of $\Phi(r)$. Unfortunately, in the binary framework, invalid multidimensional patterns are computed. For example, the binary pattern $\{Sky.S, Sky.R\}$ is a minimal transversal of $\Phi(r)$ but it is not a valid multidimensional pattern because each value of this pattern belongs to the very same attribute Sky. And we know that a multidimensional pattern can only encompasses a single value for any given attribute. This is why we have introduced [5] the concept of minimal cube transversal:

Definition 1. ***(Cube Transversal)*** *- We define the relation* $\overline{r}$ *as the difference of the coatoms of the cube lattice and the relation* r *(*$\overline{r} = \mathcal{CA}t(CL(r))\backslash r$*). Let* $t \in CL(r)$ *be a tuple,* t *is a cube transversal of* r *over* $CL(r)$ *iff* $\forall t' \in r, t + t' \neq <ALL, \ldots, ALL>$*.* t *is a cube transversal of* $\overline{r}$ *iff* $\forall t' \in r, t \not\geq_g t'$*.*

Lemma 1. *Since the contraint "t is a cube transversal of* r *(or* $\overline{r}$*)" is a monotone constraint on the cube lattice, the set of cube transversals is a convex space and thus it can be represented by its minimal border [4]. The set of minimal cube transversals of* r *and of* $\overline{r}$ *are denoted by* $cTr(r)$ *and* $cTr(\overline{r})$ *respectively and defined as follows:*

- $cTr(r) = \min_{\geq_g}(\{t \in CL(r) \mid \forall t' \in r, t + t' \neq <ALL, \ldots, ALL>\})$
- $cTr(\overline{r}) = \min_{\geq_g}(\{t \in CL(r) \mid \forall t' \in r, t \not\geq_g t'\})$

Due to the convex space property, an unseen tuple t *is a transversal of* r *(or* $\overline{r}$ *respectively) if it exists at least a tuple* t' *in* $cTr(r)$ *(*$cTr(\overline{r})$ *resp.) which generalizes* t*. Let* $\mathbb{A}$ *be an anti-chain of* $CL(r)$ *(all tuples of* $\mathbb{A}$ *are not comparable using* $\geq_g$*). We can constrain the set of minimal cube transversals of* r *using* $\mathbb{A}$ *by enforcing each minimal cube transversal* t *to be more general than at least one tuple* u *of the anti-chain* $\mathbb{A}$*. The new related definitions are the following:*

- $cTr(r, \mathbb{A}) = \{t \in cTr(r) \mid \exists u \in \mathbb{A} : t \geq_g u\}$
- $cTr(\overline{r}, \mathbb{A}) = \{t \in cTr(\overline{r}) \mid \exists u \in \mathbb{A} : t \geq_g u\}$

Example 2. By considering the multidimensional space of the relation example, the set of minimal cube transversals of r is $\{<S, C, \text{ALL}>, <S, \text{ALL}, H>, < R, W, \text{ALL} >, <\text{ALL}, W, H>\}$ and the set of minimal cube transversals of $\overline{r}$ is $\{<S, C, \text{ALL}>, <R, W, \text{ALL}>, <R, \text{ALL}, N>, <\text{ALL}, C, N>\}$. If we constrain the set $cTr(r)$ with the anti-chain composed by the single tuple $<S, C, H>$, we obtain $cTr(r, <S, C, H>) = \{<S, C, \text{ALL}>, <S, \text{ALL}, H>\}$.

3.1 Optimizing the Discovery of Minimal Cube Transversals

Assessing if a tuple t is whether a cube transversal of r or $\overline{r}$ can require $|r|$ evaluations. Minimizing the number of evaluations improves, in practice, the performance of the algorithm CTR. We introduce a new optimization based on the Bonferroni inequalities which are applied within the cube lattice framework. When the sum of the frequency of atoms of t is lower than 1, no condition is evaluated because t cannot be a cube transversal of r. A dual property is given for testing if t is a minimal cube transversal of $\overline{r}$.

Definition 2. ***(Frequency)*** *- Let $t \in CL(r)$ be a tuple, the frequency of t (denoted by $Freq(t,r)$) is the ratio between the number of tuples of r which are more specific than t and the number of tuples of r. Thus we have:*

$$Freq(t,r) = \frac{|\{t' \in r \mid t \geq_g t'\}|}{|r|}$$

Proposition 2 *Let $t \in CL(r)$ be a tuple, if t is a cube transversal of r then $\sum\limits_{t' \in \mathcal{A}t(t)} Freq(t',r) \geq 1$ and if t is a cube transversal of $\overline{r}$ then $\sum\limits_{t' \in \mathcal{A}t(t)} (1 - Freq(t',r)) \geq 1$.*

3.2 Finding Minimal Cube Transversals

In [10], it is shown that levelwise mining of minimal hypergraph transversals improves complexity results proposed by [8]. Using the cube lattice framework, we propose a levelwise algorithm called CTR (Cube TRansversal) algorithm which computes minimal cube transversals. The candidate generation step does not need any backtrack because we update the set cTr at each level and we use it for the pruning step. We improve our algorithm by a frequency-based optimization. A levelwise approach works very well when the underlying search space is a graded lattice, which is case of the cube lattice.

For mining minimal cube transversals of $\overline{r}$, we must replace conditions at lines 9 and 13 by $\sum\limits_{t' \in \mathcal{A}t(t)} (1 - Freq(t',r)) \geq 1$ and $l \geq_g t$ respectively.

Complexity of CTR: The complexity of a levelwise algorithm for finding minimal transversals of an hypergraph $\mathcal{H}$ on a set E is $\mathcal{O}(2^k|E||Tr(\mathcal{H})|)$ where $k = \max(\{|X| : X \in \mathcal{H}\})$ [10]. Using the cube lattice as the search space, this complexity is preserved with $E = \bigcup\limits_{A \in \mathcal{D}} A.a, \forall a Dim(A)$, $k = |\mathcal{D}|$ and $|cTr(r)| < |Tr(\mathcal{H})|$ since Φ is an order-embedding but not an order isomorphism.

4 Diff_Cube Operator

In ROLAP databases, the relational operator difference is fundamental for analyzing changes betwwen two uni-compatible datacubes (computed at different

Algorithm 1 CTR Algorithm

```
Input: Categorical database relation $r$ over $\mathcal{D}$ [and an anti-chain $\mathbb{A}$]
Output: $cTr(r[, \mathbb{A}])$
$i := 1$; $cTr := \{\emptyset\}$
$C_1 := \{t \in \mathcal{A}t(CL(r))\}$
$L_1 := \{t \in C_1 \mid t$ is a cube transversal $\}$
$C_1 := C_1 \backslash L_1$
while $C_i \neq \emptyset$ do
  $cTr := cTr \cup L_i$
  $C_{i+1} := \{v = t \odot t' \mid t, t' \in C_i, v \neq <\emptyset, \ldots, \emptyset>, \nexists u \in cTr : u \geq_g v$ [ and $\exists u \in \mathbb{A} : v \geq_g u$] $\}$
  for all $t \in C_{i+1}$ do
    if $\sum_{t' \in \mathcal{A}t(t)} Freq(t', r) < 1$ then $C^*_{i+1} := C^*_{i+1} \cup t$; $C_{i+1} := C_{i+1} \backslash t$
  end for
  for all $t \in r$ do
    for all unmarked $l \in C_{i+1}$ do
      if $l + t = <$ALL$, \ldots,$ ALL$>$ then mark $l$
    end for
  end for
  $L_{i+1} := \{l \in C_{i+1} \mid l$ is unmarked $\}$; $C_{i+1} := (C_{i+1} \backslash L_{i+1}) \cup C^*_{i+1}$
  $i := i + 1$
end while
return $cTr$
```

instants by using the function COUNT) of two categorical database relations (denoted by r^+ and r^- for uniformity). However, the difference of the two datacubes is not the datacube of the difference of the two relations (datacube(r^+)$\backslash$ datacube(r^-) $\neq$ datacube($r^+ \backslash r^-$)). We propose a merging algorithm which computes the boundary sets of tuples which are in the difference. Such tuples provide a condensed representation of (*i*) the difference of the two related datacubes or (*ii*) the most frequent elements in the difference.

4.1 Emerging Tuples

Let $minfreq \in]0, 1]$ a threshold given by the user, a tuple t is emerging if and only if it satisfies the two following constraints (C_1 and C_2):

- (C_1) $Freq(t, r^-) = 0$ and
- (C_2) $Freq(t, r^+) \geq minfreq$

The set $Diff\_Cube(r^+, r^-)$ encompasses all the emerging tuples of $r = r^+ \cup r^-$; *i.e.* $Diff\_Cube(r^+, r^-) = \{t \in CL(r) \mid t$ is emerging$\}$. Thus $Diff\_Cube(r^+, r^-)$ is exactly the difference of the two datacubes of the relations r^+ and r^-.

In RDBMSs provided with OLAP functionalities, such as IBM DB2 or Microsoft SQL Server, $Diff\_Cube(r^+, r^-)$ can be computed by using the Group By Cube operator [9] as follows:

SELECT A_1, ..., A_n
    FROM r^+
    GROUP BY CUBE (A_1, ..., A_n)
    HAVING count(*) $\geq minfreq * |r^+|$
MINUS
SELECT A_1, ..., A_n
    FROM r^-
    GROUP BY CUBE A_1, ..., A_n;

We assume that $|r^+|$ and $minfreq$ are thresholds given by the user. Due to the underlying Cube-By operations, it is obvious that this query is specially time and space consuming, thus avoiding to evaluated it and instead computing the borders of its result can be of great interest.

4.2 Finding Borders

The constraint C_1 (C_2 resp.) is monotone (antimonotone resp.) w.r.t. $\geq_g$, consequently $Diff\_Cube(r^+, r^-)$ is a convex space of $CL(r)$ [4] and thus it can be represented by the sets (borders) $S = \max_{\geq_g}(Diff\_Cube(r^+, r^-))$ and $G = \min_{\geq_g}(Diff\_Cube(r^+, r^-))$.

The following proposition characterizes the borders of the Diff_Cube set:

Proposition 3 *Let $Diff\_Cube(r^+, r^-)$ be the set of emerging tuples of a categorical database relation $r = r^+ \cup r^-$ and $M = \max_{\geq_g}(\{t \in CL(r^+) \mid Freq(t, r^+) \geq minfreq\}$ then:*

1. *$G = \{t \in cTr(\overline{r^-})$ on $CL(r^+) \mid Freq(t, r^+) \geq minfreq\}$*
2. *$S = \{t \in M \mid \exists t' \in G : t' \geq_g t\}$*

With the following propositions, the borders S and G of $Diff\_Cube(r^+, r^-)$ can be computed in an efficient way and a merging algorithm can be designed to enforce such a computation.

Proposition 4 *Let $Diff\_Cube(r^+, r^-)$ be the difference of the datacube of r^+ and the one of r^- and $M = \max_{\geq_g}(\{t \in CL(r^+) \mid Freq(t, r^+) \geq minfreq\}$ thus: $\forall t \in cTr(r^+), t \in G \Rightarrow \nexists u \in M \backslash S : t \geq_g u$.*

Proposition 5 *Let $r = r_1 \cup r_2$ be a categorical database relation, $cTr(r_1)$ and $cTr(r_2)$ the minimal cube transversals of r_1 and r_2 respectively. If $cTr(r_1)$ is empty, we consider that $cTr(r_1) = \{<\emptyset, \ldots, \emptyset>\}$ (idem for $cTr(r_2)$), thus we have: $cTr(r) = \min_{\geq_g}(\{t \bullet t' \neq <\emptyset, \ldots, \emptyset> \mid t \in cTr(r_1)$ and $t' \in cTr(r_2)\})$.*

For finding S and G in our new context, we propose the merging algorithm Diff_Cube. Our algorithm includes the function Max_Set_Algorithm which discovers maximal frequent multidimensional patterns. It could be enforced by modifying algorithms such as Max-Miner [1]. The correctness of Diff_Cube algorithm is given by propositions 3, 4 and 5. Its complexity is similar to the complexity of the CTR algorithm.

Algorithm 2 Algorithm Diff_Cube

```
Input: $r^+, r^- = r_1^-, ..., r_p^- \neq \{\emptyset\}$
Output: $S, G$
$M :=$ Max_Set_Algorithm$(r^+, minfreq)$
$S = \{t \in M \mid t$ is a cube transversal of $\overline{r^-}$ on $CL(r^+)\}$
$G := \{<\emptyset, \ldots, \emptyset>\}$
for $i := 1$ to $p$ do
  $G' := \{t \in cTr(\overline{r_i^-}, S)$ on $CL(r^+)\}$ \\ use CTR algorithm
  $G' := G' \backslash \{t \in G' \mid \exists u \in M \backslash S : u \geq_g t\}$
  if $G' \neq \{\emptyset\}$ then
    for all $t \in M$ do
      for all tuple $t' \in G'$ unmarked do
        if $t' \geq_g t$ then mark $t'$
      end for
    end for
    $G' := \{t \in G' \mid t$ is marked $\}$
    $G := \min_{\geq_g}(\{v = t \bullet t' \mid v \neq <\emptyset, \ldots, \emptyset>, t \in G$ and $t' \in G'\})$
  end if
end for
return $S, G$
```

Example 3. - Let us consider $r^+ = \{t \in r \mid t[EnjoySport] = `Yes'\}$, the relation $r^- = \{t \in r \mid t[EnjoySport] = `No'\}$ and $minfreq = 1/2$. Then, we have:

- $M = \{<S, W, N>, <S, W, H>\}$
- $G = \{<S, \text{ALL}, \text{ALL}>, <\text{ALL}, W, \text{ALL}>, <\text{ALL}, \text{ALL}, N>\}$
- $S = \{<S, W, N>, <S, W, H>\}$

5 Conclusion

The presented work results from a cross-fertilization between the research fields of discrete mathematics, database and machine learning. We introduce the concept of the cube transversals of a categorical database relation. This concept is used to the problem of computing the difference of two uni-compatible datacubes, a new OLAP functionality which can provides users with a focus on new trends emerging from data sets collected at different points along the time. To the best of the author knowledge it is the first time that the problem of the difference of two uni-compatible datacubes (without computing the two cubes) is studied. Set operations on convex spaces of cube lattices are an interesting future work. They allow a merging approach for mining boundary sets of constrained multidimensional patterns (with arbitrary monotone and/or antimonotone constraints).

References

1. R. J. Bayardo, Jr. Efficiently Mining Long Patterns from Databases. In *Proceedings of the International Conference on Management of Data, SIGMOD*, pages 85–93, 1998.

2. K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs. In *Proceedings of the International Conference on Management of Data, SIGMOD*, pages 359–370, 1999.
3. T. Calders, R. Ng, and J. Wijsen. Searching for Dependencies at Multiple Abstraction Levels. In *ACM Transactions on Database Systems, ACM TODS*, volume 27(3), pages 229–260, 2002.
4. A. Casali, R. Cicchetti, and L. Lakhal. Cube Lattices: a Framework for Multidimensional Data Mining. In *Proceedings of the 3rd SIAM International Conference on Data Mining, SDM*, pages 304–308, 2003.
5. A. Casali, R. Cicchetti, and L. Lakhal. Extracting semantics from data cubes using cube transversals and closures. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 69–78, 2003.
6. A. Casali, R. Cicchetti, and L. Lakhal. Mining Concise Représentations of Frequent Multidimensional Patterns. In *Proceedings of the 11th International Conference on Conceptual Structures, ICCS*, 2003.
7. G. Dong and J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, KDD*, pages 43–52, 1999.
8. T. Eiter and G. Gottlob. Identifying The Minimal Transversals of a Hypergraph and Related Problems. In *SIAM Journal on Computing*, volume 24(6), pages 1278–1304, 1995.
9. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Data Mining and Knowledge Discovery*, volume 1(1), pages 29–53, 1997.
10. D. Gunopulos, H. Mannila, R. Khardon, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the 16th Symposium on Principles of Database Systems, PODS*, pages 209–216, 1997.
11. J. Han, J. Pei, G. Dong, and K. Wang. Efficient Computation of Iceberg Cubes with Complex Measures. In *Proceedings of the International Conference on Management of Data, SIGMOD*, pages 441–448, 2001.
12. L. Lakshmanan, J. Pei, and J. Han. Quotient Cube: How to Summarize the Semantics of a Data Cube. In *Proceedings of the 28th International Conference on Very Large Databases, VLDB*, 2002.
13. S. Lopes, J. Petit, and L. Lakhal. Efficient Discovery of Functional Dependencies and Armstrong Relations. In *Proceedings of the 7th International Conference on Extending Database Technology, EDBT*, pages 350–364, 2000.
14. H. Mannila and H. Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. In *Data Mining and Knowledge Discovery*, volume 1(3), pages 241–258, 1997.

Mining Periodic Patterns in Sequence Data

Kuo-Yu Huang and Chia-Hui Chang

Department of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320
want@db.csie.ncu.edu.tw, chia@csie.ncu.edu.tw

Abstract. Periodic pattern mining is the problem that regards temporal regularity. There are many emerging applications in periodic pattern mining, including web usage recommendation, weather prediction, computer networks and biological data. In this paper, we propose a Progressive Timelist-Based Verification (PTV) method to the mining of periodic patterns from a sequence of event sets. The parameter *min_rep*, is employed to specify the minimum number of repetitions required for a valid segment of non-disrupted pattern occurrences. We also describe a partitioning approach to handle extra large/long data sequence. The experiments demonstrate good performance and scalability with large frequent patterns.

1 Introduction

Pattern mining plays an important role in data mining tasks, e.g. association mining, inter-transaction rule [6] sequential pattern and episode mining [4], etc. However, temporal regularity also plays an important roleIn association mining, we may specify a frequent pattern called the "Beer and Diaper" with support level of 10% and confidence level of 70%. If we consider the influence of time, we may find that the pattern, "Beer and Diaper", occurs at every Friday night for 20 weeks in a row. This cyclic association rule is a kind of periodic pattern which can be applied in period predictions. Besides, periodic patterns in biological data is also an important issues [3]. Periodic patterns in biological data may be a mechanism that provides regular arrays of spatial and function groups, useful for structural packing or for one to one interactions with target molecules. Periodic conservation of amino acids may be useful in structural packing of two or more polypeptide chains of the same or different proteins. Periodically placed amino acid side chains can also facilitate one to one interactions with target atoms showing similar periodicity" [3].

There have been a number of recent studies in periodic pattern mining. For example, cyclic association rules proposed by Ozden, et al. [5], partial periodic patterns by Han, et al. [1], and asynchronous periodic patterns by Yang, et al. [7, 2]. The so called **full periodicity** specifies the behavior of the time series at all points in period, while **partial periodicity** specifies the behavior at some but not all points in time. Ozden, et al. define the problem of discovering cyclic association rules that display regular cyclic variation over time [5]. This motivation is based on the observation that an association rule may not have the

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 401–410, 2004.

user-specified minimum confidence or support over the entire time spectrum, but its confidence and support may be above the minimum threshold within certain time intervals. Note that what Ozden, et al. considered are partial periodic patterns with **perfect** periodicity in the sense that the pattern reoccurs in every cycle, with 100% confidence. By studying the interaction between association rules and time, they applied three heuristics: *cycle pruning*, *cycle skipping* and *cycle elimination* to find cyclic association rules in temporal data.

Since real life patterns are usually **imperfect**, Han et al. [1] presented several algorithms to efficiently mine partial and imperfect periodic patterns, by exploring some interesting properties related to partial periodicity, such as the Apriori property and the max-subpattern hit set property, and by shared mining of multiple periods. In order to tame the restriction of cyclic association rule, Han, et al. used confidence to measure how significant a periodic pattern is. The confidence of a pattern was defined as the occurrence count of the pattern over the maximum number of periods of the pattern length in the sequence. For example, $(a, *, b)$ is a partial pattern of period 3 (The character "*" is a "don't care" character, which can match any single set of events); its occurrence count in the event series "a{b,c}baebaced" is 2; and its confidence is 2/3, where 3 is the maximum number of periods of length 3. Nevertheless, the proposed mining model works only for synchronous periodic pattern mining.

Therefore, Yang et al. [7] proposed to mine for asynchronous periodic patterns that are significant using a subsequence of symbols. Two parameters, *min_rep* and *max_dis*, are employed to qualify valid patterns. The intuition is that a pattern needs to repeat itself at least a certain number (*min_rep*) of times to demonstrate its significance and periodicy. On the other hand, the disturbance between two valid segments has to be within some reasonable bound (*max_dis*). A two-step algorithm is devised to first generate potential periods by distance-based pruning, followed by an iterative procedure to derive and validate candidate patterns and locate the longest valid subsequence for 1-pattern (called LSI). The second step then applies a level-wise search to generate the subsequences of i-patterns based on valid subsequences of all $(i-1)$-patterns with the same period length. Note that in order to discover the longest subsequence, a longer segment can be broken into small segments when two segments overlap. For some applications, such as biological data sequence, users might be more interested in the maximum segments that a periodic pattern can extend. In additions, the level-wise search for $(i-1)$-patterns might degrade for data sequences of event sets.

To address these problems, Huang and Chang in [2] propose a general model for mining asynchronous periodic patterns, where each valid segment is required to be of maximum and at least *min_rep* contiguous matches of the pattern. They decompose the problem into two subproblems, valid segment discovery and asynchronous subsequence composition. Valid segments include single-event 1-patterns, multi-event 1-patterns, and complex i-patterns $(i > 1)$. Three algorithms SPMiner, MPMiner, and CPMiner are devised respectively. Finally, all valid segments with respect to a pattern can be combined to form an asynchronous sequence by APMiner. Just like association rule mining, the computation load occurs at valid segment discovery. In this paper, we focus on the mining of valid segments and devise a progressive timelist-based verification algorithm (called PTV), which improve the performance of SPMiner and MPMiner. Experimental results show that this method offers better performance than the previous research when there are many periodic events.

The remaining parts of the paper are organized as follows. In Section 2, we define the problem of periodic pattern mining for sequence of event set. Section 3 presents our algorithm for mining periodic patterns from sequence data. Experiments and performances of the algorithm study are reported in Section 5. Complexity analysis and comparison to previous work are discussed 4. Our conclusion are presented in Section 6.

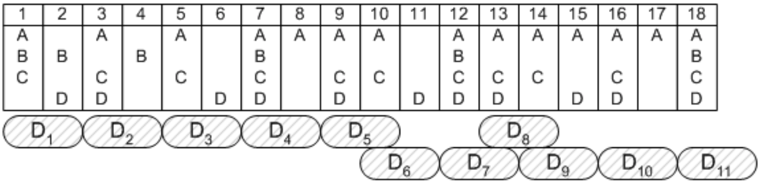
2 Problem Definition

In this section, we define the problem of periodic periodic mining. The problem definition is similar to [2]. Let E be a set of all events. An event set is a nonempty subset of E. A eventset sequence S is a set of time records where each time record is a tuple (tid, X) for time instant tid and event set X. A eventset sequence stored in form of (tid, X) is called horizontal format (see Fig. 1). We say that an event set Y is supported by a time record (tid, X) if and only if $Y \subseteq X$. An event set with k events is called a k-event set.

Definition 1. *A* ***pattern*** *with period l is a nonempty sequence $P = (p_1, p_2, \ldots, p_l)$ where p_1 is an event set and others are either an event set or *, i.e. $p_j \in (2^E - \emptyset) \cup \{*\}$ for $2 \leq j \leq l$.*

Since a pattern can start anywhere in a sequence, we only need to consider patterns that start with a non-"*" symbol. A pattern P is called an i-pattern if exactly i positions in P contain event sets.

Definition 2. *Given a pattern $P = (p_1, p_2, \ldots, p_l)$ with period l and a sequence of l event sets $D' = (d_1, d_2, \ldots, d_l)$, we say that P* ***matches*** *D' (or D'* ***supports*** *P) if and only if, for each position j $(1 \leq j \leq l)$, either $p_j = *$ or $p_j \subseteq d_j$ is true. D' is also called a* ***match*** *of P.*



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A		A		A		A	A	A	A		A	A	A	A	A	A	A
B	B		B			B					B						B
C		C		C		C		C	C		C	C	C		C		C
	D	D			D	D		D		D	D	D		D	D		D

Fig. 1. The matches of (AC,*,*) in eventset sequence S.

In general, given a sequence of event sets and a pattern P, multiple matches of P may exist. In Fig. 1, $D_1, D_2, \ldots, D_{11}$ are 11 matches of $(AC, *, *)$.

Definition 3. *Given a pattern P with period l and a sequence of event sets D, a list of k ($k > 0$) disjoint matches of P in D is called a* ***segment*** *with respect to P if and only if it forms a contiguous subsequence of D. Here, k is referred to as the number of repetitions of this segment. A segment is* ***maximum*** *if there are no other contiguous matches at either end. For convenience, we use a 4-tuple (P, l, rep, pos) to denote a segment of pattern P with period l starting from position pos for rep times.*

In Fig. 1, $D_1, \ldots, D_5$ are continuous and disjoint matches. Therefore, we can use $S_1 = \{(AC, *, *), 2, 5, 1\}$ to indicate a segment with period 2 starting from position 1 for 5 times. Note that D_1, D_2, D_3 and D_4 also form a segment but it is not maximum.

Definition 4. *A maximum segment S with respect to a periodic pattern P is a* ***valid segment*** *if and only if the number of repetitions of S (with respect to P) is at least the required minimum repetitions (i.e., min_rep).*

For Fig. 1, if we set $min\_rep = 3$ and consider only pattern AC with period 2, there will be two valid segments S_1 (D_1, D_2, D_3, D_4 and D_5) and S_2 (D_6, D_7, D_9, D_{10} and D_{11}) returned. The problem is formulated as follows: given a eventset sequence and the parameters min_rep, the problem is to find all valid segment of periodic pattern with significant periods between L_{min} and L_{max} specified by the user.

3 Progressive Timelist-Based Verification

In this section, we explore a 2-phase algorithm, Progressive Timelist-Based Verification (PTV), for mining periodic patterns in eventset sequence. In the first phase, we modify the data structure of the SPMiner in [2] to discover single-event 1-patterns. In the second phase, we devise a timelist-based verification mechanism to combine all probable segments of multi-event 1-patterns.

The inputs to PTV include a vertical format database VD and the interesting period interval specified by L_{min} and L_{max}. The timelist in VD is maintained for each event. Essentially, PTV checks the time lists of each eventset for each possible period p ($L_{min} < p < L_{max}$) by a procedure called **PeriodicyCheck**. It starts by checking possible valid segments from the timelists for each single events (Phase I). If there exists a valid segment for an event, such events are enumerated in depth first order to form event sets. Duplicate enumerates are avoided by forcing an alphabetic order on the events. For each combined event set, the timelist is obtained by timelist intersection from the constituent events (Phase II). Again, the PeriodicyCheck procedure is applied to see if valid segments exist for the event set. Enumeration stops whenever no valid segment exists for an event set.

Given a timelist and period p, the task of **PeriodicyCheck** is to output valid segments with period p. This is implemented by keeping tracks of p independent segments in a data structure called $CSeg$, where each $CSeg$ records the start

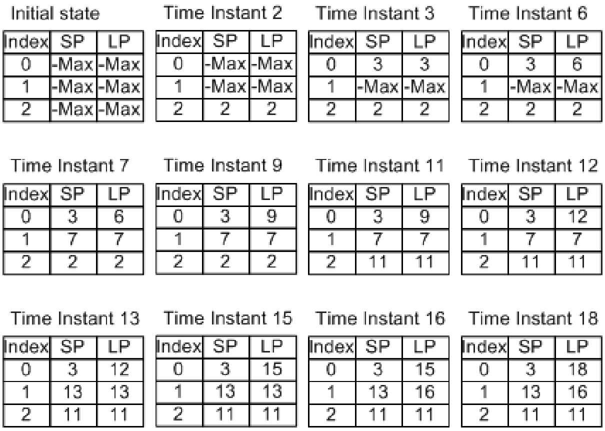
```
Procedure of PTV ($VD$, $L_{min}$,$L_{max}$)
/* Phase I */
for $p$=$L_{min}$ to $L_{max}$ do $FList_p$= NULL;
for each event $E_i \in VD$ do
    if ($|E_i.TimeList| < min\_rep$) then continue;
    for $p$=$L_{min}$ to $L_{max}$ do
        if (PeriodicyCheck($E_i$, $E_i.TimeList$, $p$))==true) then
            Append $E_i$ to $FList_p$;
/* Phase II */
for $p$=$L_{min}$ to $L_{max}$ do
    if ($|FList_p| > 1$) then
        for each event $E_i \in FList_p$ do
            $Node.Head = E_i$; $Node.Tail$ = all event $E_j \in FList_p$ $(j > i)$;
            $Node.TimeList = E_i.TimeList$; DFS($Node$, $p$);
```

```
Procedure of PeriodicyCheck ($EvtSet$, $TimeList$, $p$)
Allocate data structure $Cseg[p]$;
/* Initialization */
for $i$=1 to $p$ do
    $CSeg[i].LP = -Max$; $CSeg[i].SP = -Max$;
/* Validation */
$Valid$ = false;
for each time instant $T_i \in TimeList$ do
    $pos = T_i\%p$;
    if ($T_i - CSeg[pos].LP$) == $p$) then $CSeg[pos].LP = T_i$; continue;
    if ($CSeg[pos].LP - CSeg[pos].SP \geq (min\_rep - 1) * p$) then
        Output ($EvtSet$, $(CSeg[pos].LP - CSeg[pos].SP)/p + 1$, $CSeg[pos].SP$);
        $Valid$= true;
    $CSeg[pos].SP = T_i$; $Cseg[pos].LP = T_i$;
/* Rechecking */
for $i = 1$ to $p$ do
    if ($CSeg[pos].LP - CSeg[pos].SP \geq (min\_rep - 1) * p$) then
        Output ($EvtSet$, $(CSeg[pos].LP - CSeg[pos].SP)/p + 1$, $CSeg[pos].SP$);
        $Valid = true$;
return $Valid$;
```

```
Procedure of DFS($Node$, $p$)
for each $E_i \in Node.Tail$
    $newC.Head = Node.Head \cup E_i$;
    $newC.Tail$ = Tail($Node.Tail$);
    $newC.TimeList$ = Intersection($Node.TimeList$, $E_i.TimeList$);
    if (PeriodicyCheck($newC.Head$, $newC.TimeList$, $p$)== true) then
        DFS($newC, p$);
```

Fig. 2. PTV: Progressive Timelist-Based Verification algorithm.

position (SP) and last position (LP) for current segment. For each time instant T_i in the timelist, we compute the offset position $pos = T_i\%p$ and compare T_i to $CSeg[pos].LP$. If $T_i - CSseg[pos].LP$ is exactly p, it implies that this event set



Initial state

Index	SP	LP
0	-Max	-Max
1	-Max	-Max
2	-Max	-Max

Time Instant 2

Index	SP	LP
0	-Max	-Max
1	-Max	-Max
2	2	2

Time Instant 3

Index	SP	LP
0	3	3
1	-Max	-Max
2	2	2

Time Instant 6

Index	SP	LP
0	3	6
1	-Max	-Max
2	2	2

Time Instant 7

Index	SP	LP
0	3	6
1	7	7
2	2	2

Time Instant 9

Index	SP	LP
0	3	9
1	7	7
2	2	2

Time Instant 11

Index	SP	LP
0	3	9
1	7	7
2	11	11

Time Instant 12

Index	SP	LP
0	3	12
1	7	7
2	11	11

Time Instant 13

Index	SP	LP
0	3	12
1	13	13
2	11	11

Time Instant 15

Index	SP	LP
0	3	15
1	13	13
2	11	11

Time Instant 16

Index	SP	LP
0	3	15
1	13	16
2	11	11

Time Instant 18

Index	SP	LP
0	3	18
1	13	16
2	11	11

Fig. 3. Execution process for event D with period 3.

has occurred at $(T_i - p)$-th time instant. In this case, we update $Cseg[pos].LP$ by T_i. If otherwise, $T_i - CSseg[pos].LP$ is not p, it implies the last segment has been interrupted. In this case, output this segment if length of current segment $(CSeg[pos].LP - CSeg[pos].SP)$ is greater than $(min\_rep - 1) * p$ and reset $CSeg[pos].SP$ and $CSeg[pos].LP$ to T_i (Step 6∼13 in the *PeriodicyCheck* in Fig. 2). Step 3∼4 of PeriodicyCheck initialize the allocated data structure. In order to check the unfinished segments, we recheck $CSeg$ again and output valid segments (Step 15∼18 in the *PeriodicyCheck*).

Let us use an example to illustrate this algorithm. Given the eventset sequence in Fig. 1, with parameters $min\_rep = 4$. The singular periodic pattern can be mined by the following steps.

- Phase I: For each period p, from L_{min} to L_{max}, use PeriodicyCheck procedure to check whether an event contains valid segments. Take period 3 and event D for example, three $CSeg$ are created and initialized with $SP = -Max$ and $LP = -Max$. For each time instant T_i in $D.TimeList$, the offset is computed by $pos = T_i\%3$. As shown in Fig. 3, time instant 3 has an offset at position 0, therefore $CSeg[0]$ is set with $SP = 3$ and $LP = 3$. Since time instant 6, 9, 12, 15 and 18 all have an offset 0, $CSeg[0].LP$ is updated five times. For time instant 2 and 11, both of them have an offset at position 2. However, time instant $11 - CSeg[2].LP$ does not equal 3, indicating an interrupt of the previous segment. Finally, segment $(D, 3, 6, 3)$ is output as valid. Since event D contains valid segment, it will be inserted into $FList_3$.
- Phase II: For each period p, from L_{min} to L_{max}, enumerate possible event sets from $FList_p$ in depth-first order by a recursive procedure **DFS**. The input to **DFS** is a $Node$ data structure which contains head event set, tail event list, and the timelist for head event set. Take period 2 for example. $FList_2$ contains three events $\{A, C, D\}$. Assume an alphabetic order, $\{A, C\}$

is first enumerated by combining $Node.head = \{A\}$ with the first element from $Node.tail = \{C, D\}$. The timelist for $Node.head$ is the intersection of $A.TimeList$ and $C.TimeList$. With the timelist information, **PeriodicyCheck** procedure is then called to check if valid segments exist for this event set $\{A, C\}$. Since valid segments exist for this event set, **DFS** is called recursively with new node $(\{A, C\}, \{D\}, \{A, C\}.TimeList)$.

4 Discussion and Algorithm Comparison

In this section, we analysis the algorithms time/space complexity and discuss the solution when the sequence data is too long/large to fit in memory space.

4.1 Comparison

In this paper, PTV(I) is devised to discover all valid segments for each single event. Therefore, we compare the mining procedure of our proposed algorithm, PTV(I), with the LSI algorithm proposed in [7].

The overall time for processing PTV(I) for a given event e is n_e, where n_e is the number of occurrences of event e. For a given period length l, the time to find the singular periodic pattern for all events is hence $\sum_{\forall e} n_e$ which is equivalent to one database scans. Let D denotes the number of time instants and T be the average number of events in each time instant. The database size can be represented by $D * T$. Consequently, the time complexity to discover all valid segments of 1-pattern for all periods is $O(S * T * L_{max})$ while $L_{min} = 1$. The data structured used for PTV(I) when processing an event is $CSeg$. The size of the data structure is a multiple of L_{max}, which can be reused for all events. Therefore, the space complexity is $O(L_{max})$.

The discovery process of LSI's first step moves among three phases for segment validation (phase A), segment growth (phase B) and sequence extension (phase C). Therefore, LSI(A+B) is equivalent to PTV(I). We analyze the time complexity and space complexity of the PTV(I) below. The time complexity of LSI to discover the "longest" single event subsequence from a event sequence is $k * M * L_{max}$, where L_{max} is the maximum period length, M is the event sequence size and k is abbreviated for $min\_rep + max\_dis + L_{max}$ [7]. For a eventset sequence, the size of the database can be represented by $D * T$ for D time instants, each with an average of T events ($M = D * T$).

The space complexity of LSI is $(max\_dis + L_{max}) * N * L_{max} + min(N * L_{max}, min\_rep * L^2_{max} * N)$ as analyzed in [7]. To discover valid segments as defined in this paper, the space of LSI can be approximated by O$(L^2_{max} * N)$.

PeriodicyCheck in PTV is similar to the hash based validation (HBV) in SPMiner. In SPMiner, the access to $CSeg$ is two reads and write for Last and Rep updating; but the access to $CSeg$ needs only last position updating (Step 9 in the *PeriodicyCheck*). The MPMiner use a segment-based combination to generate all multi-event 1-pattern. However, segment-based combination technology is not a robust method while the number of the valid segments are large. PTV(II) use a timelist-based validation method, but the time complexity is hard to analysis. Therefore, we compare PTV(II) and MPMiner by experiment result later.

4.2 Extra Long/Large Sequence

Sometimes, the sequence data is too long/large to fit in memory space. In this case, we mine the periodic patterns by a partition-and-validation strategy. Firstly, the algorithm subdivides the extra-large sequence data into n non-overlapping partitions. Each partition can be handled in memory by PTV. Further, each partition is transformed into vertical format. For the fist partition, the valid segment can be mined by Initialization and Validation step in procedure PeriodicyCheck. For the succeeding partitions, the initial start (last) position is inherited from the last partition and valid segment pattern is explored by validation phase. In the final partition, valid segment is discovered by Validation and Rechecking step.

5 Experimental Results

To assess the performance of algorithm PTV, we conduct several experiments on a computer with a CPU clock rate of 1.13GHz and 256MB of main memory, the program is written by visual C++ in windows XP platform.

5.1 Biological Data

We first apply PTV to discover periodic conservation of the protein sequences, which is an important problem in bioinformatics. We used data in the PROSITE database of the ExPASy Molecular Biology Server (http://www.expasy.org/). We selected a protein sequence P17437 (Skin secretory protein XP2) with a known periodic pattern {A,P,A,P,A,*,*,E,*,*}, which reported in [3]. As expected, several periodic patterns which are related to the known periodic conservation are discovered. It is indicated that our algorithm can be used in protein sequence. It is worth to note that we also discover an interesting and longest pattern {A,P,A,P,A,E,G,E,A,P} occurring 11 times (approximately 46%) in the known periodic pattern. It may be a core pattern, since the partial slots of the pattern allow some mutations.

5.2 Synthetic Data

For the purpose of performance evaluation, we use the same synthetic data as [2]. The default parameters of synthetic generator are data size D = 50K, event N = 1000, Avg. event in a time slot T = 10, potential pattern C = 3, pattern length L = 4, frequent event in a time slot I = 4, number of segment for each pattern S = 10. In order to make PTV more efficiency, we also use the PCD pruning strategy to reduce unnecessary period checking [2]. We have run a series of experiments using PTV. The general performance, the effect of parameters, and the scalability of our methods are considered here. We start by looking at the performance of the PTV with parameter $min\_rep = 25$, $L_{min} = 1$ and $L_{max} = 20$.

5.3 Valid Segments for Single Event 1-Patterns

In this section, we compare the scalability of the three segment validation algorithms, including PTV(I), SPMiner and LSI. The scalability of PTV(I) is shown in Fig. 4(a). The total running time for PTV(I) increase smoothly, as analyzed in Section 4.1; whereas the running time for LSI increases dramatically since the distance-based pruning technique has comparatively less to prune. The scalability for PTV(I) was also better than SPMiner. In Fig. 4(b) the total running time for PTV(I) is linear to T, whereas the running time for LSI increases dramatically. It is also evident that the number of pruned patterns is rapidly decreasing when the T increasing.

5.4 Valid Segments for Multi-event 1-Patterns

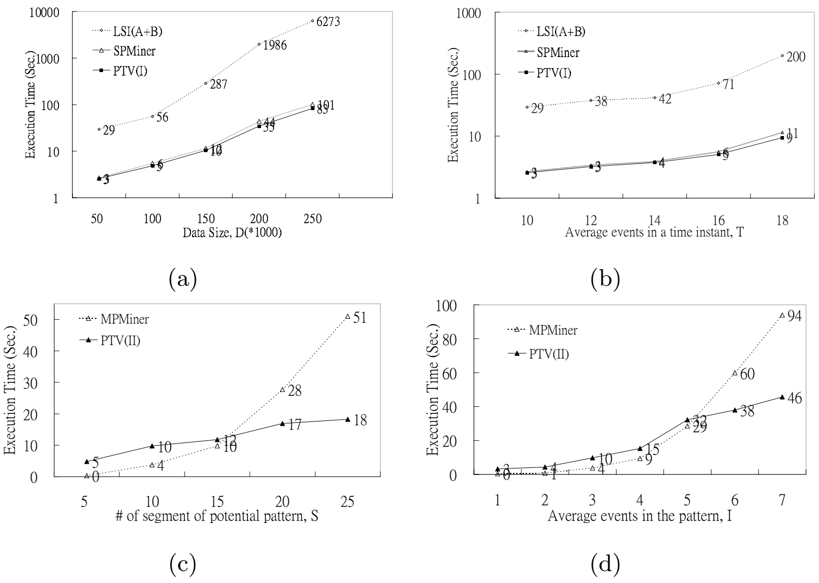
Since LSI is devised for event sequence. Therefore, we demonstrate the efficiency of PTV(II) by comparing PTV(II) with MPMiner for multi-event 1-patterns. Fig. 4(c)(d) shows the execution time of these two methods. As we can see, MPMiner outperforms PTV(II) while S is small (S less than 15). However, when S is large, the execution time of PTV(II) increases smoothly. But the running time of MPMiner increases sharply. This is because the time complexity of MPMiner has an exponential relation to the number of segments in the worst case. In contrast, PTV(II) is comparably stable with respect to the number of segments. Fig. 4(d) shows the overall execution time of PTV(II) and MPMiner to find all valid segments for eventset. The x-axis shows the value of average event in the pattern, whereas the y-axis shows the overall execution time of PTV(II) and MPMiner. The execution time of PTV(II) increases more smoothly than MPMiner in Fig. 4(d).

6 Conclusion

In this paper, an efficient method for periodic pattern mining is defined. An algorithm progressive timelist-based validation (PTV) algorithm is devised to discover all valid segments in data sequence. The PTV algorithm the vertical database once and keeps only those timelists for events with valid segments. The experiments show that our algorithm outperform previous research. Periodic pattern mining can be used for data characteristics summarization and periodicity predication.

Acknowledgements

This work is sponsored by Ministry of Economic Affairs, Taiwan under grant 93-EC-17-A-02-S1-029.



(a)

(b)

(c)

(d)

Fig. 4. Performance comparison.

References

1. J. Han, G. Dong, and Y. Yin. Efficient mining paritial periodic patterns in time series database. In *Proceedings of the 15th International Conference on Data Engineering, (ICDE'99)*, pages 106–115, 1999.
2. K.Y. Huang and C.H. Chang. Asynchronous periodic patterns mining in temporal databases. In *Proceedings of the IASTED International Conference on Databases and Applications (DBA'04)*, pages 43–48, 2004.
3. M. V. Katti, R. Sami-Subbu, P. K. Ranjekar, and V. S. Gupta. Amino acid repeat patterns in protein sequences: Their diversity and structural-function implications. *Protein Science*, 9:1203–1209, 2000.
4. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210–215, 1995.
5. B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proceedings of the 14th International Conference on Data Engineering, (ICDE'98)*, pages 412–421, 1998.
6. A. K. H. Tung, J. Han H. Lu, and L. Feng. Efficient mining of intertransaction association rules. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):43–56, 2003.
7. J. Yang, W. Wang, and P.S. Yu. Mining asynchronous periodic patterns in time series data. *IEEE Transaction on Knowledge and Data Engineering*, 15(3):613–628, 2003.

Author Index