Alfredo Cuzzocrea
Umeshwar Dayal (Eds.)

LNCS 7448

Data Warehousing and Knowledge Discovery

14th International Conference, DaWaK 2012
Vienna, Austria, September 2012
Proceedings

Springer

Alfredo Cuzzocrea Umeshwar Dayal (Eds.)

Data Warehousing and Knowledge Discovery

14th International Conference, DaWaK 2012
Vienna, Austria, September 3-6, 2012
Proceedings

Springer

Volume Editors

Alfredo Cuzzocrea
ICAR-CNR and University of Calabria
via P. Bucci 41C, 87036 Rende (CS), Italy
E-mail: cuzzocrea@si.deis.unical.it

Umeshwar Dayal
Hewlett-Packard Labs
1501 Page Mill Road, MS 1142
Palo Alto, CA 94304, USA
E-mail: umeshwar.dayal@hp.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-32583-0 e-ISBN 978-3-642-32584-7
DOI 10.1007/978-3-642-32584-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012944032

CR Subject Classification (1998): H.2.7-8, H.2.4, H.2, I.2.6, H.3, H.4, J.1, H.5.3

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Data warehousing and knowledge discovery is an extremely active research area where a number of methodologies and paradigms converge, with coverage on both theoretical issues and practical solutions. Under a broad vision, data warehousing and knowledge discovery has been widely accepted as a key technology for enterprises and organizations, as it allows them to improve their abilities in data analysis, decision support, and the automatic extraction of knowledge from data. With the exponentially growing amount of information to be included in the decision-making process, data to be considered become more and more complex in both structure and semantics. As a consequence, novel developments are necessary, both at the methodological level, e.g., complex analytics over data, and at the infrastructural level, e.g., cloud computing architectures. Orthogonal to the latter aspects, the knowledge discovery and retrieval process from huge amounts of heterogeneous complex data represents a significant challenge for this research area.

Data Warehousing and Knowledge Discovery (DaWaK) has become one of the most important international scientific events bringing together researchers, developers, and practitioners to discuss the latest research issues and experiences in developing and deploying data warehousing and knowledge discovery systems, applications, and solutions.

The 14th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2012) continued the tradition by discussing and disseminating innovative principles, methods, algorithms, and solutions to challenging problems faced in the development of data warehousing and knowledge discovery, and applications within these areas. In order to better reflect novel trends and the diversity of topics, like the previous edition, DaWaK 2014 was organized into four tracks: Cloud Intelligence, Data Warehousing, Knowledge Discovery, and Industry and Applications.

Papers presented at DaWaK 2014 covered a wide range of topics on cloud intelligence, data warehousing, knowledge discovery, and applications. The topics included data warehouse modeling, spatial data warehouses, mining social networks and graphs, physical data warehouse design, dependency mining, business intelligence and analytics, outlier and image mining, pattern mining, and data cleaning and variable selection.

It was encouraging to see that many papers covered emerging important issues such as social network data, spatio-temporal data, streaming data, non-standard pattern types, complex analytical functionality, multimedia data, as well as real-world applications. The wide range of topics bears witness to the fact that the data warehousing and knowledge discovery field is dynamically responding to the new challenges posed by novel types of data and applications.

From 112 submitted abstracts, we received 99 papers from Europe, North and South America, Asia, and Oceania, further confirming to us the wide interest in the topics covered by DaWaK within the research community. The Program Committee finally selected 36 papers, yielding an acceptance rate of 32%.

We would like to express our most sincere gratitude to the members of the Program Committee and the external reviewers, who made a huge effort to review the papers in a timely and thorough manner. Owing to the tight timing constraints and the high number of submissions, the reviewing and discussion process was a very challenging task, but the commitment of the reviewers ensured a successful result . We would also like to thank all authors who submitted papers to DaWaK 2012, for their contribution to the excellent technical program.

Finally, we send our warmest thanks to Gabriela Wagner for delivering an outstanding level of support on all aspects of the practical organization of DaWaK 2012. We also thank Amin Anjomshoaa for his support of the conference management software.

September 2012

Alfredo Cuzzocrea
Umeshwar Dayal

Organization

Program Committee Co-chairs

Alfredo Cuzzocrea	ICAR-CNR and University of Calabria, Italy
Umeshwar Dayal	Hewlett-Packard Laboratories, Palo Alto, CA, USA

Program Committee

Alberto Abelló	Universitat Politecnica de Catalunya, Spain
Reda Alhajj	University of Calgary, Canada
Torben Bach Pedersen	Aalborg University, Denmark
Elena Baralis	Politecnico di Torino, Italy
Ladjel Bellatreche	ENSMA, France
Bettina Berendt	KU Leuven, Belgium
Petr Berka	University of Economics, Prague, Czech Republic
Jorge Bernardino	ISEC - Instituto Superior de Engenharia de Coimbra, Portugal
Elisa Bertino	Purdue University, USA
Stephane Bressan	National University of Singapore, Singapore
Erik Buchmann	Karlsruhe Institute of Technology, Germany
Longbing Cao	University of Technology Sydney, Australia
Nitesh Chawla	University of Notre Dame, France
Frans Coenen	The University of Liverpool, UK
Bruno Cremilleux	Université de Caen, France
Judith Cushing	The Evergreen State College, USA
Alfredo Cuzzocrea	University of Calabria, Italy
Karen Davis	University of Cincinnati, USA
Frank Dehne	Carleton University, Canada
Antonios Deligiannakis	Technical University of Crete, Greece
Alin Dobra	University of Florida, USA
Dejing Dou	University of Oregon, USA
Curtis Dyreson	Utah State University, USA
Todd Eavis	Concordia University, Canada
Johann Eder	University of Klagenfurt, Austria
Floriana Esposito	University of Bari "Aldo Moro", Italy
Vladimir Estivill-Castro	Griffith University, Australia
Christie Ezeife	University of Windsor, UK
Ling Feng	Tsinghua University, China
Eduardo Fernandez_Medina	University of Castilla-La Mancha, Spain
Mohamed M. Gaber	University of Portsmouth, UK

Sergio Greco	University of Calabria, Italy
Frank Hoppner	Ostfalia University of Applied Sciences, Germany
Jimmy Huang	York University, Canada
Seung-Won Hwang	postech
Hasan Jamil	Wayne State University, USA
Chris Jermanine	Rice University, USA
Domingo-Ferrer Josep	Rovira i Virgili University, Spain
Murat Kantarcioglu	University of Texas at Dallas, USA
Panagiotis Karras	Rutgers University, USA
Martin Kersten	CWI, The Netherlands
Taghi Khoshgoftaar	Florida Atlantic University, USA
Arno Knobbe	Universiteit Utrecht, The Netherlands
Jens Lechtenboerger	Westfalische Wilhelms - Universität Münster, Germany
Wolfgang Lehner	Dresden University of Technology, Germany
Carson K. Leung	The University of Manitoba, Canada
Jinyan Li	University of Technology, Sydney, Australia
Xuemin Lin	University of New South Wales, Sydney, Australia
Patrick Martin	Queen's University, Ontario, Canada
Michael May	Fraunhofer Institut für Intelligente Analyse und Informationssysteme, Germany
Carlos Ordonez	University of Houston, USA
Apostolos Papadopoulos	Aristotle University, Greece
Jeffrey Parsons	Memorial University of Newfoundland, Canada
Lu Qin	The Chinese University of Hong Kong, Hong Kong
Zbigniew W. Ras	University of North Carolina, USA
Mirek Riedewald	Northeastern University, USA
Stefano Rizzi	University of Bologna, Italy
Domenico Sacca'	University of Calabria, Italy
Maria Luisa Sapino	Università degli Studi di Torino, Italy
Kai-Uwe Sattler	Ilmenau University of Technology, Germany
Timos Sellis Sellis	Institute for the Management of Information Systems and NTUA, Greece
Neeraj Sharma	India IBM Labs
Alkis Simitsis	HP Labs
Domenico Talia	University of Calabria, Italy
David Taniar	Monash University, Australia
Yufei Tao	Chinese University of Hong Kong, Hong Kong
Dimitri Theodoratos	New Jersey Institute of Technology, USA
A Min Tjoa	Vienna University of Technology, Austria
Juan-Carlos Trujillo Mondéjar	University of Alicante, Spain
Panos Vassiliadis	University of Ioannina, Greece

Millist Vincent	University of South Australia, Australia
Gottfried Vossen	University of Münster, Germany
Wei Wang	University of New South Wales, Sydney, Australia
Robert Wrembel	Poznan University of Technology, Poland
Wolfram Wöß	Johannes Kepler University Linz, Austria
Carlo Zaniolo	University of California, Los Angeles, USA
Bin Zhou	University of Maryland, Baltimore County, USA
Esteban Zimányi	Universite Libre de Bruxelles, Belgium

External Reviewers

Elio Masciari	ICAR-CNR, Italy
Fabrizio Angiulli	University of Calabria, Italy
Roberto De Virgilio	Università RomaTre, Italy
Michelangelo Ceci	University of Bari, Italy

Table of Contents

BPMN-Based Conceptual Modeling of ETL Processes

Zineb El Akkaoui[1], José-Norberto Mazón[2], Alejandro Vaisman[1], and Esteban Zimányi[1]

[1] Department of Computer and Decision Engineering (CoDE)
Université Libre de Bruxelles,
Brussels, Belgium
{zelakkao,avaisman,ezimanyi}@ulb.ac.be
[2] Department of Software and Computing Systems (WaKe)
Universidad de Alicante,
Alicante, Spain
jnmazon@dlsi.ua.es

Abstract. Business Intelligence (BI) solutions require the design and implementation of complex processes (denoted ETL) that extract, transform, and load data from the sources to a common repository. New applications, like for example, real-time data warehousing, require agile and flexible tools that allow BI users to take timely decisions based on extremely up-to-date data. This calls for new ETL tools able to adapt to constant changes and quickly produce and modify executable code. A way to achieve this is to make ETL processes become aware of the business processes in the organization, in order to easily identify which data are required, and when and how to load them in the data warehouse. Therefore, we propose to model ETL processes using the standard representation mechanism denoted BPMN (Business Process Modeling and Notation). In this paper we present a BPMN-based metamodel for conceptual modeling of ETL processes. This metamodel is based on a classification of ETL objects resulting from a study of the most used commercial and open source ETL tools.

1 Introduction

The term Business intelligence (BI) refers to a collection of techniques used for identifying, extracting, and analyzing business data, to support decision-making. BI applications include a broad spectrum of analysis capabilities, including On-Line Analytical Processing (OLAP) and data mining tools. In most cases, organizational data used by BI applications come from heterogeneous and distributed operational sources that are integrated into a data warehouse (DW). To achieve this integration, the data warehousing process includes the extraction of the data from the sources, the transformation of these data (e.g., to correct semantic and syntactic inconsistencies) and the loading of the warehouse with the cleansed, transformed data. This process is known as ETL (standing for Extraction, Transformation, Load). It has been widely argued that the ETL process development is complex, error-prone, and time-consuming [9,13]. Today's business dynamics requires fresh data for BI, posing new challenges to the way in which the development of ETL process is carried out. Real-time data warehousing and right-time data warehousing [11] are already established and accepted concepts. These new requirements call for agile and flexible ETL tools, that can quickly produce and modify executable code based on these changing needs. To address these challenges, we believe that ETL processes must be aware of business processes, in order to easily identify which data are required and how to include them in the data warehouse. We remark that the need for a tighter integration of data and processes has been acknowledged by the database community [1]. As an example, consider an scenario where a sales business process is such that a discount applied to a sale is directly related to the customer's purchases during the last year: the more purchases a customer has performed, the greater the discount that will be applied. In this scenario, the value of the discount that is stored in the data warehouse may change according to the related sales business process. Moreover, ETL tools in modern BI projects must interact with foreign applications and processes, for example, to get real-time currency rate through web services. To accomplish these goals, we propose to model ETL processes using the standard business process representation mechanism BPMN (standing for Business Process Model and Notation)[1].

To allow matching ETL and business process concepts, we first need to produce a classification of ETL. Even though some efforts were carried out to produce such a classification (e.g., [13]), a generic, vendor-independent classification is still missing. Therefore, our *first contribution* is a classification of ETL processes, that accounts for both, the data and process aspects of ETL. Based on this classification, as our *second and main contribution*, we extend BPMN in a such a way that ETL processes are modeled analogously to business process. Concretely, we propose a novel vendor-independent *metamodel* which allows designers to focus on the ETL semantics rather than in implementation issues. Based on this metamodel, conceptual models for ETL can be produced. Since these models are based on BPMN, a translation to a logical model and code generation are strainghtforward. Metamodel consistency conditions can be defined using Object Constraint Language (OCL)[2] expressions [3]. The rationale behind our proposal is the characterization of the ETL process as a combination of two perspectives: (i) as a *control process*, responsible of synchronizing the transformation's flows; (ii) as a *data process*.

This paper is organized as follows. Section 2 studies existing work on ETL models and taxonomies. Section 3 presents preliminaries and our running example, while Section 4 introduces the rationale for our approach. The ETL classification is presented in Section 5. The BPMN-based metamodel is discussed in Section 6. We conclude in Section 7.

[1] http://www.bpmn.org
[2] http://www.omg.org/spec/OCL/2.0.

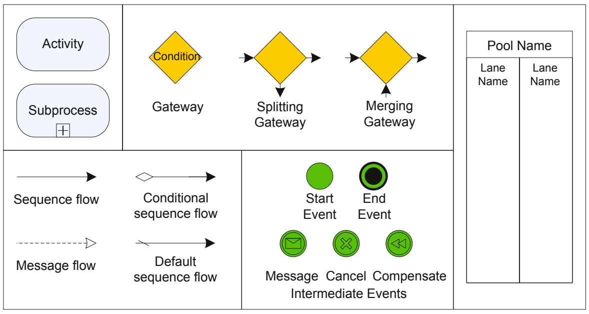


Fig. 1. BPMN main objects and notation

2 Related Work

Conceptual models that use a workflow language for ETL design have been previously proposed [7,8]. These approaches use modeling languages specifically designed for ETL, thus, opposite to our work, not aimed at integrating BP and ETL. Moreover, an ETL programming approach using the Python language (which prevents a vendor-independent design) has been proposed in [10]. The work we present in this paper is built along the lines of [2] and [5]. The former introduces a basic set of BPMN constructs that are used to translate a conceptual ETL model into BPEL (a workflow execution language). In [5] the authors present a Model-Driven Architecture approach to design the ETL processes by means of the UML Activity Diagram; in a sequel of this work, a representation of the ETL process is produced through a manually-designed specific-vendor metamodel [4]. None of these proposals are aimed at providing a vendor-specific code generation utility. On the contrary, our proposal allows a straightfoward translation from a vendor-independent conceptual model, to a vendor-specific executable code generation. Two key mechanisms allow us to do this: the characterization of the ETL process as a combination of a *control process* and a *data process*, and the ETL classification. With respect to the latter, a limited number of efforts attempted to classify ETL tasks. Among these works, Vassiliadis et al. [13] present a taxonomy of ETL transformations, although only addressing the transformation component of ETL. The Transaction Processing Council (TCP) has built a taxonomy of ETL tasks aimed at providing a test platform for evaluating and comparing ETL tools [15]. This work is based on a benchmark on several ETL tools. Our approach takes into account these two taxonomies in addition to the analysis of several ETL tools, to produce a general classification that includes most of the ETL components.

3 Preliminaries and Running Example

The term *business process* denotes a collection of related, structured activities or tasks that produce an *specific service or product* for a particular customer or customers. In the case of ETL, the service consists in taking data from the data sources to the data warehouse. In this paper, we propose a characterization of ETL processes as business processes, in the same way as this is done with other processes in an organization. We perceive the ETL architecture as a combination of *control and data processes*, where control processes manage the coarse-grained groups of tasks and/or sub-processes, and data processes operate at finer granularity, detailing how input data are transformed and output data are produced. For example, populating each fact (dimension, view, temporary, etc.) table in the data warehouse constitutes a data process. Given this assumption, modeling and designing ETL processes using BPM tools appears natural. BPMN 2.0 is an OMG[3] standard, widely used for modeling business processes at a high level of abstraction, and with a well-defined semantics. We briefly introduce BPMN next. Fig. 1 depicts the most relevant objects in the BPMN specification, which are organized in the following classes. *Activities* represent a task performed as part of a process. Can either be atomic or a collection of atomic tasks (denoted Subprocesses). *Connections* link objects in the diagram. For example, the Gateway component is a particular connection with multiple incoming and outgoing connections. An *Event* is an action that occurs during the execution of a process, and that can catch or throw processes. For example, a process could be modeled to start when a certain condition is met or when a timer notifies that a certain time period has elapsed. *Swimlanes* allow subdividing the BPMN diagram into Pools (process containers) and Lanes (division of a pool in roles) for an organizational purpose. Finally, *Artifacts* allow adding information to the model. Three types of artifacts are defined: data objects, groups, and annotations.

Running Example. We now present the running example we use throughout the paper. A retailing company has a business process related to sales in which the value of the discount applied to a customer is directly related to the purchases made by the customer in the previous twelve months. Thus, data should be consolidated in a data warehouse, and an ETL process must be defined. Operational data reside in a relational database and other kinds of support (e.g., flat and XML) files. These data must be mapped to a data warehouse, designed as a collection of facts and dimension tables. The main data source is a relational database, whose logical schema is shown in the left hand side Fig. 2, and contains information about Orders placed by Customers, processed by Employees, and delivered by Shippers. The logical schema of the *target data warehouse* is shown in Fig. 2 (right hand side). There is a fact table FactSales, and several dimension tables. Dimensions are organized in levels that conform an aggregation hierarchy. For example, the dimension DimCustomer contains the following hierarchy levels: DimCustomer→DimGeography→ DimState→DimCountry→DimArea. The last four levels are also shared with the DimSupplier dimension. The data contained in the hierarchy DimArea→ DimCountry→ DimState come from an XML file called Territories.xml. In addition, to identify which state a city belongs to, the file `cities.txt` is used. It contains three fields separated by tabs.

[3] http://www.omg.org

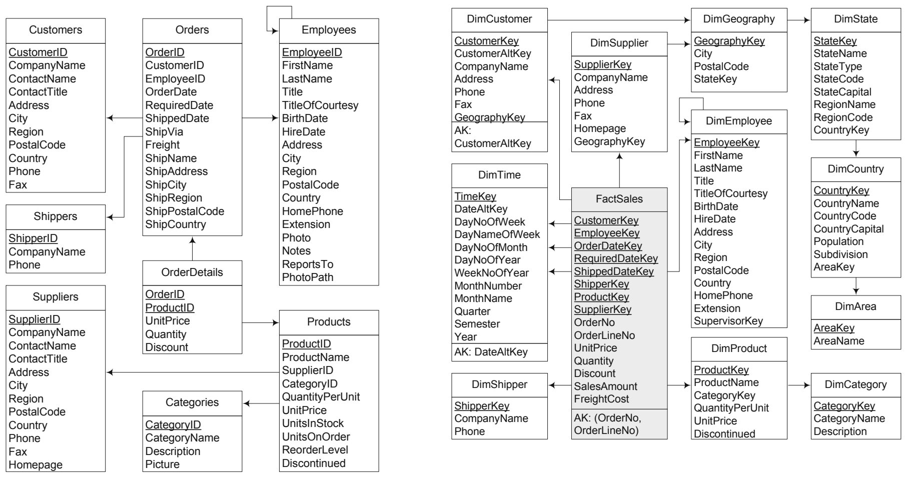


Fig. 2. Source database schema (left), and target data warehouse schema (right)

4 ETL Control Process and Data Processes

We characterize the ETL process as a combination of two perspectives: (i) a *control process* view; (ii) a *data process* view. The control process is able to manage the branching and synchronization of the flow, and handles execution errors and exceptions. This is called *process orchestration*. It also coordinates the information exchanged with foreign processes. This allows to have a high-level overview of the whole ETL process. We illustrate the notion of control process using the portion of our running example that loads the Sales data warehouse. This ETL process is shown in Fig. 3. The control process entails several activities, synchronized by sequences and gateways, controlled through events, and organized in swimlanes. Further, a process can be depicted at different levels of detail. For example, the Temporary Tables Load task from Level 1 is subdivided into subprocesses at Levels 2 and 3. Level 1 shows the overall process in which the dimensions must be loaded prior to loading the fact table. Also, the control process can communicate with external processes through messages such as Convert Currency WS, a web service that provides a currency conversion for generating the current Euro value of UnitPrice from the OrderDetails table, when it is expressed in a foreign currency. The control process can also collaborate with other remote data warehouse processes as it is the case of DimEmployee Load.

The *data process* view captures the original nature of the ETL processes. Typically, this process takes data from the operational databases or other data sources to the data warehouse, providing precise information about the input and output data of each (data) process element. A data process can also create mappings between any data store, not necessarily from the data sources to the data warehouse. In our approach, one fact, dimension (or hierarchy level) is populated at a time through an specific data process. The synchronization between

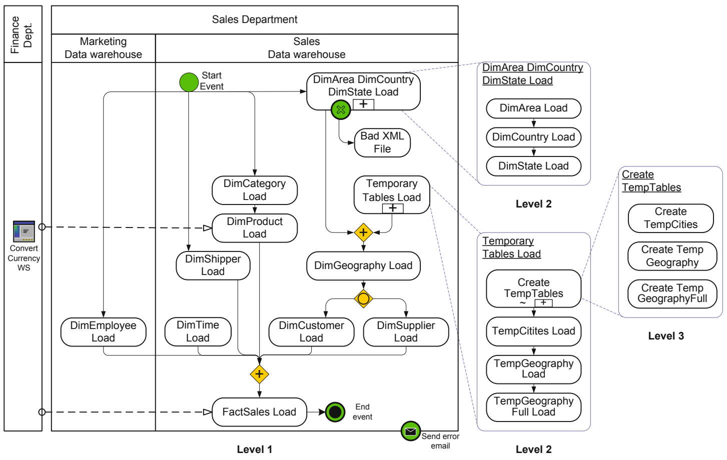


Fig. 3. The ETL control process for loading the Sales fact table

these different data processes is handled at the *control level*. Fig. 4 depicts the data processes that populate the DimGeography dimension. Data come from the Customers and Suppliers tables (see Fig. 2, left), more precisely from the City, Region, PostalCode, and Country attributes. As shown in Fig. 4 (a), attribute State may be null in the Customer and Supplier tables. In these cases, data should be filled with its corresponding value using the TempCities table. Referential integrity in the temporary TempGeography table is also checked previously to the final loading, using lookup tasks. For example, the StateName could be written in the original language or in its English translation (e.g., Kärnten or Carinthia, respectively, for a state in Austria). Also, the state and/or country name can be abbreviated (AZ for Arizona and USA for United States). Fig. 4 (b) shows the sequence of lookup data tasks for these cases.

In summary, although the *data process* allows a fine-grained tracking of data subject to the transformation(s) task, it provides no way of synchronizing these data with external events and decisions. It is therefore definitely complementary with the *control process*, which allows to determine what operations will be performed, in what order, and under which circumstances.

5 A Classification of ETL Objects

We now present a classification of ETL objects, which is the basis of the meta-model we present in Section 6. This classification is founded on a study of some of the most used technologies in the proprietary and open source marketplaces, like Oracle Warehouse Builder (OWB), SQL Server Integration Services (SSIS), and Talend Open Studio, and in existing ETL taxonomies (e.g., [13]). We organize the tasks in a tree structure where the leaves represent the model components and the inner leaves represent the categories they belong to. Of course, categorization

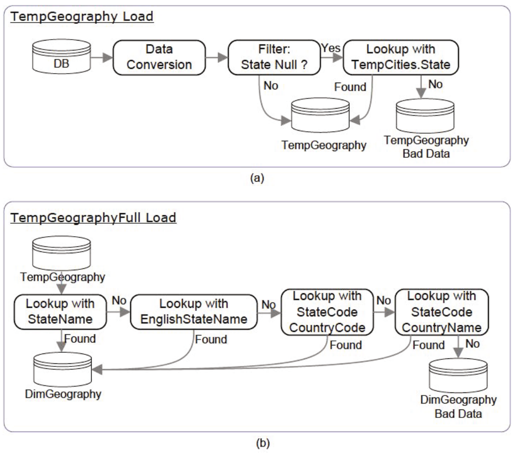


Fig. 4. ETL data process for loading the DimGeography dimension

is a subjective process. For example, [13] is an input-output relationship-based taxonomy. Other task taxonomies are implicitly included in some Business Process languages, which we do not address here. In the ETL classification tree shown in Fig. 5, an ETL component may belong to the following five categories: Task, Sequence, Event, Container, and Artifact. Elements in the tree could be *control objects, data objects, or both.* That means, this classification, together with the characterization in Section 3, induces two different classification trees which we denote: (i) ETL control classification tree; (ii) ETL data classification tree. The former accounts for control processes, while the latter accounts for data processes. This is indicated by the type of line of the rectangles in Fig.5: the ones filled belong to (i), the ones dotted, to (ii). Note that the first two levels in the tree are common to (i) and (ii). At the third level, Control Task belongs to (i), and Data Task belongs to (ii). At this same level, category 'Connection' belongs to both classes of trees. We only show the first levels in this classification. Finer levels will be addressed in Section 6.

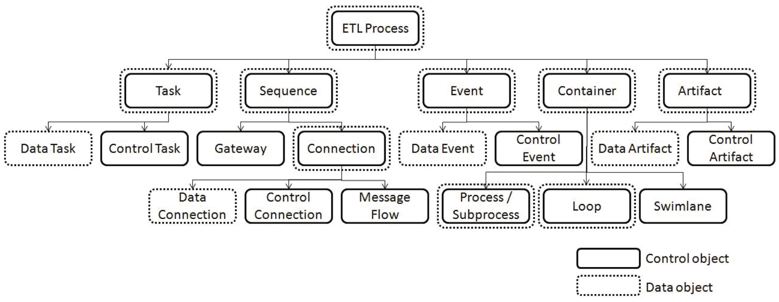


Fig. 5. ETL classification tree

5.1 ETL Control Classification Tree

We briefly describe the main subclasses in the second level of the tree.

Task. The Control Task process manages the execution sequence of Data process tasks. It also manages other applications, e.g., Foreign Control Tasks, and interacts with other business processes which we have denoted Foreign Business Process. For instance, it may launch other ETL processes (not necessarily in the same data warehouse). A Control Task could be further classified as: (i) a Data Process; or (ii) a Foreign Control Task (not shown in Fig. 5). In Fig. 3, DimCategory Load is a data process task, and Create Table is a foreign control task.
Sequence. Gateways and Connections represent relationships between control process objects. Control tasks exchange data using Connections. Control Connections are used by control tasks from the same process, while communication between tasks from different processes use Message flows. A message flow entails all inter-process communication mechanisms such as message queues and web services. A Gateway allows the synchronization and the branching of the flow, by merging multiple ingoing control connections and/or branching multiple outgoing control connections.

Event. Represents something that happens and affects the sequence and timing of the process tasks. A control event could be of the type: Start, Non-boundary, Boundary, and End (which in turn could be further classified). Each type of event enables a set of particular behaviors. For instance, a Start event of type *timer start* may be added to depict the execution schedule of the ETL process. Other common event types include *error*, *message*, or *terminate* events.
Container. A container is either a Swimlane, a Process/Subprocess, or a Loop. The Swimlane is a structuring control object that comprises Pools and Lanes. Subprocesses represent semantically coupled adjacent objects, that accomplish a significant portion or stage of a process. Finally, it is usual that one or more tasks must be executed multiple times, sequentially or not, which is addressed by the Loop container.

5.2 ETL Data Classification Tree

We now discuss our classification of constructs used when creating data processes. We focus on the data task, crucial in the process.
Task. This class corresponds to the Data Task class (see Fig.5). We have identified seven major categories of data tasks. Each category may be further subdivided into subcategories until defining the target data tasks at the leaf level in the tree (further details provided in the next section). These categories are:

1. Variable Derivation. Focus on variable manipulation, where a variable can be a single element or a sequence. Transformations could be: Alter Variable , Add/Delete Variable (modify the input schema by adding or deleting fields), and Variable Extraction (focuses on extracting data from a variable).
2. Branching Flow. Given as input a set of rows, the branching flow tasks deliver multiple output sets of rows. Can be further classified in Row-based

Branching (filters rows based on conditions), and Column-based Branching. In the former, we can also identify the classes Data Condition Branching (where a condition is applied on the row), and Lookup Branching (used to locate the matching records in a reference table).
3. Merging Flow. Includes tasks that combine multiple row sets into a single one. When the input row sets have identical schemes, we have Merge or Union tasks. The Join task allows combining multiple sets of columns into one set of columns using different types of join, such as inner and left joins.
4. Aggregation. Includes the Standard Aggregation task, typical in SQL, and the Custom Aggregation task, which accounts for particular data warehouses such as temporal or spatial warehouses, which use custom aggregation functions.
5. Pivot and Sort. Data input of these tasks are displayed in the output organized differently e.g., sorted and/or pivoted.
6. Data Input. The entry point of data into the process from any possible source of data. Can be a Database Input, a File Input, or a Web Service Input.
7. Data Output. Entails the set of tasks responsible of loading data into external data stores. Can be of types Database Output, and File Output.

The *Sequence* class corresponds to the Data Connection class (recall that in the Control Classification Tree the latter also includes Gateways). This class does not perform any action on the flow, although, it may make some conditions to be activated. The *Event* class corresponds to the Data Event class, which can be of type Intermediate event (aimed at detecting potential exceptions during the data task execution), or Start/End event, which wait for a triggering event to occur, for starting and/or ending a process or sub-process. The other classes are analogous to the ones in the control classification tree.

6 A Metamodel for ETL Processes

We now present the metamodel for ETL processes, based on the BPMN 2.0 metamodel, and in the classification presented in Section 5. Because of the former, several control concepts have been borrowed from the BPMN metamodel, and extended with specific ETL control and data concepts. Each task in a leaf of the trees in Fig. 5 is matched to a concrete class in the metamodel, and tasks in inner nodes are matched to abstract classes in the metamodel. The classification of Section 5 induces two kinds of metamodels: the *control* metamodel, and the *data* metamodel, which we study next. (We depict the metamodel using the Meta Object Facilities (MOF), an OMG standard for developing metamodels. We use the implementation of MOF provided by the Eclipse tree editor[4]).

6.1 Control Metamodel

The control metamodel is depicted in Fig. 6. The ControlProcess class is the main class of the metamodel. It represents an ETL process or a foreign business process respectively depicted by the ETLProcess and ForeignBusinessProcess

[4] `http://wiki.eclipse.org/Ecore`

classes. The ControlProcess class entails several ControlObjects that are linked to each other by means of Connections. For representing business information transferred through connections, a DataObject can be attached to them. Also, a control object is a generalization of the ControlTask, ControlEvent, and Gateway classes. The ControlTask is an autonomous and executable unit of an ETL process. We distinguish between a data process control task, depicted by the DataProcess class (DimGeography_Load in Fig. 3), and a foreign application participating in the ETL process, represented by the ForeignControlTask class (for example Bad_ XML_File in Fig. 3) . The ControlEvent class typically allows defining handling strategies related to particular events that may occur during an ETL process. Based on their position within the process components, four types of event classes are specified: StartEvent, EndEvent, BoundaryEvent, and NonBoundaryEvent classes. Depending on the event type, the event handling strategy can be configured through the eventTrigger property. Examples of handling strategies are: a *cancellation* of the execution flow, and a *message exchange*, which notifies an external application about the event by sending and receiving messages. In addition, other events can be defined in order to orchestrate the task execution, for instance, a *timer*, that delays the execution of the subsequent tasks for a period of time The metamodel in Fig. 6 also includes various subtypes of gateway, the component responsible of managing the execution flow merging and splitting. For instance, the ParallelSplitGateway class models a split gateway able to activate all the outgoing connections once it is activated. Analogously we define the ParallelMergeGateway classto model the synchronization of ingoing connections. The ExclusiveSplitGateway class activates exactly one path at a time based on an exclusive condition, and the InclusiveSplitGateway class activates alternative outgoing paths based on conditions.

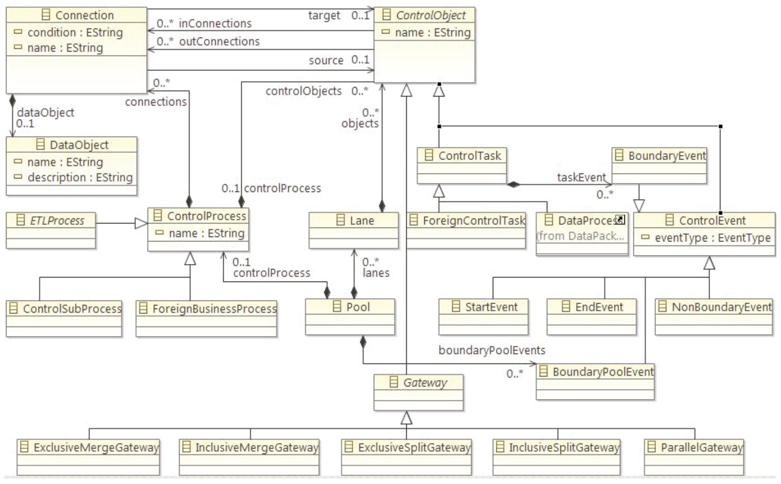
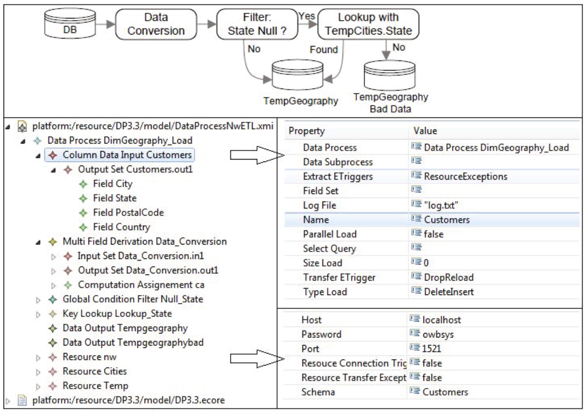


Fig. 6. Control metamodel

6.2 Data Metamodel

Designing full-fledged ETL processes requires extending the BPMN metamodel to support data processes. Therefore, we developed a set of metamodels, namely: Core, Resource, IDS, Condition, Computation, and Query. Due to space limitation we limit ourselves to only explain the Core metamodel in detail.

The *Core metamodel* (Fig. 8) provides the facilities required for modeling data processes that are part of an ETL process. The main class is DataProcess, which aims at providing mechanisms to manage the data extracted from the source until loading them into the data warehouse. A DataProcess class is composed of several DataTask classes for defining different ETL operations that can be applied to data. Fig. 7 depicts a small part of the DimGeography_Load data process of Fig. 4, implemented over the Eclipse Ecore tree editor. It includes the Data Conversion data task. The incoming and outgoing data flows of a task are respectively depicted by the InputSet and OutputSet classes. Both, DataProcess and DataTask classes are related to DataEvent classes to ensure the control and customization of exception handling through triggers using the eventTrigger property. Data tasks sharing the same objective or having similar characteristics may be organized in subprocesses, represented by the DataSubProcess class.



```
platform:/resource/DP3.3/model/DataProcessNwETL.xmi
  Data Process DimGeography_Load
    Column Data Input Customers
      Output Set Customers.out1
        Field City
        Field State
        Field PostalCode
        Field Country
    Multi Field Derivation Data_Conversion
      Input Set Data_Conversion.in1
      Output Set Data_Conversion.out1
      Computation Assignement ca
    Global Condition Filter Null_State
    Key Lookup Lookup_State
    Data Output Tempgeography
    Data Output Tempgeographybad
    Resource nw
    Resource Cities
    Resource Temp
platform:/resource/DP3.3/model/DP3.3.ecore
```

Property	Value
Data Process	Data Process DimGeography_Load
Data Subprocess	
Extract ETriggers	ResourceExceptions
Field Set	
Log File	"log.txt"
Name	Customers
Parallel Load	false
Select Query	
Size Load	0
Transfer ETrigger	DropReload
Type Load	DeleteInsert

Host	localhost
Password	owbsys
Port	1521
Resouce Connection Trig	false
Resource Transfer Except	false
Schema	Customers

Fig. 7. A portion of a data process as seen in the tree editor

The classification presented in Section 5 facilitates the metamodeling of data tasks, and it is also useful for guiding the creation of new tasks. According to such classification, data tasks can be represented as subclasses (depicted in italics in Fig. 8) of the abstract DataTask class. The concrete classes are predefined data tasks. The data task classes yield an intuitive, organized and well-founded task palette to be used for the ETL design. We next describe some of the classes of the metamodel that correspond to the data task classification.

Within the FieldTransformation abstract class, the OneFieldDerivation subclass applies a computation and creates a new field called fName. Similarly, the MultiFieldDerivation class applies a computation for every output field (e.g. the case

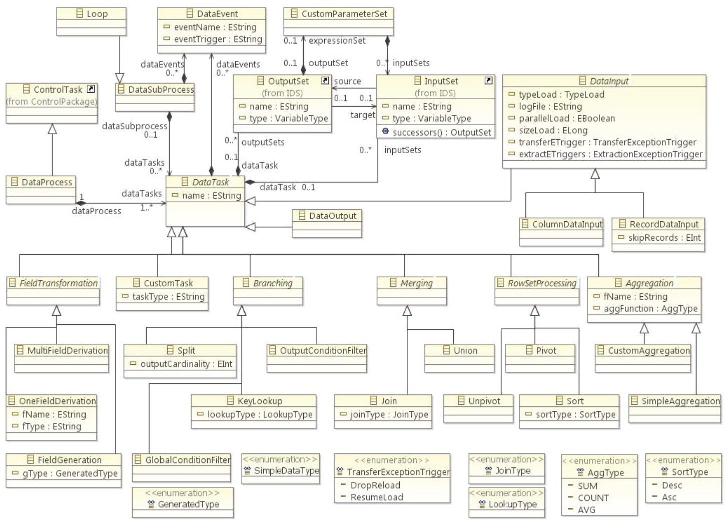


Fig. 8. Core metamodel

of Data Conversion in Fig. 7). The metamodel also specifies how to model the merging tasks from two incoming flows using the Join and Union classes. The Join class requires a condition for defining any kind of joining flows, namely unique, inner, and outer join. On the contrary, the Union class does not require a condition, and simply applies the non-ordered union of rows. These classes are able to determine the fields to merge. Aggregation tasks are handled by the CustomAggregation and SimpleAggregation classes, also referred to in the classification. The DataInput, and DataOutput classes represent the extracting (and loading) tasks from and to the external environment, particularly the databases. Several classes indicate the specific extraction properties to be provided by the user. In this sense, a selectQuery should be provided for the ColumnDataInput class representing the extraction task from a database; or a list of XPath queries xpathQueries should be provided for the extraction task from an XML file, modeled by the XMLDataInput class. Other properties can be associated to these classes, as properties of the parent DataInput class. For example, the property typeLoad allows specifying the type of loading such as update, delete and insert, or advanced update using specific algorithms. This input task can specify the exception handling strategies, depicted by the transferExceptionTrigger, and extractExceptionTrigger properties. Further, parallelLoad in the DataInput class indicates if parallel load is allowed or not. For instance, in the data process of Fig. 7, the data input task properties are depicted on the right hand side, e.g. the transfer trigger is set to DropReload and the type of load to DeleteInsert.

The *Resource metamodel* provides tools to model at the conceptual level both, the data resources and their access information. This is inspired in the metamodels in the Common Warehouse Metamodel (CWM) specification[5]. The rationale is

[5] http://www.omg.org/cgi-bin/doc?formal/03-03-02

that modeling ETL processes should account for the data streaming through the ETL process (non-persistent). Therefore, the designer specifies how the ETL process interacts with the resources through a set of resource configurations. The main concept of the metamodel is the Resource class, representing a real-world data resource. A Resource is composed of FieldSet, which groups a set of Fields (actually the data items). Each Field has a name to identify it, a type (whose value is provided by the SimpleDataType Enumeration), a set of properties to qualify the type of the Field (i.e., scale, precision, and dataFormat), a primaryKey property to recognize if the Field is a key of the FieldSet, and a foreignKey property along with the referenced foreignFieldSet class, to define the field as a foreign key if it corresponds.

The *IDS metamodel* (Intermediate Data Store) models the non-persistent data in an ETL process. This metamodel comprises the OutputSet and InputSet classes. Also, for each InputSet and OutputSet of a data task, the IDSFieldValue provides the values property to represent field values. Therefore, a field can be contained either in a FieldSet for persistent data related to resources, or in InputSet, and OutputSet for non-persistent data related to the process execution.

The *Expression metamodel* supports three types of expressions: computations, conditions, and queries. Creating a technology-independent metamodel for these expressions involves several challenges. On the one hand, ETL models should allow an easy translation into expression languages tailored to some tools. On the other hand, ETL models should be enough abstract and technology-independent. Our framework addresses these challenges by proposing an ETL expression language, where the expression creation follows an OCL-like syntax. The *Computation metamodel* allows defining a large collection of computations related to tasks. A computation consists of a linear function applied on a set of fields in order to generate a new field. The *Condition metamodel* is aimed at specifying any kind of condition in an expression.

Metamodel Consistency OCL rules are built to ensure the metamodel's consistency using the OCLInEcore Eclipse tool. In the following example, an OCL constraint prevents the definition of custom parameters for predefined tasks.

```
1 Not dataTask.oclIsTypeOf(CustomTask) implies
2 dataTask.outputSets.customParameterSets.isEmpty()
```

Line 1 tells that the constraint only concerns tasks other than CustomTask (i.e., predefined data tasks, such as Aggregation and Filter). This constraint forbids the specification of customParameterSets, using the OCL function isEmpty(). For instance, the Aggregation task has parameters fName and aggFunction.

7 Conclusion and Open Problems

We presented a BPMN-based metamodel for producing conceptual models of ETL processes. Conceptual models can be straightforwardly mapped into logical models and implemented in any vendor tool, using Model-to-Text transformations. The framework accounts for ETL components, selected after a study of the most popular ETL tools. These components are addressed from two perspectives: control and data. The use of OMG standards promotes reusability, extensibility, and collaborative work.

Future work includes developing a validation procedure for the models produced using this framework. This will allow to produce a rigourous comparison between the outcome of this methodology, and other ones, not only in terms of workflow structure, but also in terms of flexibility, adaptability to change, usability, and performance. Changes can occur during the lifecycle of the warehouse, not only in sources, but also within the warehouse for example, due to changes in requirements, ETL evolution [6]. accounts for this. The question is: how does this affect the conceptual and logical models for ETL?

References

1. Deutch, D., Milo, T.: A quest for beauty and wealth (or, business processes for database researchers). In: Proceedings of PODS 2011, pp. 1–12 (2011)
2. El Akkaoui, Z., Zimányi, E.: Defining ETL worfklows using BPMN and BPEL. In: Proceedings of DOLAP 2009, pp. 41–48 (2009)
3. El Akkaoui, Z., Zimányi, E., Mazón, J.N., Trujillo, J.: A model-driven framework for ETL process development. In: Proceedings of DOLAP 2011, pp. 45–52 (2011)
4. Muñoz, L., Mazón, J.N., Trujillo, J.: Automatic generation of ETL processes from conceptual models. In: Proceedings of DOLAP 2009, pp. 33–40 (2009)
5. Muñoz, L., Mazón, J.-N., Pardillo, J., Trujillo, J.: Modelling ETL Processes of Data Warehouses with UML Activity Diagrams. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 44–53. Springer, Heidelberg (2008)
6. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Sellis, T., Vassiliou, Y.: Rule-Based Management of Schema Changes at ETL Sources. In: Grundspenkis, J., Kirikova, M., Manolopoulos, Y., Novickis, L. (eds.) ADBIS 2009. LNCS, vol. 5968, pp. 55–62. Springer, Heidelberg (2010)
7. Simitsis, A.: Mapping conceptual to logical models for ETL processes. In: Proceedings of DOLAP 2005, pp. 67–76 (2005)
8. Simitsis, A., Vassiliadis, P.: A methodology for the conceptual modeling of ETL processes. In: Proceedings of CAiSE 2003, CEUR Workshop Proceedings (2003)
9. Simitsis, A., Vassiliadis, P.: A method for the mapping of conceptual designs to logical blueprints for ETL processes. Decision Support System 45(1), 22–40 (2008)
10. Thomsen, C., Pedersen, T.B.: pygrametl: a powerful programming framework for extract-transform-load programmers. In: Proceedings of DOLAP 2009, pp. 49–56 (2009)
11. Thomsen, C.: Pedersen T. B, and W. Lehner. Rite: Providing on-demand data for right-time data warehousing. In: ICDE, pp. 456–465. IEEE (2008)
12. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 307–320. Springer, Heidelberg (2003)
13. Vassiliadis, P., Simitsis, A., Baikous, E.: A taxonomy of ETL activities. In: Proceedings of DOLAP 2009, pp. 25–32 (2009)
14. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S.: A generic and customizable framework for the design of ETL scenarios. Information Systems 30(7), 492–525 (2005)
15. Wyatt, L., Caufield, B., Pol, D.: Principles for an ETL Benchmark. In: Nambiar, R., Poess, M. (eds.) TPCTC 2009. LNCS, vol. 5895, pp. 183–198. Springer, Heidelberg (2009)

Enhancing Coverage and Expressive Power of Spatial Data Warehousing Modeling: The SDWM Approach

Alfredo Cuzzocrea[1] and Robson do N. Fidalgo[2]

[1] ICAR-CNR & University of Calabria, Via P.Bucci 41C 87036 Rende (CS), Italy
cuzzocrea@si.deis.unical.it
[2] Center for Informatics, Federal University of Pernambuco, CDU, Recife (PE), Brazil
rdnf@cin.ufpe.br

Abstract. This paper proposes a novel perspective of research on the challenging issue of modeling Spatial Data Warehouses (SDW) that nicely contributes to improve state-of-the-art proposals. This conveys in the so-called Spatial Data Warehouse Metamodel (SDWM) that allow us to enhance both coverage and expressive power of SDW modeling by means of the following amenities: (*i*) separating the conceptual SDW modeling from the conceptual (spatial) OLAP modeling; (ii) supporting the modeling of complex constructs in SDW; and (iii) stereotyping attributes and measures as spatial objects directly. All these contributions finally depict a novel perspective of research in the investigated scientific field, which breaks the actual trend of state-of-the-art initiatives, by pinpointing their limitations. We complete our analytical contribution by means of a real-life application implemented via SDWM, which highlights the benefits deriving from applying SDWM in contrast with traditional SDW modeling methodologies.

Keywords: Data Models, Spatial Databases, Spatial Data Warehouses.

1 Introduction

The process of decision-making may involve the use of tools, such as Data Warehouse (DW) [1], On-Line Analytical Processing (OLAP) [2], Geographical Information Systems (GIS) [3] and Data Mining [4]. In this context, there are two different types of technologies: one for storing data and other for querying data. That is, from a database (e.g. DW), any query tool (e.g. OLAP, GIS or Data Mining) may be used to analyze its data. Hence, the DW concepts (i.e. fact table, dimension table and attributes) must be independent of those associated to query tools, in particular, with regard to OLAP tools, which has concepts (i.e. hierarchy and level) that are frequently mixed with the concepts of DW modeling (i.e. dimensions and attributes). That is, in DW there are neither hierarchies nor levels, because if these concepts were intrinsic to a DW, any query tool for DW would be able to process multilevel queries (i.e. drill-down and roll-up [2]), but only OLAP query tools can do it. For this reason, in this paper we argue that DW modeling should not mix its concepts with concepts of any query tool and our focus is on concepts and techniques for DW (or Spatial DW – SDW) modeling.

Much research has focused in SDW modeling (see Section 5). However, we identified that most of these works defines metamodels that mix concepts for DW dimensional modeling with concepts for OLAP data cube modeling, which we disagree, because, as already stated, a DW (conventional or spatial) can be analyzed/queried by any data analysis tool (not only OLAP). Moreover, we also note other limitations (also see in Section 5), for example: the most proposed metamodels (*i*) does not support important techniques of DW modeling (e.g. degenerate dimensions, role-playing dimensions and bridge tables) and (*ii*) represents the spatiality in a SDW stereotyping the dimensions and fact tables as spatial or hybrid, rather than directly stereotyping the attributes/measures as spatial. With aim of overcoming such limitations, we have proposed the Spatial Data Warehouse Metamodel (SDWM) [5]. However, due to space restriction, a more elaborated specification of SDWM cannot be made in our previous work. Therefore, specifically in this paper, we present a more detailed and precise definition of SDWM, which is formalized and presented using UML metaclasses.

The remaining of this paper is organized as follows. In Section 2 we briefly present the main concepts and techniques that may be used for DW/SDW modeling. Next, in Section 3, we present a more detailed and precise specification of SDWM. In Section 4 we give an overview about SDWCASE (our tool for modeling DW/SDW according to SDWM) and present a real-life use of our work. Then, in Section 5 we make a brief discussion about relevant works for SDW modeling. Finally, in Section 6 we present some final remarks and indications for future work.

2 Conceptual Background

DW is a typical database that is usually designed using the star model [1]. It has two types of tables: fact and dimension, where a fact table is used to store some metrics of a business and a dimension table to record its descriptive information. Basically, the structure of a fact table is defined by its identifier (formed from a non-empty set of references for dimensions), a possibly empty set of degenerated dimension, a possible empty set of attributes and a possible empty set of measures. The structure of a dimension table is defined by its identifier (a surrogate key [1]), a possibly empty set of references for sub-dimensions (outrigger or mini dimensions [1]) and a non-empty set of descriptive attributes.

There are many techniques/concepts for modeling dimensions and fact tables, for example [1]: fact less fact table, additive facts, semi-additive facts, non-additive facts, degenerate dimensions, role-playing dimensions, bridge tables, slowly changing dimensions, monster dimensions, junk dimensions, heterogeneous dimensions, outriggers dimensions and mini dimensions. Many of these techniques/concepts only provide guidelines for modeling the DW schema. That is, they do not provide additional information (metadata) required to allow a Computer-Aided Software Engineering (CASE) tool to correctly generate code. For example, they do not provide extra and essential information to define table identifiers nor references/links among tables, as well as to create new tables or views. On the other hand, some techniques/concepts bring additional information/metadata required to a CASE tool generates correct code, namely: degenerate dimensions, role-playing dimensions and bridge tables (or many-to-many relationship). That is, a degenerate dimension ensures that it can be used only in a fact table and that it will be part of the identifier of the fact table, but it cannot be a reference/link for a dimension. In turn, a role-playing dimension allows creating different views of the same dimension. Finally, a bridge table (or many-to-many relationship) allows creating a third table with two one-to-many relationships, where we can specify the name of this table and some additional attributes.

A lot of data stored in a DW has some spatial context (e.g., address, ZIP-code, city, state, country). This means that if one intends to properly use this data in decision support systems, it is necessary to consider the use of SDW [6]. A SDW is an extension of the traditional DW through the inclusion of a spatial component in dimension tables (i.e. a spatial attribute) and/or in fact tables (i.e. a spatial measure), where this spatial component (a spatial feature type) has a position (a geometric attribute) more a location (a descriptive attribute), being the latter optional. That is, we can have a feature type with only a geometric attribute.

Modeling languages are defined in terms of their domain concepts and rules (abstract syntax), and the notation used to represent these concepts (concrete syntax). The abstract syntax is defined by a metamodel, which describes the concepts of the language, the relationships between them, and the structuring rules that constrain the combination of model elements according to the domain rules. The concrete syntax specifies how the domain concepts included in the metamodel are represented, and is usually defined by a mapping between the metamodel and a graphical notation. Each model is written in the language of its metamodel. Thus, a metamodel is useful to provide a precise definition of all modeling concepts and the well-formed rules needed for creating syntactically correct models. That is, a metamodel provides facilities for building CASE tools. For example, a CASE tool based on a metamodel can: detect errors earlier or even prevent them from happening, guide toward preferable design patterns, check completeness by informing about missing elements, minimize modeling work by conventions and default values, make full code generation and keep specifications consistent. Moreover, a metamodel also can serve as a basis for integrating and sharing the models/elements with other tools. So, a metamodel is as useful for a modeling language/CASE tool as a grammar is for a programming language/compiler [7][8].

3 The Spatial Data Warehouse Metamodel (SDWM)

SDWM is a metamodel that embeds the following significant features: (*i*) disassociating DW dimensional modeling from OLAP data cube modeling; (*ii*) representing the spatiality in a SDW directly stereotyping attributes/measures as spatial, rather than stereotyping dimension/fact table as spatial or hybrid; (*iii*) capturing whether the geometry of a spatial attribute/measure can be normalized and/or shared; (*iv*) supporting the following DW modeling techniques: degenerated dimension, many-to-many relationship (bridge table) and role-playing dimensions; (*v*) providing a set of stereotypes with pictograms that aims to be concise and friendly; (*vi*) being used as a basic metamodel for a CASE tool that aims to model logical schemas of SDW, as well as to check whether these schemas are syntactically valid. In Figure 1 we introduce SDWM using the UML class diagram. In next paragraphs, based on [9], we give the definitions of our metamodel.

In Figure 1, we have three enumerations, which cover one of the possible valid values for an attribute. The *Cardinality* enumeration represents whether a relationship is *many-to-one, one-to-many* or *many-to-many*. This enumeration is important to define the primary/foreign key (in RDBMS) or OID/REF (in *Object Relational* DBMS). That is, the table on the "*many*" side has a foreign key (or a REF) from the table on the "*one*" side. With respect to *many-to-many* cardinality, we apply the bridge table technique [1], which creates a third table with two *one-to-many* relationships between the original tables. In turn, the *DataType* and *GeometricType* enumerations represent the primitive or spatial data types supported by SDWM, respectively. We highlight that these enumerations are just data type indications, which will be translated for specific data types of a DBMS. Furthermore, we also point out that the spatial data types are based on the *Simple Feature Access* (SFA) specification of *Open Geospatial Consortium* (OGC) [10]. In Definition 1, we present our enumerated types.

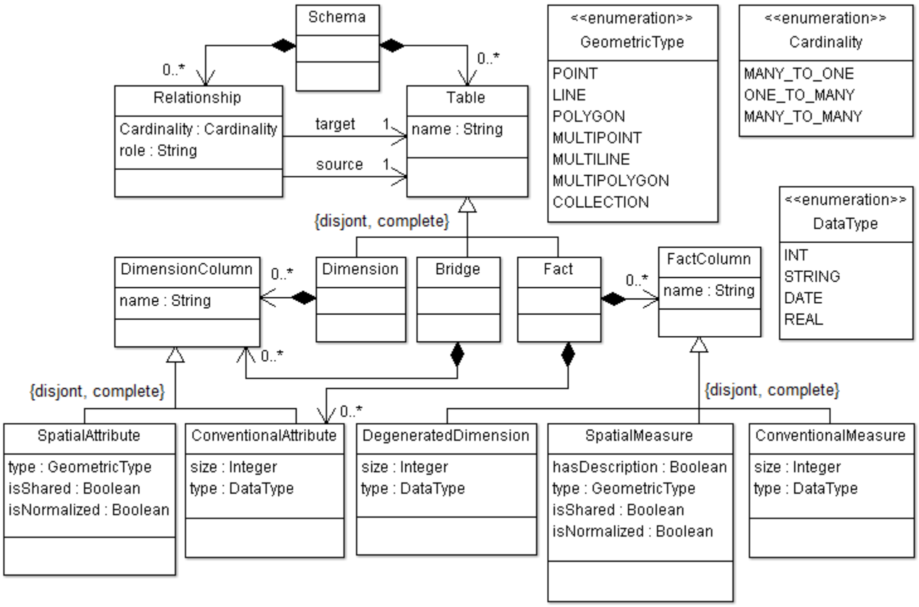


Fig. 1. SDWM Metamodel

Definition 1 – Enumerations. Let *DataType*, *Cardinality* and *GeometricType* be enumerations, such that *DataType* = {INT, STRING, DATE, REAL}, *Cardinality* = {MANY_TO_ONE, ONE_TO_MANY, MANY_TO_MANY} and *GeometricType* = {POINT, LINE, POLYGON, MULTIPOINT, MULTILINE, MULTIPOLYGON, COLLECTION}.

We use domains to state the data types (primitive or symbolic) for the attributes of SDWM. In Definition 2, we introduce the domains that our metamodel uses.

Definition 2 – Domain. A domain is a set of values of some base type *T*. Primitive types comprehend the types **String** (the set of all strings), **Integer** (the set of all integer numbers), and **Boolean** (the set formed by the elements **true** and **false**). Symbolic types comprehend all enumeration types stated in Definition 1.

Our metamodel has five main metaclasses, namely: *Schema*, *Relationship*, *Table*, *DimensionColumn* and *FactColumn*. *Schema* is the root metaclass that corresponds to the drawing area for a SDW schema. For this reason, *Schema* is a composition of zero or more *Table* and zero or more *Relationship*. At last, *FactColumn* and *DimensionColumn* are just a set of different types of columns (in Definition 4, we define these columns). In Definition 3, we present such main metaclasses.

Definition 3 - Main Metaclasses. Let *Schema*, *Relationship*, *Table*, *FactColumn* and *DimensionColumn* be sets, such that $\forall$ *sch:Schema, st:Schema* $\nrightarrow$ *Table and sr:Schema* $\nrightarrow$ *Relationship* $\therefore$ *st(sch)* $\cap$ *sr(sch)* $= \emptyset \land$ *dom(st)* $\cup$ *dom(sr)* = *Schema.*

Besides the main metaclasses, our metamodel has eight specialized metaclasses, namely: *Fact*, *Dimension*, *Bridge*, *SpatialMeasure*, *DegenerateDimension*, *ConventionalMeasure*, *SpatialAttribute* and *ConventionalAttribute*. That is, a *Table* is specialized in *Fact*, *Dimension* or *Bridge*, which capture the concepts of fact table, dimension table and a bridge table, respectively. A *FactColumn* is specialized in *SpatialMeasure*, *DegenerateDimension* and *ConventionalMeasure*, which correspond to a spatial feature type, a descriptive attribute and a measurable attribute, respectively. Finally, a *DimensionColumn* is specialized in *SpatialAttribute* and *ConventionalAttribute*, which represent a spatial feature type and a descriptive attribute. Note that all inheritances are disjoint and complete. Moreover, a *Fact* is a composition of zero or more *FactColumn* and zero or more *ConventionalAttribute*. In turn, a *Dimension* and a *Bridge* are a composition of zero or more *DimensionColumn*. In this context, it is important to point out that a *Dimension* differs from a *Bridge*, a *SpatialMeasure* from a *SpatialAttribute* and a *DegenerateDimension* from a *ConventionalAttribute* because they have different stereotypes, since they represent different concepts and have distinct behaviors. For example: (*i*) a *SpatialAttribute* always has a position (i.e. a geometry) and a location (i.e. a description), while a *SpatialMeasure* only need to have a position, given that it can be defined without a location (*hasDescription* = false) and (*ii*) a *DegenerateDimension* cannot be defined in a dimension table. In Definition 4, we formalize all specialized metaclasses of our metamodel.

Definition 4 - Specialized Metaclasses. Let *Fact*, *Dimension*, *Bridge*, *SpatialMeasure*, *DegenerateDimension*, *ConventionalMeasure*, *SpatialAttribute*, and *ConventionalAttribute* be sets, such that: *Fact* $\subseteq$ *Table*, *Dimension* $\subseteq$ *Table*, and *Bridge* $\subseteq$ *Table* (*Fact* $\cap$ *Dimension* $\cap$ *Bridge* $= \emptyset \land$ *Fact* $\cup$ *Dimension* $\cup$ *Bridge* = *Table*); *SpatialMeasure* $\subseteq$ *FactColumn*, *DegenerateDimension* $\subseteq$ *FactColumn* and *ConventionalMeasure* $\subseteq$ *FactColumn* (*SpatialMeasure* $\cap$ *DegenerateDimension* $\cap$ *ConventionalMeasure* $= \emptyset \land$ *SpatialMeasure* $\cup$ *DegenerateDimension* $\cup$ *ConventionalMeasure* = *FactColumn*), and *SpatialAttribute* $\subseteq$ *DimensionColumn* and *ConventionalAttribute* $\subseteq$ *DimensionColumn* (*SpatialAttribute* $\cap$ *Conventional Attribute* $= \emptyset \land$ *SpatialAttribute* $\cup$ *ConventionalAttribute* = *DimensionColumn*),

$\forall$ *dc:DimensionColumn*, *dcb:DimensionColumn* $\nrightarrow$ *Bridge* and *dcd:DimensionColumn* $\nrightarrow$ *Dimension* $\therefore$ *dcb(dc)* $\cap$ *dcd(dc)* = $\emptyset$ $\wedge$ *dom(dcb)* $\cup$ *dom(dcd) = DimensionColumn*, $\forall$ *fcf:Fact* $\nrightarrow$ *FactColumn* $\therefore$ *dom(fcf) = Fact*, and $\forall$ *fca:Fact* $\nrightarrow$ *ConventionalAttribute* $\therefore$ *dom(fca) = Fact*.

In order to capture the tables that are source and target in a relationship, we have the associations named *Source* and *Target* between *Table* and *Relationship*. In Definitions 5 and 6, we present these associations.

Definition 5 - Source Associations. Let $source_R$ be injective functions from *Relationship* to *Table* ($source_R$: *Relationship* $\rightarrowtail$ *Table*).

Definition 6 - Target Associations. Let $target_R$ be an injective function from *Relationship* to *Table* ($target_R$: *Relationship* $\rightarrowtail$ *Table*).

Attributes are formalized as functions from their metaclasses to the domains they use. In Definition 7, we start by specifying cardinality as an attribute used in metaclass *Relationship*.

Definition 7 - Cardinality Let *cardinality* and *role* be functions from metaclass *Relationship* to domain *Cardinality* (*cardinality*: *Relationship* $\rightarrow$ *Cardinality*).

A dimension can play different roles (views) in a fact table. In this situation we use the technique called role-playing dimension. In Definition 8, we introduce the role attribute.

Definition 8 - Role Let role be a function from metaclass *Relationship* to domain ***String*** (*role*: *Relationship* $\rightarrow$ ***String***).

Another important attribute is *Name*. This attribute stores a label that identifies a metaclass. In Definition 9, we specify *Name* as an attribute used in several metaclasses.

Definition 9 - Name Let $name_T$ be a function from metaclass *Table* to domain **String** ($name_T$: *Table* $\rightarrow$ ***String***), $name_{FC}$ be a function from domain *FactColumn* to domain **String** ($name_{FC}$: *FactColumn* $\rightarrow$ ***String***), $name_{DC}$ be a function from domain *DimensionColumn* to domain **String** ($name_{DC}$: *DimensionColumn* $\rightarrow$ ***String***).

In order to define whether the position of a spatial measure/attribute must be normalized in a different table from its location, the *isNormalized* attribute is defined as a Boolean. That is, whether this attribute is defined as true, the geometric information is normalized in a separate table from the table that stores the descriptive information of the spatial attribute/measure. Otherwise (*isNormalized* = **false**), the geometric information is defined in the same table that stores the descriptive information of spatial attribute/measure. In Definition 10, we present the attribute *isNormalized*.

Definition 10 - isNormalized Let $isNormalized_{SM}$ be a function from *SpatialMeasure* to domain **Boolean** ($isNormalized_{SM}$: *SpatialMeasure* $\rightarrow$ ***Boolean***) and $isNormalized_{SA}$ be a function from *SpatialAttribute* to domain **Boolean** ($isNormalized_{SA}$: *SpatialAttribute* $\rightarrow$ ***Boolean***).

SDWM also allows defines whether the position of a spatial attribute/measure can be shared among several spatial attributes/measures. To accomplish this, it is necessary to define the same name and the same geometric type. Furthermore, for each spatial attribute/measure that will share its geometry, the attributes *isNormalized* and *isShared* must be defined as true. For example, considering an insured dimension and a fact table about accidents, these tables can share the neighborhood geometry where the insured lives and the local that the accident occurs. We also define this attribute using Boolean. In Definition 11, we specify attribute *isShared*.

Definition 11 - isShared Let $isShared_{SM}$ be a function from *SpatialMeasure* to domain **Boolean** ($isShared_{SM}$: *SpatialMeasure* → ***Boolean***) and $isShared_{SA}$ be a function from *SpatialAttribute* to domain **Boolean** ($isShared_{SA}$: *SpatialAttribute* → ***Boolean***).

Spatial measures have the attribute *hasDescription*, which allows to define whether the spatial measure has a location/description (*hasDescription* = true) and a position/geometry or, otherwise (*hasDescription* = false), whether the spatial measure has only a position/geometry. We also define this attribute using Boolean. In Definition 12, we introduce attribute *hasDescription*.

Definition 12 - hasDescription Let *hasDescription* be a function from *SpatialMeasure* to domain **Boolean** (*hasDescription: SpatialMeasure* → ***Boolean***).

Each measure/attribute has an associated type from our allowed data types. These types are captured as attributes of SWDM metaclasses. This is formalized in Definition 13.

Definition 13 - Type Let $type_{SM}$ be a function from *SpatialMeasure* to *GeometricType* ($type_{SM}$: *SpatialMeasure* → *GeometricType*), $type_{DD}$ be a function from *DegeneratedDimension* to *DataType* ($type_{DD}$: *DegeneratedDimension* → *DataType*), $type_{CM}$ be a function from *ConventionalMeasure* to *DataType* ($type_{CM}$: *ConventionalMeasure* → *DataType*), $type_{SA}$ be a function from *SpatialAttribute* to *GeometricType* ($type_{SA}$: *SpatialAttribute* → *GeometricType*), and $type_{CA}$ be a function from *ConventionalAttribute* to *DataType* ($type_{CA}$: *ConventionalAttribute* → *DataType*).

Some metaclasses may have a size to restrict the data types of its columns. Thus we state this in Definition 14.

Definition 14 - Size Let $size_{DD}$ be a function from *DegeneratedDimension* to domain **Integer** ($size_{DD}$: *DegeneratedDimension* → ***Integer***), $size_{CM}$ be a function from *ConventionalMeasure* to domain **Integer** ($size_{CM}$: *ConventionalMeasure* → ***Integer***), and $size_{CA}$ be a function from *ConventionalAttribute* to domain **Integer** ($size_{CA}$: *ConventionalAttribute* → ***Integer***).

SDWM uses stereotypes with pictograms to increase its capacity of expression and visualization. In Figure 2 we show the stereotypes with pictograms to represent the main SDWM constructors.

Stereotype	Pictogram	Description
Fact	Ft	Fact Table
Dimension	Dt	Dimension Table
Bridge	Bt	Bridge Table
ConventionalAttribute	Ⓒ	Conventional Attribute
ConventionalMeasure	C	Conventional Measure
DegeneratedDimension	ⓓ	Degenerated Dimension
SpatialAttribute	Ⓢ	Spatial Attribute
SpatialMeasure	S	Spatial Measure
Relation	⁄	Relation

Fig. 2. SDWM Metamodel Stereotypes

In Figure 3 and Figure 4 we present the stereotypes with pictograms for primitive types and spatial types of SDWM, respectively. Furthermore, we point out that, when *isNormalized* or *isShared* are defined as true the spatial attribute/measure appears in bold or italic font, respectively.

Stereotype	Pictogram	Description
INT	int	Integer type
STRING	str	String type
DATE	dt	Date type
REAL	rl	Real type

Fig. 3. SDWM Primitive Type Stereotypes

Stereotype	Pictogram	Description
POINT	●	Point geometry
LINE	⌇	Line geometry
POLYGON	⏢	Polygon geometry
MULTIPOINT	⁖	Multiple Points geometry
MULTILINE	⌇⌇⌇	Multiple Lines geometry
MULTIPOLYGON	⏢⏢	Multiple Polygons geometry
COLLECTION	▨	Geometry Collection geometry

Fig. 4. SDWM Geometry Type Stereotypes

4 A Real-Life Case Study

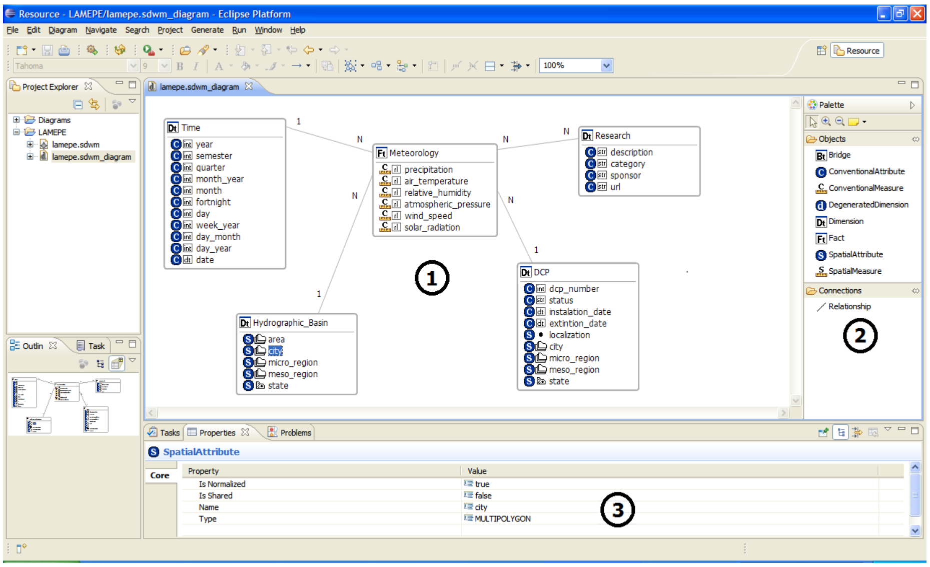
In order to evaluate the correctness and usefulness of our metamodel, we have developed a CASE tool, called SDWCASE, that was used to design a SDW with meteorological data from the Laboratory of Meteorology of Pernambuco (LAMEPE). This laboratory has a net of meteorological Data Collection Platform (DCP) for monitoring atmospheric conditions. Meteorological data are important information for predictions about the occurrence and volume of rainfall in the state of Pernambuco.

SDWCASE is a CASE tool for specifying a SDW schema that offers a concise and friendly GUI that is based on the stereotypes with pictograms of SDWM (see Figures

2, 3 and 4). With our CASE tool, the designer can interact with the SDW schema by inserting, excluding, editing, visualizing at different zoom levels, exporting a figure (e.g. JPG, GIF, PNG) or XMI file. Moreover, SDWCASE also allows the validation of the modeled schema. For example, (*i*) two tables (dimension or fact) or two attributes (in the same table) cannot have the same name; (*ii*) a table cannot be isolated or associated with itself; (*iii*) measures and degenerated dimensions can only exist in a fact table; and (*iv*) dimension tables and bridged tables can only have attributes. The first and the second validations are ensured by programming, but the third and the fourth are intrinsically/automatically ensured by our metamodel (see Figure 1). SDWCASE is implemented using Java/Eclipse technologies [11] (i.e. Eclipse Modeling Framework (EMF), Graphical Modeling Framework (GMF) and Epsilon Framework), which are conform to the Essential Meta Object Facility (EMOF) [12] standard and, in its current version, it generates code only for PostgreSQL with PostGIS. However, it can be done for any spatial DBMS. For this, basically, we need to develop a new compiler considering the data types and reserved words of a specific SQL/DDL.

In Figure 5 we show the SDWCASE GUI with the LAMEPE SDW using many-to-many relationship. The SDWCASE GUI has a palette (area 2 in Figure 5) with all elements (defined in SDWM) that the designer needs to model a SDW. The modeling tasks starts with a click on the desired element in the palette and place it in the drawing area (area 1 in Figure 5). Next, the designer may edit the properties of the element (area 3 in Figure 5), and add new elements or relationships. Note that (*i*) each element is easily identified by its pictogram and (*ii*) the SDW schema is concise. That is, only using spatial attributes/measures, we can represent the spatiality in a SDW with a short notation.

Also, in Figure 5 we have one fact table and four dimension tables. The fact table (*Meteorology*) has only conventional measures and the dimension tables (*Time*, *Hydrographic Basin*, *Localization*, *DCP*, *Research* and *Researcher*) have the following spatial attributes: dimension *Localization - city*, *micro_region*, *meso_region* and *state* (attributes of type *MultiPolygon*); dimension *Hydrographic Basin - area* (attribute of type *MultiPolygon*); and dimension *DCP - localization* (attribute of type *Point*). In Figure 5 there is a many-to-many relationship between the fact table *Meteorology* and the dimension *Research*. In this case, our CASE tool abstracts the creation of a third table to implement this relationship. However, an explicit bridge table also can be defined in SDWCASE, where we can specify the name of this table and some additional attributes (e.g. *weighting_factor* and *primary_indicator*). In Figure 6 we redefine this many-to-many relationship as an explicit bridge table (*Research_Collection*) and, to illustrate a full use of our metamodel and CASE tool, we apply the following concepts: role-playing dimensions, spatial measure, degenerated dimension and conventional attribute. For this, we remove the DCP dimension and redefine its attributes *installation_date* and *extintion_date* as two role-playing dimensions, as well as its attributes *location*, *DCP_number* and *status*, respectively, as a spatial measure, a degenerated dimension and a conventional attribute in that fact table.



Property	Value
Is Normalized	true
Is Shared	false
Name	city
Type	MULTIPOLYGON

Fig. 5. SDWCASE GUI with LAMEPE SDW using many-to-many relationship

In SDWCASE the definition of primary keys and foreign keys for each table is abstracted from designer. That is, a primary key correspond to a surrogate key that is created automatically at the moment of code generation and foreign keys are defined from the cardinality of the relationships. That is, the table on the "many" side has a foreign (in RDBMS) or REF (in *Object Relational* DBMS) from the table on the "*one*" side. For role-playing dimensions we create a view for each one and put its primary key as a foreign key in fact table. In the case of spatial attribute/measure defined as normalized, its position (i.e. geometry) is normalized in a different table from its location (i.e. description), where this new table exports its primary key (also a surrogate key) to be a foreign key in original table. In this context, it is important enhance that in SDWCASE the position of a spatial attributes/measure can be shared among several spatial attributes/measures. To accomplish this, it is necessary to define the same name and the same geometric type. Furthermore, for each spatial attribute/measure that will share its position, the attributes *isNormalized* and *isShared* must be defined as true.

Once finished the SDW modeling, the designer may validate its schema using our CASE tool. The validation was implemented using the language EVL (Epsilon Validation Language) [13]. Once validated, the designer may automatically generate the SQL/DLL code for the modeled schema. The generation code for SQL/DDL scripts is implemented using the Epsilon Generation Language (EGL) [13]. Due to scope and space limitations, a detailed/completed specification of SDWCASE is outside of this paper.

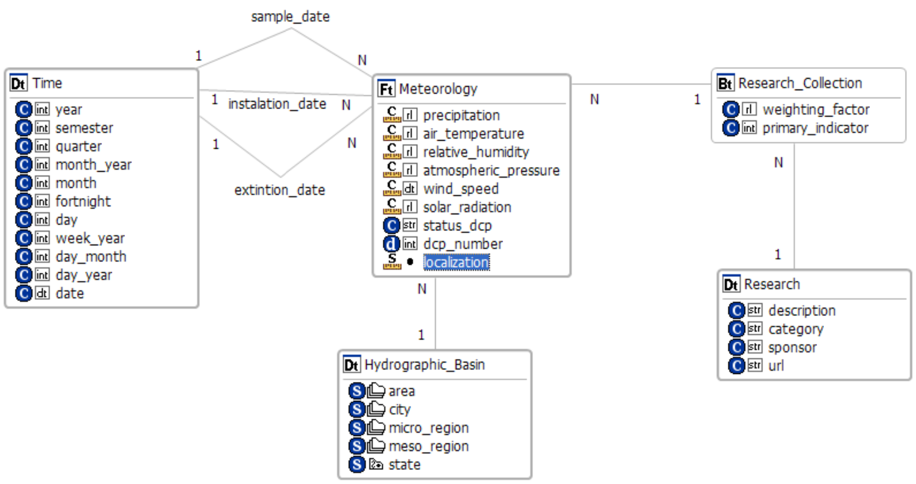


Fig. 6. LAMEPE SDW using bridge table, degenerated dimension and role-playing dimension

5 Related Work

In this Section, we present and evaluate the most popular and important works related to metamodel for SDW. In order to do a systematic evaluation of these works, we are using the following features:

1. disassociating DW dimensional modeling from OLAP data cube modeling;
2. making a CASE tool available to users;
3. supporting the following DW modeling techniques: degenerated dimensions, many-to-many relationships and role-playing dimensions;
4. supporting spatial attributes rather than spatial or hybrid dimension;
5. supporting spatial measures;
6. providing a set of stereotypes with pictograms that aim to be concise – i.e., it provides a short notation;
7. capturing whether the geometry of a spatial attribute/measure can be normalized and/or shared.

Han et al. [14][15] define three types of spatial dimensions: non-geometric spatial dimension, geometric spatial dimension and mixed spatial dimension. The first type has only descriptive/conventional levels, the second one has only geometric levels, and the third one has both geometric and conventional levels. The authors also make a distinction between numerical and spatial measure, where the latter is a collection of geometries. As we can note, this work mixes the concepts of DW and OLAP modeling and does not support the concept of spatial attributes, consequently the position of a spatial attribute cannot be normalized and/or shared. Furthermore, this work does not support DW modeling techniques and no metamodel or CASE tool was proposed.

Fidalgo et al. [16][17][18] define a metamodel and a CASE tool for SDW modeling. Similarly the previous work, the authors specify concepts of measures (conventional and spatial) and dimensions (conventional, spatial and hybrid). However, the metamodel and the CASE tool, although support the degenerated dimension technique, do not support: spatial attributes, many-to-many relationships and role playing dimensions. That is, the position of a spatial attribute cannot be defined as normalized and/or shared. Moreover, the use of dimensions stereotyped with spatial pictograms does not provide a concise/short notation, because it defines one dimension for each spatial information, which pollutes the SDW schema whether it has much spatial information.

Malinowski and Zimányi [19][20] define an extension of ER model to represent dimensions, hierarchies and spatial measures/levels. The extension makes use of classes and relationships, both stereotyped with spatial pictograms, in order to model the geometry of spatial levels and the topological relationships between these levels. As already stated, we stress that the use of dimensions stereotyped with spatial pictograms does not provide a concise notation, because it also defines one class for each spatial level, which pollutes the SDW schema whether it has many spatial information. Moreover, as we can observe, this work also mix DW modeling with OLAP modeling and, although the authors define a metamodel, it does not support spatial attributes neither DW modeling techniques, it does not capture whether the geometry of a spatial attribute must be normalized and/or shared and it is not used as a basic metamodel for a CASE tool.

Glorio and Trujillo [21][22] define an UML profile and a CASE tool that use a set of stereotypes with pictograms for dimensions, hierarchies and spatial measures/levels. As we can note, both metamodel and CASE tool, also mix DW modeling with OLAP modeling and do not consider the concept of spatial attribute. Consequently, this work does not capture whether the geometry of a spatial attribute must be normalized and/or shared. Moreover, although this work supports the technique degenerated dimension, it does not support many-to-many relationships neither role playing dimensions. This work provides a set of spatial stereotypes with pictograms. However, the authors represent a spatial level as a stereotyped class and, similar to the last two works, this does not provide a short notation. That is, this work defines one stereotyped class for each spatial level, which, as already stated, pollutes the SDW schema whether it has much spatial information.

As we can note, these works support spatial measure, but they do not support spatial attributes. Consequently, they cannot capture whether the geometry of a spatial attribute/measure must be normalized and/or shared. Moreover, only Fidalgo et al. [16][17][18] do not mix DW modeling with OLAP modeling, as well as, only Fidalgo et al. [16][17][18] and Glorio and Trujillo [21] [22] support degenerated dimension technique, but these works do not support many-to-many relationships neither role playing dimensions techniques. Finally, although these works provide a set of spatial stereotypes with pictograms, they represent the spatial information as a stereotyped class, which does not provide a concise/short notation, because it pollutes the SDW schema whether it has much spatial information. In Table 1 we compare our work with the related works discussed in this Section.

Table 1. Analysis of related works and our proposal

	Han et al.	Fidalgo et al	Malinowski and Zimányi	Glorio and Trujillo	Our Proposal
DW vs. OLAP Modeling	NO	YES	NO	NO	YES
CASE Tool	NO	YES	NO	YES	YES
Degenerated Dimensions	NO	YES	NO	YES	YES
M-N Relationships	NO	NO	NO	NO	YES
Role-Playing Dimensions	NO	NO	NO	NO	YES
Spatial Attributes	NO	NO	NO	NO	YES
Spatial Measures	YES	YES	YES	YES	YES
Short notation	NO	NO	NO	NO	YES
Normalized/Shared Geo.	NO	NO	NO	NO	YES

6 Final Remarks

Metamodels play a fundamental role in modeling language definition and CASE tool creation, because it describes the constructors and the valid constructions of a modeling language/CASE tool. That is, a metamodel specifies modeling elements, their relationships and their well-formed rules, disallowing the specification of incorrect or ambiguous models [7][8]. Many proposals have focused in metamodel and/or CASE tool for SDW. However, most of these works defines metamodels that (*i*) mixes concepts of DW dimensional modeling (i.e. dimensions and its attributes) with concepts of the OLAP data cube modeling (i.e. hierarchy and its levels); (*ii*) does not support important techniques of DW modeling (e.g. degenerate dimensions, role-playing dimensions and/or bridge tables), (*iii*) represents the spatiality in a SDW stereotyping the dimensions and fact table as spatial or hybrid, rather than directly stereotyping the attributes/measures as spatial; (*iv*) defines a complex taxonomy of spatial dimensions and measures, (*v*) does not provide a concise and friendly set of stereotypes with pictograms; and/or (*vi*) is not used as a basic metamodel for a CASE tool that aims to model logical schemas of SDW, as well as to check the consistency/integrity of these schemas.

In this paper, in order to give a contribution to solve this problem, we have presented a more detailed and precise definition of SDWM, which describes the constructors and the restrictions needed to design SDW schemas that are consistent and unambiguous. Our metamodel is more straightforward and more expressive than its related works, because it (*i*) represents the spatiality in a SDW assigning spatial stereotypes directly in attributes and measures, (*ii*) disassociates the DW dimensional modeling from the OLAP data cube modeling, (*iii*) captures whether the geometry of a spatial attribute/measure can be normalized and/or shared, (*iv*) proposes a set of stereotypes with pictograms that aims to provide a short/concise notation, and (*iv*) supports the following DW modeling techniques: degenerated dimension, bridge table and role-playing dimensions. For this, SDWM can be used as a basic metamodel for a CASE tool that aims to provide facilities to make the design of invalid or incorrect SDW schema much harder, as well as to make the automatic SQL/DDL code generation from these schemas. To the best of our knowledge, our proposal is the first one to gather these features, depicting a novel perspective of research, since our work breaks the actual trend of state-of-the-art.

To evaluate our proposal, SDWM has been implemented in a CASE tool and tested with a real-life application that illustrates a full use of our metamodel, demonstrating that the semantic and syntax of our metamodel are modeled correctly, and its notation is unambiguous. The CASE tool is named SDWCASE. It is implemented in Java using EMF, GMF and Epsilon Frameworks, which are conform to the EMOF standard. In its current version, SDWCASE generates SQL/DDL code for PostgreSQL with PostGIS. In future work, other spatial DBMS will also be covered. Other direction for future work is the: definition of a metamodel and CASE tool to model and query a Spatial OLAP cube.

Acknowledgment. This work was supported by the process BEX 5026/11 CAPES/DGU from the CAPES Brazilian foundation.

References

[1] Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., Becker, B.: The Data Warehouse Lifecycle Toolkit, 2nd edn. John Wiley & Sons (2008)

[2] Thomsen, E.: OLAP Solutions: Building Multidimensional Information Systems, 2nd edn. John Wiley & Sons (2002)

[3] Heywood, D.I., Cornelius, M.S., Carver, D.S.: An Introduction to Geographical Information Systems, 3rd edn. Prentice-Hall (2006)

[4] Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann (2011)

[5] Del Aguila, P.S.R., Fidalgo, R.N., Mota, A.: Towards a more straightforward and more expressive metamodel for SDW modeling. In: Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP, pp. 31–36 (2011)

[6] Damiani, M.L., Spaccapietra, S.: Spatial data warehouse modelling. In: Darmont, J., Boussaid, O. (eds.) Processing and Managing Complex Data for Decision Support, pp. 12–27. Idea Group Publishing (2006)

[7] Kelly, S., Tolvanen, J.-P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Pr. (2008)

[8] Moreno, N., Romero, J.R., Vallecillo, A.: An Overview of Model-Driven Web Engineering and the Mda. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling and Implementing Web Applications, pp. 353–382. Springer, London (2008)

[9] Schmitt, P.H.: UML and its Meaning (2003), http://formal.iti.kit.edu/~beckert/teaching/Spezifikation-SS04/skriptum-schmitt.pdf

[10] Open Geospatial Consortium Inc, OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture (2006)

[11] Eclipse.org, Eclipse Modeling Project, http://www.eclipse.org/modeling/ (accessed: April 2012)

[12] OMG, Meta Object Facility (MOF) Core Specification - Version 2.4.1 (2011), `http://www.omg.org/spec/MOF/2.4.1/PDF/`

[13] Kolovos, D., Rose, L., Paige, R.: The Epsilon Book. Eclipse (2011)

[14] Stefanovic, N., Han, J., Koperski, K.: Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Transactions on Knowledge and Data Engineering 12(6), 938–958 (2000)

[15] Bédard, Y., Merrett, T., Han, J.: Fundamentals of spatial data warehousing for geographic knowledge discovery. Geographic Data Mining and Knowledge Discovery 2, 53–73 (2001)

[16] Fidalgo, R.N., Times, V.C., da Silva, J., Souza, F.F.: GeoDWFrame: A Framework for Guiding the Design of Geographical Dimensional Schemas. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 26–37. Springer, Heidelberg (2004)

[17] Times, V.C., Fidalgo, R.N., da Fonseca, R.L., da Silva, J., de Oliveira, A.: A Metamodel for the Specification of Geographical Data Warehouses. In: Kozielski, S., Wrembel, R., Sharda, R., Voß, S. (eds.) New Trends in Data Warehousing and Data Analysis, vol. 3, pp. 1–22. Springer, US (2009)

[18] da Silva, J., de Oliveira, A.G., Fidalgo, R.N., Salgado, A.C., Times, V.C.: Modelling and querying geographical data warehouses. Information Systems 35(5), 592–614 (2010)

[19] Malinowski, E., Zimányi, E.: Logical Representation of a Conceptual Model for Spatial Data Warehouses. GeoInformatica 11(4), 431–457 (2007)

[20] Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications (Data-Centric Systems and Applications). Springer (2009)

[21] Glorio, O., Trujillo, J.: An MDA Approach for the Development of Spatial Data Warehouses. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2008. LNCS, vol. 5182, pp. 23–32. Springer, Heidelberg (2008)

[22] Glorio, O., Trujillo, J.: Designing Data Warehouses for Geographic OLAP Querying by Using MDA. In: Gervasi, O., Taniar, D., Murgante, B., Laganà, A., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2009, Part I. LNCS, vol. 5592, pp. 505–519. Springer, Heidelberg (2009)

Sprint Planning Optimization in Agile Data Warehouse Design

Matteo Golfarelli, Stefano Rizzi, and Elisa Turricchia

DEIS - Univ. of Bologna,
V.le Risorgimento 2, 40136 Bologna, Italy
{matteo.golfarelli,stefano.rizzi,elisa.turricchia2}@unibo.it

Abstract. Agile methods have been increasingly adopted to make data warehouse design faster and nimbler. They divide a data warehouse project into sprints (iterations), and include a sprint planning phase that is critical to ensure the project success. Several factors impact on the optimality of a sprint plan, e.g., the estimated complexity, business value, and affinity of the elemental functionalities included in each sprint, which makes the planning problem difficult. In this paper we formalize the planning problem and propose an optimization model that, given the estimates made by the project team and a set of development constraints, produces an optimal sprint plan that maximizes the business value perceived by users. The planning problem is converted into a multi-knapsack problem with constraints, given a linear programming formulation, and solved using the IBM ILOG CPLEX Optimizer. Finally, the proposed approach is validated through effectiveness and efficiency tests.

Keywords: Agile methods, Optimization, Data warehouse design.

1 Introduction

As empirical studies suggest [9,2], agility is one of the most promising directions to overcome the problems of traditional software engineering approaches. The twelve principles stated in the Agile Manifesto [3] are followed by several agile methods, such as *Scrum* and *eXtreme Programming*, that have been adopted by an increasing number of companies to make the software development process faster and nimbler. Agile principles also find large application for designing data warehouses (DWs), that are characterized by a particularly long and expensive development process, so some agile approaches to DW design have been devised in recent years [13,11].

A key practice shared by all agile methods is incremental and iterative design and implementation. The DW system is described in terms of detailed user functionalities (*user stories*) [13]; a user story can correspond for instance to a set of correlated reports, a piece of ETL, or a conceptual schema for a fact. At each iteration (*sprint* in the Scrum terminology), the team should deliver the set of user stories that maximizes the utility for the users and fulfills a set of development constraints [22]; typical constraints include limiting the duration of an iteration, respecting dependencies and correlations among user stories (e.g., logical design must follow conceptual design), reducing the non-delivery risk.

Clearly, the sprint planning phase is critical to ensure the project success. User story prioritization and definition of sprint boundaries are obtained by sharing and averaging the estimates given by the different team members about story complexity, utility, and dependencies. For example, advancing high-valued stories (e.g., those that implement critical analysis reports) could lead to an early significant result for users; similarly, risky user stories (e.g., those that implement cleaning procedures for very dirty data) can be advanced to avoid late side-effects, but at the price of a higher probability of delays in the initial stages. The success of a sprint planning phase mainly depends on the accuracy of estimates and on the capability of properly taking several variables and constraints into account. While the first issue is mainly related to the team experience, the second one can be formulated as an optimization problem whose complexity increases with the project size. Clearly, a non-optimal solution to this problem leads to inefficiencies that easily turn into extra-costs and project delays.

Though some commercial tools support agile project management for generic software systems [23,8], they do not provide any support to optimal sprint planning. In this direction, in this paper we formalize the sprint planning problem for DW projects and propose an optimization model that, given the team estimates and a set of development constraints, produces an optimal sprint plan that maximizes the business value perceived by users. Remarkably, though our approach fully complies with the agile principles that give the team experience and knowledge a key role in delivering an effective plan, it also relieves the team from the difficult task of quickly producing an optimal plan. The optimization problem is formalized as a multi-knapsack problem with constraints, and is solved using the IBM ILOG CPLEX Optimizer [14].

The paper is organized as follows. Section 2 reviews the related literature; Section 3 summarizes the key agile practices in DW design; Section 4 formalizes the sprint planning problem and proposes an optimization model that takes agile principles into account; Section 5 presents a set of tests on both synthetic and real projects to prove efficiency and effectiveness of our approach; Section 6 draws the conclusions and sketches our future work.

2 Related Literature

A pioneering work in the field of agile DW design is [13], that breaks with strictly sequential approaches by applying two Agile development techniques, namely *scrum* and *eXtreme Programming*, to the specific challenges of DW projects. To better meet user needs, the work suggests to adopt a *user stories decomposition* step based on a set of architectural categories for the back-end and front-end portions of a DW. In [11] the potential advantages arising from the application of modern software engineering methodologies to a DW project are analyzed, and a design methodology called *Four-Wheel-Drive* (4WD) is proposed. 4WD aims at making the DW design process more reliable, robust, productive, and timely; to this end it adopts six key principles (incrementality, prototyping, user involvement, component reuse, light documentation, and automated schema transformation), most of which perfectly fit the agile paradigm.

More generally, in the software engineering field several approaches inspired by agile principles have been proposed [2]. In [9], the authors propose a systematic review and comparison of different agile methods, focusing on both organizational and technical features. They also emphasize the increasing penetration of Scrum and Extreme Programming practices in industries. The Scrum approach is deeply discussed in [22], where its key ideas and its life-cycle are described. A more pragmatic work is presented in [7], that focuses on user stories and gives practical hints for estimating their complexity and business value.

As to tools for agile project management, a few solutions are available. AgileFant [1] offers a set of basic functionalities to monitor the progress of project iterations. *Mingle* [23] and *ScrumWorks* [8] provide a more complete set of agile parameters to deal with user story risk, complexity, and business value. However, all these solutions lack in providing an automated solution to the sprint optimization problem; similarly, to the best of our knowledge, no research prototypes have been developed to this purpose.

In the broader context of project scheduling, several models and algorithms have been proposed in the literature (see [15] for a survey). According to the classifications proposed in [12] and [5], our problem is categorized as resource-constrained with renewable resources (i.e., manpower) available on a period-by-period basis. As in the basic PERT/CPM model, finish-start precedences with zero time lag are considered and no preemption of activities is allowed.

The project scheduling literature provides no model that perfectly fits the problem discussed here, and this is where operational research comes into play. Sprint optimization can be formulated as a multi-knapsack problem [17], where sprints are the knapsacks, while user stories are the items. The sprint optimization problem is made original by its objective function and the way how affinity and risk affect the solution. Though the multi-knapsack problem is NP-hard [10], branch-and-bound techniques can efficiently compute exact solutions for medium-sized instances. For large problems, heuristic methods can be used to find approximate results.

3 A Summary of Agile Data Warehouse Design Practices

The success of a DW project is directly related to customer satisfaction, so agile methods strive to better comply with user requests. In particular, agile principles aim at reducing the delivery time and making the development process more flexible; indeed, accelerating the time-to-market leads to overcoming the business pressure, while flexibility ensures fast reactions to both technology evolution and user requirement changes. To achieve these goals, agile methods propose several complemental practices:

– *Incremental process*: The DW system is broken up into smaller portions which are scheduled, developed, and integrated when completed; each portion represents an increment in business functionality, that users can validate. For instance, 4WD is based on nested iteration cycles: a *data mart cycle* that defines and maintains the global plan for the development of the whole DW and, at each iteration, designs and releases one data mart; a *fact cycle* that refines the data mart plan and incrementally designs and releases its facts; a *modeling* and an *implementation* cycle that include the activities for delivering reports and applications concerning a single fact [11].

– *Iteration*: The DW system is built in iterations, where each cycle expands the product until the project is completed. Since the process is also incremental, each iteration includes analysis, design, coding, and testing. Noticeably, a stepwise refinement based on short iterations increases the quality of projects by supporting rapid feedback and quick deliveries [4,18].
– *User involvement*: Analysis specifications are difficult to be understood during the preliminary life-cycle phases. Continuous interaction with users is promoted to progressively refine the specifications, reduce inadequate requirements, and increase the trust between users and developers. In more general terms, a user-centered design increases customer satisfaction [11].
– *Continuous and automated testing*: To facilitate requirement validation and obtain better results, a DW is developed by refining and expanding an evolutionary prototype that progressively integrates the implementation of each increment [20]. Unit tests are written for each release of the prototype and automated tests are used whenever possible to accelerate error detection.
– *Lean documentation*: A well-defined documentation is a key feature to comply with user requirements. Small and simple formal schemata are preferred to extensive specifications; thanks to continuous user involvement, up-to-date and clear documentation can be achieved [16,21].

Figure 1 shows the general life-cycle of an agile DW project. Depending on the specific methodology adopted, the macro-analysis returns a high-level description of the requirements in terms of facts to be designed, functional areas to be covered, or analysis applications to be developed. The project team and the users break these requirements into user stories and assign a utility and a development complexity (measured in terms of *story points*) to each of them. Typical examples of user stories include: one or more forms for manual input of data to ETL; a report or a group of related reports; the conceptual schema of a fact or a conformed hierarchy; an ETL unit; the glossary for a functional area; a security profile.

Then, the team assigns a priority to each user story and defines possible correlations (*affinities*) among stories. The resulting list composes the *data warehouse backlog*, that must be partitioned into sprints to produce a *plan*. Sprints should be short and regular enough to guarantee a prompt feedback from users. During each sprint, the team carries out a micro-analysis of the user stories involved, then its members take charge of one or more user stories that are then designed, implemented, and tested. After closing a sprint, the users verify if the stories developed match the requirements they expressed. The approved stories are delivered, while the remaining ones are reinserted in the backlog; noticeably, new requirements may arise at this stage from user feedbacks.

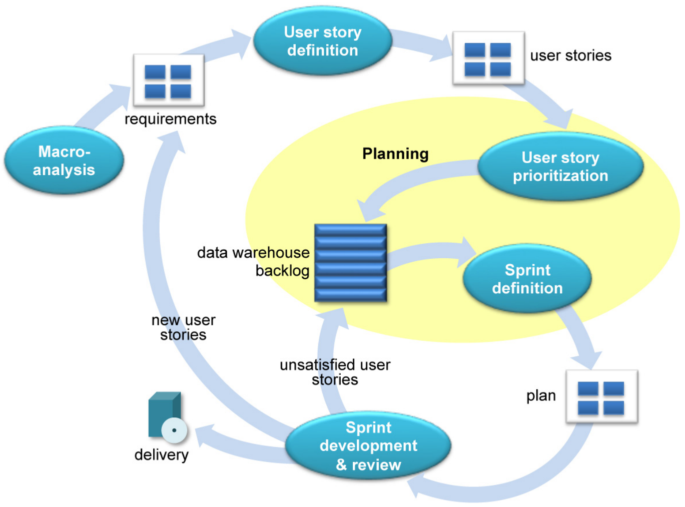


Fig. 1. Agile life-cycle for DW design

4 An Optimization Model

Our formulation of the sprint planning problem takes into account the main variables that affect user stories prioritization and sprint composition. The underlying concepts are:

- *Plan*: a sequence of sprints. All modern DW design methodologies agree on incrementally releasing one data mart at a time, so we will assume the scope of a plan is that of a data mart.
- *Sprint*: the time-bound unit of iteration, typically a one- to four-week period, depending on the project complexity and risk assessment. A sprint includes a set of user stories, and it normally ends with a delivery. A maximum duration is fixed for each sprint by the project team.
- *User story*: a relatively small piece of functionality valuable for users [7]. It represents a light specification that can be later detailed thanks to a continuous communication with the user, but at the same time it must be sufficiently described to estimate its development complexity.
- *Utility*: the business value of a user story as it is perceived by the user that defines it. In general it is quantified through a positive numerical score (typically ranging between 10 and 100 [19]).
- *Story point*: a unit of measurement for the development complexity of user stories. Team members assign story points to each user story based on their experience and knowledge of the domain and project specificities. Story points are non-dimensional and are preferred to time/space measures to avoid subjective and incomparable estimates. Typical complexities of user stories range between 1 and 10 story points [19].

- *Risk*: the risk that the project is not completed as desired. We consider risks related to two different characteristics of user stories: (i) A *critical* story is one that has a strong impact on the other stories, so that taking a wrong solution for it can dramatically affect the success of the project (e.g., the conceptual design of a conformed dimension); (ii) An *uncertain* story is one for which it is somehow hard to estimate the complexity due to unexpected problems that could arise (e.g., changes in the analysis requirements or faulty/incomplete source data). Both types of risk are estimated by positive numbers; here we adopted four classes of risk: 1 (no risk), 1.3 (low risk), 1.7 (medium risk), and 2 (high risk).
- *Affinity*: the degree of correlation between user stories. Similar stories have higher utility if they are included in the same sprint, because users better perceive the overall business value of the functionality delivered. For instance, an "incremental data extraction" story may have low utility on its own, but its utility increases if delivered together with the complemental "incremental data loading" story. The affinity range we adopted is [0, 0.5], meaning that the utility of a story can be increased at most by 50%.
- *Dependence*: a development constraint between two user stories, indicating that a user story (post-condition) cannot start before the other (pre-condition) is completed. Though agile methods discourage user story relationships to improve the project flexibility, some development dependencies must necessarily be preserved (e.g., logical design must follow conceptual design). The *dependency type* of a user story takes value AND (all the pre-condition stories must be completed) or OR (at least one of the pre-condition stories must be completed).
- *Development speed*: the estimated number of story points the team can deliver per day. It is used to convert the sprint duration into the *sprint capacity* (i.e., the maximum number of story points the team can deliver in a sprint).

We can now list the goals an optimal plan should pursue:

♯1 *Customer satisfaction.* It can be obtained by delivering user stories with higher utility first. In the agile philosophy, this also increases the user awareness and trust.

♯2 *Affinity management.* Similar stories should be carried out in the same sprint to increase their value for users.

♯3 *Risk management.* It can be achieved by (i) advancing critical user stories to avoid late side-effects, on the one hand; (ii) distributing uncertain stories in different sprints and postponing them to reduce the risk that the sprint delivery is delayed, on the other hand.

Besides, all constraints related to the sprint capacity and inter-story dependencies must obviously be met.

The problem of determining an optimal plan, i.e., one that achieves these goals, can be converted into a multi-knapsack problem [17], where the knapsacks are the sprints and the items are the stories. Story points measure the weight of an item, while utility represents its value. Knapsack capacity is measured as the story points that the team can deliver given the sprint duration and the team development speed, i.e., as the sprint capacity. The objective function to be maximized is the cumulative utility of the project (goal ♯1), where the utility of each story is increased if some similar stories are included in the same sprint (goal ♯2) and/or if that story is critical (goal ♯3-i). Finally, in the formulation of the capacity constraint, the story points of user stories are increased by their uncertainty, which discourages the inclusion of two uncertain stories in the same sprint (goal ♯3-ii). The multi-knapsack problem is NP-hard [10]; the linear programming formulation we adopt is shown in the following.

Definition 1 (Sprint Planning Problem). *Given a set of m sprints S and a set of n user stories U, let:*

- $x_{ij} = 1$ *iff story j is included in sprint i, 0 otherwise;*
- u_j *be the utility of story j;*
- p_j *be the number of story points of story j;*
- p_i^{max} *be the capacity of sprint i, measured in story points;*
- r_j^{cr} *be the criticality risk of story j;*
- r_j^{un} *be the uncertainty risk of story j;*
- a_j *be the affinity of story j;*
- $Y_j \subset U$ *be the set of stories similar (i.e., with some affinity) to story j;*
- y_{ij} *be an accessory variable related to the number of stories in Y_j included in sprint i;*
- $D_j \subset U$ *be the set of stories the story j depends on;*
- $U \supseteq U^{AND} \cup U^{OR}$*, where U^{AND} and U^{OR} are the subsets of stories having dependency type AND and OR, respectively.*

The sprint planning problem consists in determining an optimal assignment of the x_{ij}'s, i.e., in finding which stories compose each sprint in an optimal plan. Its linear programming formulation is as follows:

$$z = Max \sum_{k=1}^{m} \sum_{i=1}^{k} \sum_{j=1}^{n} u_j (r_j^{cr} x_{ij} + a_j \frac{y_{ij}}{|Y_j|}) \qquad (1)$$

$$s.t. \sum_{j=1}^{n} p_j r_j^{un} x_{ij} \leq p_i^{max} \qquad \forall i \in S \qquad (2)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in U \qquad (3)$$

$$\sum_{k=1}^{i} \sum_{z \in D_j} x_{kz} \geq x_{ij} \qquad \forall i \in S, j \in U^{OR} \qquad (4)$$

$$\sum_{k=1}^{i} \sum_{z \in D_j} x_{kz} \geq x_{ij} |D_j| \qquad \forall i \in S, j \in U^{AND} \qquad (5)$$

$$y_{ij} \leq \sum_{k \in Y_j} x_{ik} \qquad \forall i \in S, j \in U \qquad (6)$$

$$y_{ij} \leq |Y_j| x_{ij} \qquad \forall i \in S, j \in U \qquad (7)$$

The explanation of the elements of this formulation is as follows:

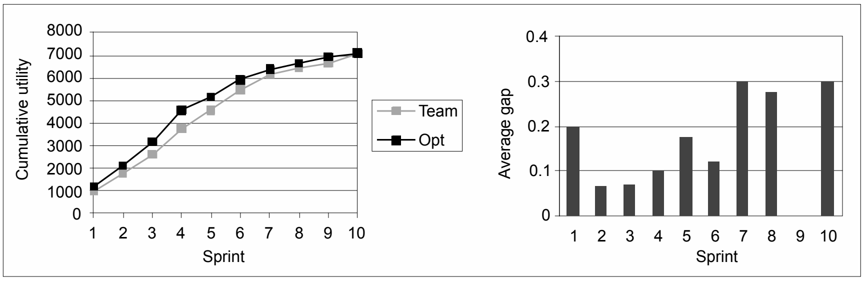
(1) The objective function z states that the optimal plan maximizes the cumulative utility function. The criticality risk r_j^{cr} increases the utility u_j of a critical story j, thus encouraging an early placement of critical stories. Affinity is managed through term $a_j \frac{y_{ij}}{|Y_j|}$, that increases the utility of a story j proportionally to the fraction of similar stories included in sprint i.
(2) These inequalities ensure that the sum of the story points of the stories included in each sprint i does not exceed the sprint capacity p_i^{max}. The story points p_j of story j are increased according to the uncertainty risk r_j^{un} of that story, so as to fairly distribute uncertainty risk among the sprints.
(3) This constraint imposes that each story is included in exactly one sprint.
(4) These inequalities handle OR dependencies by stating that at least one story in D_j is placed before each story j.
(5) These inequalities handle AND dependencies by stating that all stories in D_j are placed before each story j.
(6) These inequalities manage affinity by bounding the number of stories similar to j in sprint i. Using an inequality is necessary to accommodate the fact that, if sprint i includes stories similar to j but j is not part of i, it is $y_{ij} = 0$ (see constraint 7).
(7) These inequalities state that y_{ij} is zero if story j is not part of sprint i, otherwise it cannot be greater than the number of stories similar to it.

CPLEX solves this optimization problem using a branch-and-cut approach [6], that is, a method of combinatorial optimization for solving integer linear programming problems (i.e., linear programming problems where some or all the unknowns are restricted to integer values —the x_{ij}'s and y_{ij}'s in our case). The method is an hybrid of branch-and-bound and cutting plane methods that dramatically improves the performance of classic branch-and-bound methods by incorporating *cutting planes*, that is, inequalities that improve the linear programming relaxation of integer linear programming problems.

5 Model Validation

5.1 Effectiveness Tests

To verify the effectiveness of our model we applied it to a real DW project in the area of pay-tvs, carried out by an Italian system integrator who has successfully been adopting agile methods for five years. The subproject we describe here had an overall duration of 8 months; it included 44 user stories —mostly related to the development of reports, complex ETL units, and forms for manual input of data— and consisted of 10 sprints with an average duration of 17 days. 52 dependencies and just one affinity were involved. The project team included 4 members, but in a few cases one additional programmer was added to support the team.



(a) (b)

Fig. 2. Comparison of cumulative utilities (a) and difference in sprint composition (b) for the optimal and the team plans

The goal of the test presented here is to compare the sprint plan manually defined by the project team with the one generated by our model, using a development speed of 2.43 story points per day. Figure 2.a compares the cumulative utilities of the optimal plan (Opt) and of the plan defined by the team (Team). The curve of the optimal plan is always higher mainly due to a better optimization of sprint composition, but also to a better handling of risk. Indeed, in the team's plan some critical stories with low utility (essentially related to infrastructural needs) were advanced too much.

To better understand how the two plans differ in terms of sprint composition, we measured their difference as the average of the gaps of all user stories:

Definition 2 (User Story Gap). *Let j be a story. Let i^{team} and i^{opt} be the sprints j belongs to in the team plan and in the optimal plan, respectively. The* gap *of story j is*

$$gap(j) = \frac{1}{N-1}|i^{team} - i^{opt}|$$

where N is the maximum number of sprints in the two plans.

The user story gap ranges from 0 to 1, where 0 means that the story belongs to the same sprint in both plans. As shown in Figure 2.b, the average gap is always lower then 0.3, denoting a good correspondence between the two plans. The main difference arises in sprints 1, 7, 8, and 10. In particular, in sprint 1, the team plan aimed at anticipating critical stories, thus exceeding the sprint capacity. The strong difference in the composition of the first sprint necessarily affected the subsequent sprints. Noticeably, both plans made good use of affinity relationships.

In order to have a further evaluation of the optimal plan, we discussed it with the team chief after the project end. Here are the main outcomes:

– The team spent a couple of days in defining their plan, while the optimal plan was generated in a few seconds.
– The team was used to collecting user story estimates using standard forms, but the level of detail required by our framework is slightly higher. This was

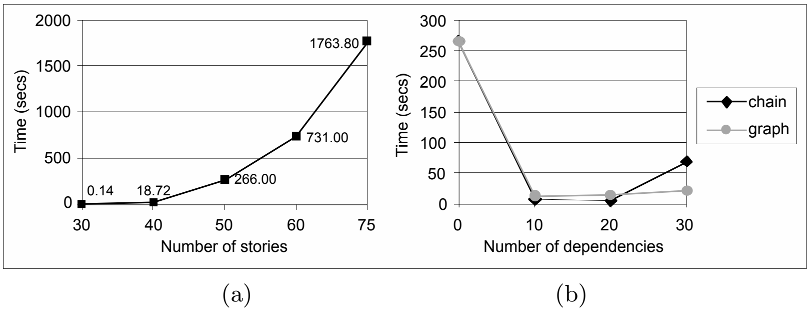


Fig. 3. Time for computing the optimal plan for projects (a) with an increasing number of stories and no dependencies, and (b) with an increasing number of dependencies and 50 stories

perceived has a positive aspect since it leads to more refined estimates, thus producing a better plan.

- The team chief recognized that his plan failed in properly distributing risks, which led to some delay in the first sprint.
- The optimal plan was judged to be feasible and realistic, showing that the elements considered in our model provide a good distribution of user stories.
- Most of the differences in sprint compositions were evaluated as improvements over the team plan. In particular, the team plan did not take into account the side effects of postponing some stories, thus causing the stories depending on them to be delayed too much.

5.2 Efficiency Tests

These tests were carried out on an Intel Core 2 Duo platform with 3 Gb of RAM, running at 3 GHz under Windows XP professional. To test the model behavior on a broad benchmark we generated a set of 58 synthetic projects; utility and story points of the user stories were randomized in the intervals [10,100] and [1,10], respectively. The maximum sprint duration was set to 15 days, while the development speed was set to 3 story points per day (i.e., sprint capacity was 45 story points). All planning problems were solved using CPLEX; performances were measured in seconds.

First of all we evaluate performances in function of the total number of user stories on projects that do not include dependencies. Figure 3.a reports the average time needed to compute the exact solution. As expected for a multi-knapsack problem, the computation time grows non-linearly, reflecting an exponential increase in the search space.

The presence of dependencies makes planning harder for the project team. To study their impact on our model, two types of dependencies were added to our benchmark projects: (1) *chain* dependencies, where each story depends on at most another story; and (2) *graph* dependencies, where a story can depend on several stories. In both cases dependencies were obviously acyclic. Figure 3.b shows how the computation time changes in function of the number of dependencies. This figure suggests that a small number of dependencies tends to reduce the computation time because dependencies allow a set of unfeasible plans to be pruned, thus reducing the search space. However, when the number of dependencies is high, the computation time increases again because finding a feasible plan becomes harder for the solver. Noticeably, we observed that both chain and graph dependencies show similar trends.

Though the time to obtain an exact solution for very complex problems (more than 100 stories) can be too high, the time to obtain a good feasible solution is always limited. CPLEX can be configured so that it first looks for a feasible solution, then it tries to improve it until the exact one is found; at each step it returns the utility of the best solution found so far (i.e., an upper bound to the utility of the optimal solution) and a lower bound to the utility of the optimal solution. We measure the suboptimality at each step (i.e., how the current solution is far from the optimal one) as the ratio between the lower and the upper bounds. Remarkably, a solution that is less than 1% worse than the optimal one is always produced within 5 seconds.

6 Conclusions and Future Work

In this paper we formalized the sprint planning problem for agile DW design and we proposed a multi-knapsack model to solve it. We tested our model on a set of (both synthetic and real) projects. We found that, for medium-sized problems, an exact solution is determined in a time that is fully compatible with the development process (i.e., from some seconds to a few minutes), while for large problems a heuristic solution that is just a few percentage points far from the exact one can be returned in a couple of seconds. As to effectiveness, the team chief judged the optimal plan to be feasible and realistic, and most of the differences in sprint composition were evaluated as improvements over the team plan. Currently, the optimal plan is delivered to the team in tabular form; however, to present the plan in a more effective way, our optimization module could be coupled with existing softwares for agile project management.

We are currently working on extending our model to better support the planning activity. First of all we will accommodate iterative planning, i.e., given a first solution the team will manually adjust it by pinning some user stories to some sprints, and then run the optimizer again. This requires an extension of our model to deal with further types of constraints while preserving the overall structure of the resulting plan. Further improvements that will make the model best fit for real cases are: (1) allowing different development speeds for different sprints due to a variable team composition; (2) modeling different team capabilities (e.g., design, implement, test) so that, in each sprint, the team will be able to deliver a different number of story points for each capability.

References

1. Aalto University, SoberIT: Agilefant (2011), http://www.agilefant.org/
2. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New directions on agile methods: A comparative analysis. In: Proc. ICSE, pp. 244–254 (2003)
3. Beck, K., et al.: Manifesto for agile software development (2001), http://agilemanifesto.org/
4. Boehm, B.W.: A spiral model of software development and enhancement. IEEE Computer 21(5), 61–72 (1988)
5. Brucker, P., Drexl, A., Möhring, R.H., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112(1), 3–41 (1999)
6. Caprara, A., Fischetti, M.: Branch-and-cut algorithms. In: Dell'Amico, M., Maffioli, F. (eds.) Annotated Bibliographies in Combinatorial Optimization. Wiley Interscience Series in Discrete Mathematics (1997)
7. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional (2004)
8. Collabnet: ScrumWorks (2011), http://www.danube.com/
9. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Information & Software Technology 50(9-10), 833–859 (2008)
10. Fréville, A.: The multidimensional 0-1 knapsack problem: An overview. European Journal of Operational Research 155(1), 1–21 (2004)
11. Golfarelli, M., Rizzi, S., Turricchia, E.: Modern Software Engineering Methodologies Meet Data Warehouse Design: 4WD. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 66–79. Springer, Heidelberg (2011)
12. Herroelen, W., Demeulemeester, E., Reyck, B.D.: A classification scheme for project scheduling problems. Tech. rep., Katholieke Universiteit Leuven (1997)
13. Hughes, R.: Agile Data Warehousing: Delivering world-class business intelligence systems using Scrum and XP. IUniverse (2008)
14. IBM: IBM ILOG CPLEX optimizer (2011), http://www-01.ibm.com/
15. Kolisch, R., Padman, R.: An integrated survey of deterministic project scheduling. Omega 29(3), 249–272 (2001)
16. Kruchten, P.: The 4+1 view model of architecture. IEEE Software 12(6), 42–50 (1995)
17. Martello, S., Toth, P.: Knapsack Problems: Algorithm and Computer Implementation. John Wiley and Sons Ltd. (1990)
18. Martin, J.: Rapid application development. MacMillan (1991)
19. Nichols, A.: Agile planning, estimation and tracking (2009), http://www.slideshare.net/andrewnichols/agile-planning-estimation-and-tracking
20. Pomberger, G., Bischofberger, W.R., Kolb, D., Pree, W., Schlemm, H.: Prototyping-oriented software development — concepts and tools. Structured Programming 12(1), 43–60 (1991)
21. Royce, W.W.: Managing the development of large software systems: Concepts and techniques. In: Proc. ICSE, Monterey, California, USA, pp. 328–339 (1987)
22. Schwaber, K.: SCRUM development process. In: Proc. OOPSLA (1995)
23. ThoughtWorks Studios: Mingle: Agile project management (2011), http://www.thoughtworks-studios.com/

Using OCL for Automatically Producing Multidimensional Models and ETL Processes

Faten Atigui, Franck Ravat, Olivier Teste, and Gilles Zurfluh

IRIT (UMR 5505) Toulouse University
118 route de Narbonne F-31062 Toulouse
{atigui,ravat,teste,zurfluh}@irit.fr

Abstract. During the last few years, several frameworks have dealt with Data Warehousing (DW) design issues. Most of these frameworks provide partial answers that focus either on multidimensional (MD) modelling or on Extraction-Transformation-Loading (ETL) modelling. Yet, neither the study of unifying both modelling issues nor their automation have been considered thoroughly. To overcome these limits, we suggest a generic unified method that automatically integrates DW and ETL design. The framework is handled within the Model Driven Architecture (MDA). In this paper we present a unified conceptual model that describes both the DW and its ETL process using the constellation model and the Object Constraint Language (OCL). Morevoer, we give a logical model for the ETL workflow and a set of Query/View/Transformation(QVT) mapping rules from the conceptual level to the logical level and then to the physical one. At the end, we describe the implemented prototype architecture.

Keywords: Data Warehouse, Multidimensional Modelling, Extraction-Transformation-Loading, Object Constraint Language, Model Driven Architecture.

1 Introduction

When building a Data Warehouse (DW), the designer deals with two major issues. The first issue addresses the DW design and the second, ETL process design. There are three basic steps in the DW modelling task. Conceptual modelling provides a common representation of decision-makers requirements and data incoming from various sources. The conceptual model is then mapped into a logical one in the second step. In the last step, the multidimensional (MD) structures are implemented within a target platform. The next task deals with the ETL processes description and implementation.

Current frameworks provide partial solutions that focus either on the MD structure or on the ETL processes, yet both could benefit from each other. In fact, the whole process (that creates MD structures and loads data in the warehouse) requires the use of different models. Data integration problems must be considered in order to select the appropriate models. Besides, most of the existing approaches do not provide means to automatically generate or document all of these aspects.

The Model Driven Architecture (MDA)[1] is known to be a framework that manages complexity, significantly reduces development costs, improves the software quality and accomplishes high levels of re-use [7]. In previous work [1] we have presented a general framework for MD and ETL design. The framework is a model driven approach that addresses both DW and ETL process modelling. The work in [1] details the structural features and provides formal rules to generate the MD model. This paper focuses on the behavioral features i.e ETL processes. The main contribution of this paper focuses on:

1. A unified conceptual model for MD structures and ETL that reduces design costs and efforts.
2. Extraction formulas formalization using an OCL extension that allows the early detection of inconsistencies and limits their impact.
3. Formal models and rules to accomplish automatic translation of ETL conceptual operations.

The rest of this paper is organized as follows: in section 2 we present related work. In section 3 we introduce our approach. Afterwards, we detail the conceptual model in section 4. In section 5 we present the logical and the physical models as well as the mapping rules. Section 6 shows our prototype.

2 Related Work

Several researches studied DW design problems. In the literature, this issue has been tackled from two complementary but different viewpoints [14]. The first one deals with the MD modelling and attempts to describe the DWs MD structures [15]. As for the second one, it represents processes that load and update data in the warehouse [20]. Regarding the MD design, the existing approaches can be classified into three categories [14]. Requirement-driven approaches [12] provide MD schemas based on a detailed analysis of decision-makers' needs. Data-driven approaches [5] start with the data sources analysis in order to identify the structure of the MD schema and then select relevant data for decision-making. Finally, hybrid approaches [9] consider both decision-makers' requirements and data availability within operational sources. Regarding ETL processes, academic researchers use either specific models or existing standards such as UML. [17] introduces specific graphical notations for defining ETL conceptual mechanism and presents a set of rules that map the conceptual model to the logical one. [18] suggests to ease the selection of relevant data sources that are transformed and loaded in the warehouse using semantic web technologies. Some authors extend UML notations to describe ETL workflows. [8] extends UML using a mapping diagram to represent the transformation rules between sources and MD attributes. [10] uses the UML activity diagrams to design the ETL process. Furthermore, there are tools for running ETL workflows [2], e.g. Oracle Warehouse Builder

[1] http://www.omg.org/cgi-bin/doc?omg/03-06-01

and Microsoft Integration Services. However, these tools use specific notations and languages, thus decreasing the integration and the interoperability levels of the system.

Compared to the existing approaches, our approach has the advantages of combining the description of MD data structures and ETL operations within a unified model. By using a unified model we avoid redundant steps, such as linking data sources. This model also avoids problems of inconsistency, integration and interoperability encountered when using separate models and/or methods. Besides, the facts and the dimensions include both structures and operations and therefore ready to be used in other MD models. Moreover, our approach reuses and adapts existing models and languages as the Object Constraint Language[2] and the constellation model [4], [13].

3 Overview of Our Approach

Figure 1 depicts our MDA-based framework. The designer builds a unified conceptual model (PIM: Platform Independent Model) that describes the MD structures as well as the related ETL processes. The automatic transformations (M2M: Model-To-Model and M2T: Model-To-Text) translate it into successive models (PSM: Platform Specific Model) in order to get the code tailor-made to the chosen platform. The conceptual model (PIM1) describes the MD structures that considers the decision-makers' needs and the existing data sources. The PIM1 is mapped into several logical models (PIM2) (CWM::Relational, CWM::XML, etc.) depending on a chosen deployment platform (Oracle, Mondrian, etc.). The framework provides the appropriate script that creates the MD structure and the ETL workflow. The data sources modelling levels are generated by Reverse Engineering (RE) from physical models.

The framework addresses the process of designing a DW that includes modelling the MD structures and the associated ETL processes. This paper defines a platform-independent conceptual model based on MD model enhanced with OCL expressions. We present a formal conceptual description of ETL operations using OCL. Then we introduce the formal rules that generates the ETL logical and physical models. Finally, we describe the code generation phase.

4 Unified PIM: ETL Formal Modelling

Even though there is no standard method for DW conceptual modelling [16], the constellation model is widely used to represent MD databases [14]. However, this model represents the DW structural features only and lacks a mechanism that describes the behavioral aspects. In order to overcome this limit, we extend the Object Constraint Language and we define OCL expressions within the constellation model.

[2] http://www.omg.org/spec/OCL/2.2/PDF

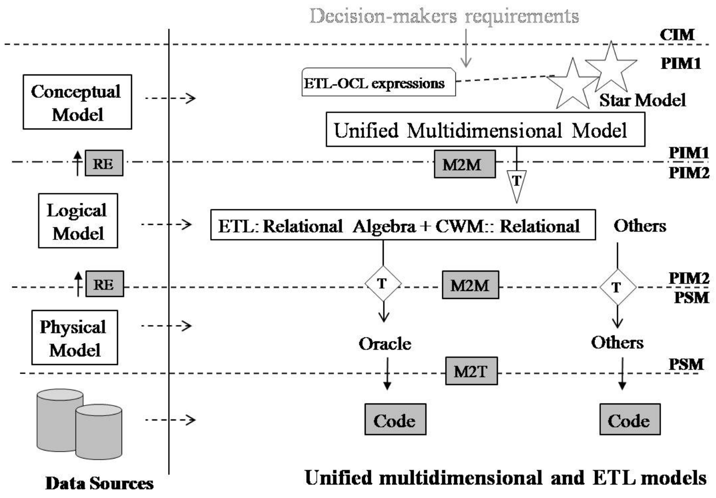


Fig. 1. Overview of our approach

4.1 Why ETL Formal Language?

In an ETL workflow, the data sources are transformed and loaded into the DW. ETL conceptual design specifies the transformation relation between the DWs target attributes and the source attributes. In order to describe the multidimensional structure and the associated ETL processes, it is important to integrate these operations into the multidimensional model. We need to express the relationships between attributes that belong to different models. We require a language that describes the ETL operations in a platform independent way and reports to the conceptual abstraction level. This language needs to be formal in order to automatically generate the final code. It must be able to express arithmetic and string operations as well as selection and aggregation operations.

The Object Constraint Language partially meets these needs. It expresses constraints and queries within platform independent model. OCL is a formal language that could be automatically translated to the logical and physical levels. The major advantage of the OCL expressions is the ability to detect some of the inconsistency that could exists between sources and target attributes [21]. This language is intended to refine the UML class diagrams. It has been extended to express constraints and queries on various models such as multidimensional models [11], [3].

4.2 ETL-OCL Syntax

In multidimensional modelling, each attribute is derived from one or more source attributes. It is possible to use OCL to express relations between attributes of different models. For this, it is crucial to express the relationship between attributes in terms of relations between concepts (classes, facts, dimensions, etc.). The two models (source and target) are then linked in order to navigate within source concepts and define ETL-OCL expressions. Each target concept (fact or dimension) is connected to one origin concept (class or association) in each source. An OCL expression is defined within a context (a class, an attribute or operation in a class diagram). OCL can express invariants, pre- and post-conditions. An ETL-OCL expression is a specific OCL expression defined within the context of a multidimensional attribute and expresses a derivation relationship between this attribute and one or more sources attributes.

To transform the data sources, ETL-OCL expressions use the operations provided by the OCL library such as concat(), size(), substring(), toInteger(), toUpperCase(), etc. for string attributes. Moreover, the OCL library provides mathematical functions applied to transform numeric attributes (min(), max(), product(), sum(), etc.). Except for data aggregation, we believe that OCL is rich enough to provide a conceptual description of the most common ETL operations. Yet, OCL lacks ways to express the data aggregation often used to transform the source data. We extend OCL with an aggregation function as follows:

$AGG(A_{si} \rightarrow AF; A_{sj}[; P_{asj}])$ where :

- A_{si}: an aggregated source attribute,
- AF: aggregation function predefined in the OCL library (sum, count, min, max, avg (sum / size)), etc.
- A_{sj}: a set of source attributes used for grouping values,
- P_{asj}: optional selection predicate applied on the aggregation.

The table 1 presents the main ETL conceptual operations and their definition using the ETL-OCL language [17], [19].

Example 1. The case study describes a company that wishes to analyze the sales of an internationally sold product. Figure 2 shows the MD model (top of the figure) used for analyzing the sales amount and quantities according to products, customers and dates. At the bottom, the class diagram describes the data source (SR) schema. Each target concept is connected to a source concept. These links are shown as dotted lines. The association roles are used to navigate within the data sources. Figure 3 presents an example of the ETL-OCL extracting formulas. Due to lack of space, we expose the extracting formulas relative to the "Customer" dimension and the "Sales" fact only.

5 Automatic Generation of Logical and Physical Models

In this section, we present the mapping rules from ETL-OCL operations to their respective logical operations in relational algebra. Then, we present the transformation rules from logical to physical operations (SQL). The mapping rules are formalized using the QVT language[3]. QVT is the OMG standard language for

[3] http://www.omg.org/spec/QVT/1.1/PDF/

Table 1. Transformation operations

Transformation operation	**Description**	**ETL-OCL**
Identity	A simple mapping relation between a target and source attribute.	Equality operator.
Conversion (classes, associations)	Convert the source attributes value, format and type.	OCL library Operations (mathematical operations, string operations).
Selection (filter)	Select attributes values that satisfy a given criteria.	Select and Reject operations.
Aggregation	Aggregates source attributes according to others sources attributes.	AGG operation.
Incorrect	Detects incorrect data (type or format).	Exception rose when the OCL expression is assessed to invalid.
Join (classes or association)	Joins two data sources related to each other with common attributes.	Navigation through association and classes to access attributes.
Merge (models)	Connects source and target models.	Links established between target model (fact and dimensions) and source classes diagram.

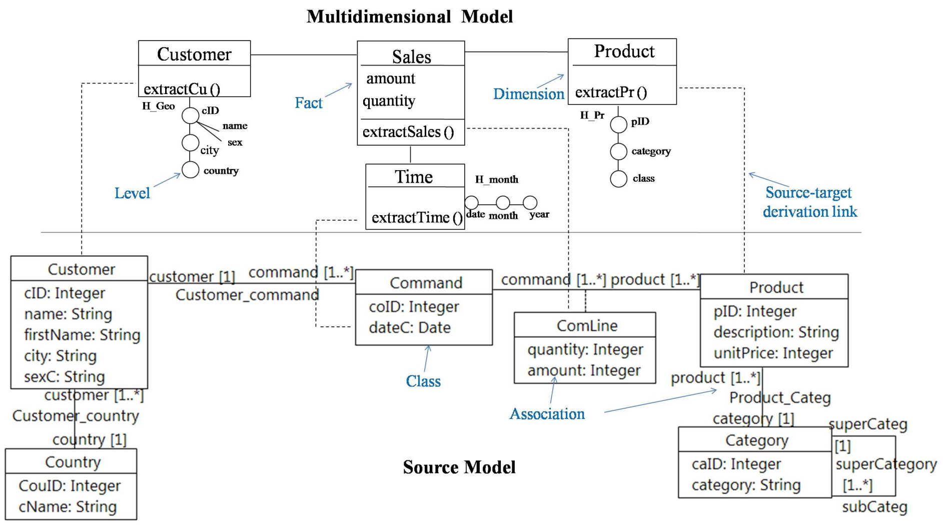


Fig. 2. Multidimensional PIM (top side) and the source class diagram (bottom side)

```
-- These ETL-OCL expressions specify the type and the extraction formulas of the
‘‘Customer" dimension attributes (the customer’s code, name, sex, city and country).
context Customer::cID: Integer
derive: SR.customer.cID
context Customer::name: String
derive: SR.customer.name.concat(SR.customer.firstName)
context Customer::sex: String
derive: IF SR.customer.sexC=‘male’ THEN ‘M’ ELSE ‘F’ ENDIF
context Customer::city: String
derive: SR.customer.city
context Customer::country: String
derive: SR.customer.country.cName


-- The sales quantity is the daily quantity sold per customer and per product.
context Sales::quantity: Real
derive: AGG(SR.comLine.quantity->sum();
SR.ComLine.product.pID, SR.comLine.command.customer.cID,SR.comLine.command.dateC)

-- The sales amount is the sum of the quantity multiplied by the product’s
unit price for each customer, product and date. context Sales::amount: Real
derive: AGG(SR.comLine.quantity*SR.ComLine.product.unitPrice->sum();
SR.ComLine.product.pID,SR.comLine.command.customer.cID,SR.comLine.command.dateC)
```

Fig. 3. Example of ETL-OCL extracting formulas

model-to-model (M2M) transformations. It is a declarative language that offers both graphical and textual syntax. Furthermore, QVT allows multi-directional transformations as well as merging models (two or more models are mapped into one model or more). A QVT transformation between two candidate models is specified by a set of relations. Relations are defined by two or more domains that identify a candidate model and a set of corresponding elements to be matched as well as a pair of “When” (pre-conditions) and “Where” (post-conditions) predicates. In the next subsections, we detail the ETL-OCL mapping rules to the relational algebra that carry out the transformation rules presented in [1]. Then we present the QVT mapping relation to the Oracle 11g platform specific models. Finally we present the Model-To-Text (M2T) transformations rules that generate the final SQL script.

5.1 Logical Model

Logical models are automatically generated from conceptual models by applying a set of rules. In order to cover as much as possible of warehousing applications, the main framework (cf. figure1) provides a set of logical models. The designer can choose the most suitable one to his application such as the normalized or the denormalized ROLAP (Relational On Line Analytical Processing) or the XML models, etc. To illustrate our framework, we define transformation rules to denormalized ROLAP often used to represent MD models [6]. In previous work [1], we have developed the structural translation rules from the MD model

Table 2. Mapping ETL-OCL operations to relational algebra operations

Conceptual operations	**Logical operations**
Identity	Projection (Π [attribute] R).
Join (classes)	Join relations related to each other with attributes ($R_1 \underset{R_1.a=R_2.a}{\bowtie} R_2$)
Select (criteria)	Selection (criteria): (σ[criteria] R)
Conversion	Conversion or aggregation functions.
Aggregation	Aggregation functions (AGG(R; group; attributes; predicates)).
Incorrect	–
Merge (models)	Creates a link between models.
–	Surrogate: generates unique surrogate keys.

to the denormalized ROLAP. This section carries out those rules and presents the behavioral mapping rules from conceptual to logical ETL models. In table 2 we examine each of the ETL-OCL operations and we identify their respective logical (relational algebra) operations.

In this table, the "Incorrect" operation belongs to the conceptual level only. It raises an exception that disables the translation to the logical level. Once the exception is treated (data type and format are evaluated to true), the translation to the logical level is enabled . On the other side, the "Surrogate" operation is specific to the relational logical level for generating synthetic unique keys.

Example 2. The ETL-OCL expressions of the conceptual model as presented in example 1 are mapped into the following relational algebra expressions. The source relations (SR) define the logical model respective to the class diagram shown in figure 2.

DW.Customer = Π[cu.cID, name||firstName AS name, city, cName AS country, sexC AS sex] SR.Customer $\underset{\texttt{couID}}{\bowtie}$ SR.Country

DW.Sales = Π[dateC AS date, pID, cID, quantity, amount] ((Sum(SR.Command $\underset{\texttt{coID}}{\bowtie}$ SR.ComLine.pID, SR.Command.cID, SR.Command.dateC; SR.ComLine.quantity AS quantity;)) $\underset{\texttt{coID}}{\bowtie}$ (Sum(SR.Command $\underset{\texttt{coID}}{\bowtie}$ SR.ComLine $\underset{\texttt{pID}}{\bowtie}$ SR.Product; SR.ComLine.pID, SR.Command.cID, SR.Command.dateC; SR.ComLine.quantity * SR.Product.unitPrice As amount;)))

Fig. 4. Example of Relational Algebra extracting formulas

5.2 Physical Model

Physical models depend on a specific platform; e.g. here we detail the transformation rules that generate the Oracle 11g materialized views. The use of materialized views is advantageous since the computation, the load and the refresh tasks are performed automatically by the database management system.

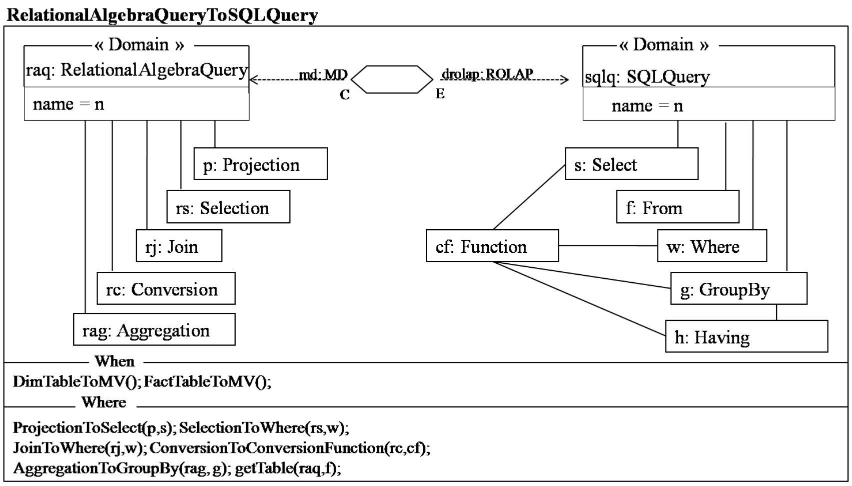

RelationalAlgebraQueryToSQLQuery
« Domain »
raq: RelationalAlgebraQuery
name = n
md: MD
C
drolap: ROLAP
E
« Domain »
sqlq: SQLQuery
name = n
p: Projection
rs: Selection
rj: Join
rc: Conversion
rag: Aggregation
s: Select
f: From
cf: Function
w: Where
g: GroupBy
h: Having
When
DimTableToMV(); FactTableToMV();
Where
ProjectionToSelect(p,s); SelectionToWhere(rs,w);
JoinToWhere(rj,w); ConversionToConversionFunction(rc,cf);
AggregationToGroupBy(rag, g); getTable(raq,f);

Fig. 5. Relational Algebra to SQL expression rule

The QVT structural rules that maps fact and dimension tables into materialized views are detailed in [1]. In the following, we present the main and the relational algebra to the (Oracle 11g) SQL expression rules.

RelationalAlgebraToOracleSQL Relation. This relation is shown in figure 5 and maps each relational algebra expression into an SQL query. The "When" clause specifies that the appropriate ROLAP table must be already mapped into a materialized view. The "Where" clause specifies that relations, attributes and operators are mapped into their equivalent SQL relations, attributes and operators.

5.3 Code Generation: MOF M2T Transformation Rules

The last phase of our framework describes the code generation process. The code is obtained from the physical model using the MOF (Meta-Object Facility) Model-to-text which is an OMG standard[4]. For instance, starting from a single relational data source, the "Customer" and the "Country" tables' tuples are loaded in the "Customer" materialized view. The figure 6 shows the SQL script that creates the "Customer" and the "Sales" materialized views.

6 Implementation

Figure 7 shows our prototype architecture. This prototype allows loading a multidimensional model and ETL-OCL expressions. Then the prototype executes a set of model to model (QVT) and model to text (MOF M2T) transformations

[4] `http://www.omg.org/spec/MOFM2T/1.0/PDF`

```
CREATE MATERIALIZED VIEW Customer
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
AS SELECT cID, concat (name,firstName) AS name, city, cName AS country
FROM SR.Customer, SR.Country
WHERE SR.Customer.couID = SR.Country.couID ;


CREATE MATERIALIZED VIEW Sales
BUILD IMMEDIATE
REFRESH COMPLETE ON DEMAND
AS SELECT SR.Customer.cID, pID, dateC AS date, SUM (quantity) AS quantity,
SUM(amount*unitPrice) AS amount
FROM SR.Command, SR.Customer, SR.ComLine
WHERE SR.Command.coID = ComLine.coID AND Command.cID = Customer.cID
GROUP BY pID, SR.Customer.cID, dateC ;
```

Fig. 6. Example of the SQL script

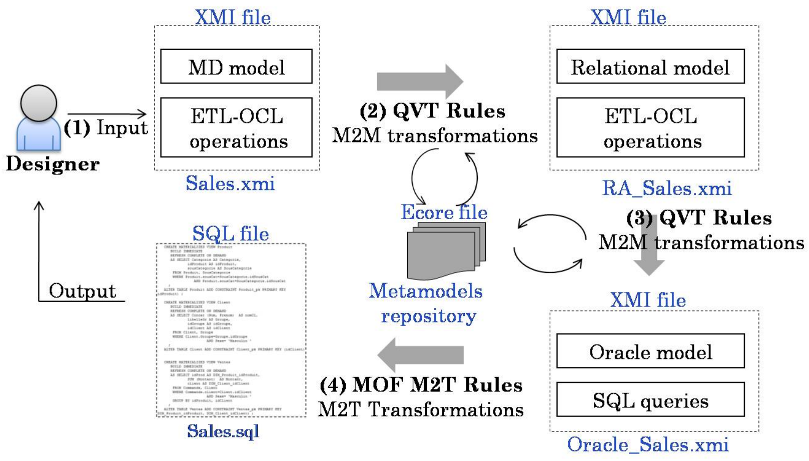


Fig. 7. The prototype architecture

rules in order to generate the SQL script that creates and loads data in the warehouse. These rules use a set of metamodels stored as "Ecore"[5] files. The prototype is based on three main steps:

– Step 1: Creating the unified metamodels ("Ecore" files):
  1. The conceptual metamodel describes the multidimensional structure and ETL operations.
  2. The logical metamodel describes relational structures and algebraic expressions.
  3. The physical metamodel describes Oracle materialized views and their SQL queries.

[5] http://www.eclipse.org/modeling/emf/?project=emf

- Step 2: Creating the QVT model to model transformation rules:
  1. The transformations rules from the conceptual model to the logical model.
  2. The merging rules of both conceptual and logical models into the physical model.
- Step 3: Creating the MOF M2T model to text transformation rules.

The transformations and merging rules have been implemented using the Medini-QVT[6] Eclipse plugin. This plugin uses the QVT textual syntax. Then, the model to text transformation rules are implemented within the MOFScript[7] Eclipse plugin. This plugin implements transformation rules using the MOF M2T language.

As shown in figure 7, the designer creates an instance of the multidimensional model and their associated ETL-OCL operations as an XMI[8] file. Then, this conceptual model (2) is automatically transformed into a logical relational model. Afterwards, both conceptual and logical models (3) are combined to get an Oracle physical model. Finally, the latter (4) is transformed into SQL commands by applying a set of MOF M2T rules.

7 Conclusion

In this paper we have presented a unified and automatic model driven approach for multidimensional and ETL design with the support of MDA.

As for conceptual design, the method provides a unified description of the multidimensional structure and the associated ETL processes. Thus, the unified model allows reducing design costs and efforts. The conceptual model presents a constellation and a set of ETL-OCL expressions that describe the multidimensional attributes extracting formulas. These extraction formulas are formalized using an OCL extension allowing the early detection and management of inconsistencies as well as limiting their impact in the lower levels. We have detailed the mapping rules between the ETL-OCL and the relational algebra. Then, we have presented the transformation rules from the logical model to the Oracle 11g materialized views and dimensions. The model-to-model transformations rules are formalized using QVT. While the model-to-text rules are formalized using the MOF model-to-text transformation language. Finally, we have presented our prototype.

In future work, we plan to focus on the requirements formalization phase (CIM) and on automating the transition between the CIM and the conceptual PIMs. This can be done by defining a set of QVT rules. Additionally, we intend to develop our method by considering different implementation platforms. Moreover, we plan to treat multiple data sources. We also intend to improve the prototype by adding a graphical interface.

[6] http://projects.ikv.de/qvt
[7] http://www.eclipse.org/gmt/mofscript/
[8] XMI: XML Metadata Interchange

References

1. Atigui, F., Ravat, F., Tournier, R., Zurfluh, G.: A unified model driven methodology for data warehouses and ETL design. In: ICEIS (1), pp. 247–252 (2011)
2. Barateiro, J., Galhardas, H.: A survey of data quality tools. Datenbank-Spektrum 14, 15–21 (2005)
3. Cabot, J., Mazón, J.-N., Pardillo, J., Trujillo, J.: Specifying Aggregation Functions in Multidimensional Models with OCL. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 419–432. Springer, Heidelberg (2010)
4. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: A conceptual model for data warehouses. Int. J. Cooperative Inf. Syst. 7(2-3), 215–247 (1998)
5. Golfarelli, M., Rizzi, S.: Methodological framework for data warehouse design. In: DOLAP, pp. 3–9 (1998)
6. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley (1996)
7. Kleppe, A.G., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)
8. Luján-Mora, S., Vassiliadis, P., Trujillo, J.: Data Mapping Diagrams for Data Warehouse Design with UML. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 191–204. Springer, Heidelberg (2004)
9. Mazón, J.-N., Trujillo, J.: A hybrid model driven development framework for the multidimensional modeling of data warehouses! SIGMOD Record 38(2), 12–17 (2009)
10. Muñoz, L., Mazón, J.-N., Trujillo, J.: Automatic generation of ETL processes from conceptual models. In: DOLAP, pp. 33–40 (2009)
11. Pardillo, J., Mazón, J.-N., Trujillo, J.: Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses. Inf. Sci. 180(5), 584–601 (2010)
12. Prat, N., Akoka, J., Comyn-Wattiau, I.: A UML-based data warehouse design method. Decision Support Systems 42(3), 1449–1473 (2006)
13. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Graphical Querying of Multidimensional Databases. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 298–313. Springer, Heidelberg (2007)
14. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: DOLAP, pp. 3–10 (2006)
15. Romero, O., Abelló, A.: A survey of multidimensional modeling methodologies. IJDWM 5(2), 1–23 (2009)
16. Sen, A., Sinha, A.P.: A comparison of data warehousing methodologies. Commun. ACM 48(3), 79–84 (2005)
17. Simitsis, A.: Mapping conceptual to logical models for ETL processes. In: DOLAP, pp. 67–76 (2005)
18. Simitsis, A., Skoutas, D., Castellanos, M.: Representation of conceptual ETL designs in natural language using semantic web technology. Data Knowl. Eng. 69(1), 96–115 (2010)
19. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 307–320. Springer, Heidelberg (2003)
20. Vassiliadis, P.: A survey of extract-transform-load technology. IJDWM 5(3), 1–27 (2009)
21. Warmer, J., Kleppe, A.: The Object Constraint Language: Getting Your Models Ready for MDA, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)

A Case Study on Model-Driven Data Warehouse Development

Thomas Benker and Carsten Jürck

Department of Information Systems – Systems Engineering
University of Bamberg
{thomas.benker,carsten.juerck}@uni-bamberg.de

Abstract. The development of data warehouses (DW) is still complex and costly. In recent years, a growing number of contributions suggested the application of Model-Driven Architecture (MDA) to DW development. However, most of these approaches are not evaluated in practice. Therefore, we conducted an evaluation of the Multidimensional Model-Driven Architecture (MD²A), based on a case study derived from a real-world DW project. The case study results can be summarized as follows: The suggested full-automatic transformation of conceptual schemas is not feasible under all circumstances. Necessary performance optimizations in the logical level cannot be derived from the conceptual. Additionally, we can conclude that the integration of evolution concepts into an MDA approach for DWs is desirable for two reasons: First, when conceptual schemas are changed and transformed into the logical level, previously made manual optimizations have to be conserved. Second, transformations from the logical level to code have to consider the protection of historicized data.

Keywords: data warehouse, model driven architecture, case study.

1 Introduction

Data warehouse (DW) systems are the cornerstones of corporate information management. They provide access to integrated, quality-assured, subject-oriented, and historical data for decision support (cf. [1]). However, the construction of these systems is still a complex and costly task. In recent years, more and more contributions advocated the use of the Model-Driven Architecture (MDA) for data warehouse engineering (e. g., [2-6]). Each of these approaches presents a framework for a distinct number of DW components and MDA viewpoints including the corresponding meta-models and transformations. However, most of these approaches, to the best of our knowledge, are not evaluated against requirements of real-world projects. For this reason, suggestions for future research based on practical insights are currently missing.

We present a case study based on a real-world project to evaluate the Multidimensional Model-Driven Architecture (MD²A) of Mazón et al. ([6]). We selected this framework for two reasons: First, it is one of the earliest contributions and a good sample for other MDA approaches in the context of data warehouse engineering (e.g., [2-5]). Second, it is the best described approach measured by the number of contributions. The publications cover various data warehouse engineering topics like the unified development process and an MDA framework, including transformations and meta-models as well as extensions to special problems like spatial-data warehousing or secure XML data warehouses (cf. [7-11]).

The fundamental research question in this work is: Does the MD^{2}A approach work and which assumptions have to be made? To answer this research question we followed a three step process: First, we created a case study, based on a two-year project in the public sector, with the goal of building a data warehouse for human resources. Second, we implemented the MD^{2}A framework based on the Eclipse Modeling Framework. Finally, we used the prototype to implement the case study and evaluated the results against the requirements of the project.

The remainder is structured as follows: Section 2 gives a brief introduction to the design process of data warehouse systems in general. Section 3 describes the profile of the Human Capital Management (HCM) Project, which serves as an outline for our case study. It also presents the central requirements identified in the course of the project. Section 4 gives a short introduction to the MD^{2}A approach. The application of this approach in the HCM case study is presented in section 5. Thereby, the focus is set to the conceptual design of multidimensional schemas and the derivation of logical data structures. Section 6 encompasses a discussion of the results. The paper ends with a conclusion and future research proposals (section 7).

2 The Data Warehouse Design Process

In [12] the authors distinguish, as a consensus of different approaches, four phases within a data warehouse design process: the requirements analysis, the conceptual design, the logical design, and in some approaches the physical design. We agree to these steps which are also adopted by the reviewed MD^{2}A approach. Currently, there is a lack of agreement on the particulars that are considered within each phase. Therefore, we explicate our perspective on this process. The first phase is the requirements analysis. Its purpose is the collection and filtering of user requirements. Together with an analysis of the operational data sources, the specification of user requirements delivers the input for the conceptual design phase [13]. In this phase a multidimensional schema is created. It serves as foundation for the reconciliation between designer and user. Therefore, the conceptual design should emphasize business concepts and not on technical issues [12]. The logical schema is derived from the conceptual schema and is specific to a certain platform type (e. g., relational database). It is adjusted for the implementation on a certain target system type, considers platform specific constraints, and is optimized to reach non-functional requirements [12]. An example of a non-functional requirement could be the minimization of query response time. The logical design encompasses the translation of the multidimensional model into tables, and the improvement of the schema design such as view materialization [14] or junk dimensions and rapidly changing dimensions as presented in [15]. We consider all adoptions of the relational data structures to be in logical design, because they are largely independent from a specific data base system. All optimizations, like indices or partitions, which are very likely to be database-specific, are considered in the physical design.

3 The Human Capital Management Project

The CEUS (computer-aided decision support system) project[1] is funded by the Bavarian State Ministry of Sciences, Research and the Arts (StMWFK). The StMWFK planned to build a human resources data warehouse system based on the SAP HCM module for the Universities Augsburg, Bamberg, Erlangen-Nuremberg, and Munich. SAP data sources are integrated as DW via an extraction, transformation, and load (ETL) process. The resulting DW system will be operated and maintained in the CEUS project. The goal of the project is to deliver a specialized data mart for tactic and strategic management of human resources processes. The development of the DW systems took place in parallel with the roll out of the HCM module in the participating organizations. Therefore, we used a hybrid iterative development process for this project, i.e. starting simultaneously with the analyses of SAP HCM data sources, and the collecting of user requirements. During the course of the project several prototypes had to be provided for clarification of requirements and refinement of the multidimensional schema, due to the missing user experience with the operational systems. This project works as a blueprint for the present case study and is representative of most of the development projects implemented in CEUS. We simplified the requirements and corresponding multidimensional schemas for the sake of clarity and due to space constraints.

In the following, two use cases are presented that were captured during user interviews.

Functional requirements (F) for the project can be stated as follows:

F1: A human resources employee needs to analyze the number of persons and full-time equivalents for a given day regarding the working time and the organizational unit of a person.

F2: A human resource employee needs to analyze the number of employees that have part time jobs regarding the organizational unit, job description and gender.

Non-functional requirements (NF) for the project can be stated as follows:

NF1: Reports are categorized in groups describing the desired response time: For example (group 1) reports should be finished within a minute; (group 2) reports should be finished within 3 minutes; (group 3) reports can take more than 3 minutes.

Organizational requirements concerning the project proceeding and used approaches can be stated as following:

O1: The development has to follow a prototyping approach, i.e. first a prototype of a central system part is implemented and then iteratively evaluated and enhanced. This is because of low user experience with the new operational system. It is not expected that valid and complete requirements will be achieved by merely using interviews.

O2: Data warehouse systems are not expected to remain static after their development. Further changes in functional requirements are very likely. These changes must be integrated during the systems production phase, too.

[1] `http://ceushb.de/`

4 Introduction of the MultiDimensional Model Driven Architecture (MD^{2}A)

In [6] the authors present an approach that is motivated by the goal to define a holistic data warehouse development approach. Holistic means that it covers all parts of such a system, e.g., ETL processes, data sources or DW repository. This approach is combined with the Model Driven Architecture to achieve, among others, the following advantages: (i) automatic generation of the data warehouse and, therefore, enhanced productivity of its development; (ii) enhanced portability of the platform independent model (PIM) between different target platforms; (iii) reusability of transformations in different projects; and (iv) concentration on business issues during development. The focus in [6] and, in subsequent publications is on the application of the MDA to multidimensional modeling, called MD²A. After a short introduction of the MDA and the MD²A approach, in the following section we describe the application in context of our case study.

The Model Driven Architecture represents a standard of the Object Management Group (OMG) and can, in short, be characterized as follows ([16]): Its main contribution is to provide an architecture, which allows the separate specification of the functionality of a system and its target platform. The PIM is used for the functional system specification without concerning peculiarities of certain platforms. Model transformations use this PIM together with a platform description to produce a platform specific model (PSM) for the system. The PSM itself can also be used as input of a transformation producing code for the implementation of the system on a certain platform.

The MD^{2}A approach classifies the conceptual model of multidimensional schemas as PIM and presents a UML profile ([17]) as a meta-model for multidimensional modeling. At the platform specific layer the authors pay only attention to relational OLAP systems. For their specification they propose to use the relational package of the Common Warehouse Metamodel (CWM, [18]). The transformations are implemented by using the relational part of the Query/View/Transformation Specification (QVT, [19]).

5 Application of MD^{2}A to the Human Capital Project

This section describes the application of the MD²A to the Human Capital Project in detail. We developed a prototype using the Eclipse Modeling Framework (EMF) with its meta²-model ecore and operational QVT. The UML profile was implemented following the specification in [17]. The relational package of the CWM is an adjusted implementation from the Technical University of Dresden[2]. The transformations of dimensions and associated elements follow the specifications in [6]. Transformations of fact related elements are derived from [17]. The following subsections present the

[2] `http://svn-st.inf.tu-dresden.de/svn/dresdenocl/trunk/ocl20forEclipse/eclipse/tudresden.ocl20.pivot.tools.CWM/resources/model/CWM.ecore`

application of MD²A, starting with conceptual design and ending with the derivation of relational structures.

5.1 Conceptual Data Warehouse Design

Following the data warehouse design process of section 2, we defined our conceptual model using the UML profile, illustrated in Fig. 1. The core concept of the meta-model is the Fact object type, representing a collection of measures. Measures are modeled using the FactAttribute object type. Each fact object is aggregated along multiple dimensions (represented by the Dimension object type) and its Bases. Bases are used to model an acyclic graph representing the various hierarchy levels of a dimension. A detailed description of the meta-model can be obtained from [17].

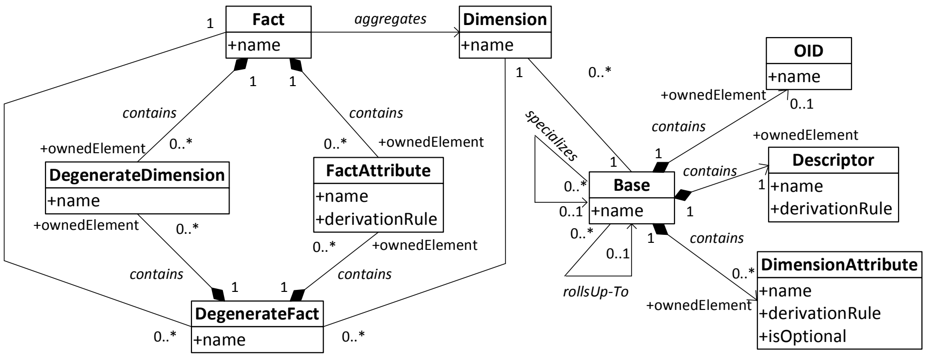
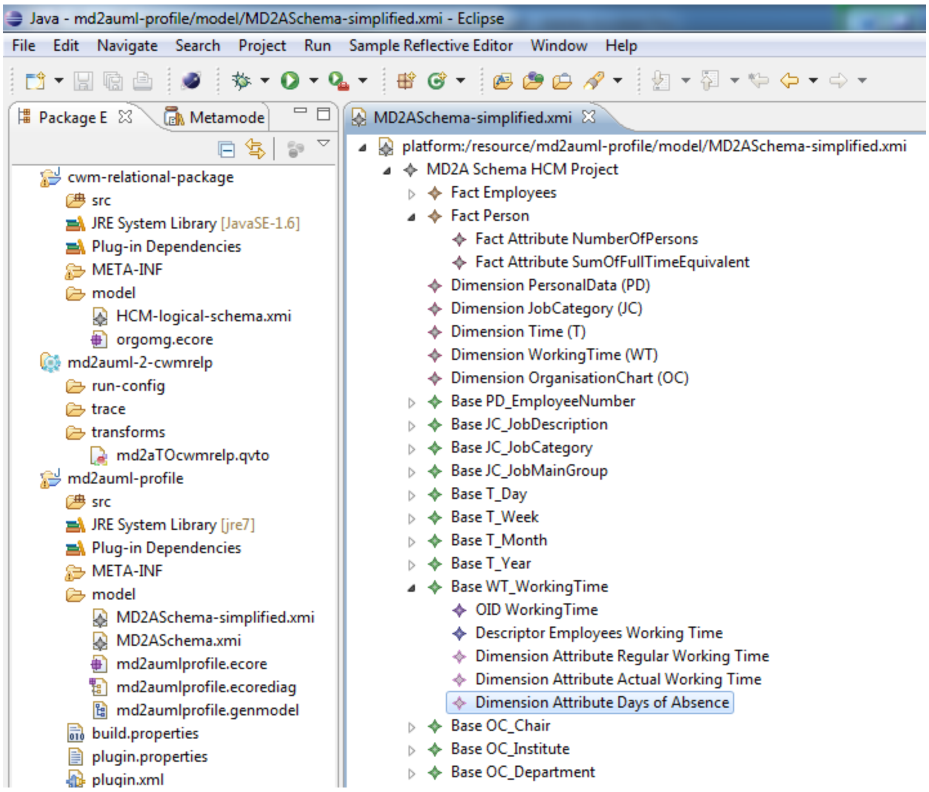


Fig. 1. Meta-Model of the UML Profile for Conceptual Modeling [17]

The conceptual schema of our case has five dimensions and two facts in order to provide the needed analytical capabilities described in section 3 (***F1***, ***F2***). An excerpt of the resulting conceptual schema is shown as tree-based diagram in Fig. 2. We modeled the facts, Employees and Person. While a person represents a unique natural person, an employee represents a natural person with a specific contract type. It has to be considered that one person can have more than one contract and to address this, one person may represent multiple employees one for each contract. Both facts share the measure 'sum of full time equivalent[3]' and a measure counting the number of persons, respectively employees. The fact objects are characterized by five dimensions: 'PersonalData,' 'JobCategory,' 'Time,' 'WorkingTime,' and 'OrganizationalChart.' Each dimension is associated with at least one root base object representing the lowest hierarchy level. This is the base 'WorkingTime' in the case of the dimension 'WorkingTime.' A Base can further be characterized by a number of DimensionAttributes. The Base 'WorkingTime' for instance has the three DimensionAttributes 'regular working time,' 'planned working time,' and the 'days of absence.'

[3] Sum of fulltime equivalent = actual working time / planed working time.



```
Java - md2auml-profile/model/MD2ASchema-simplified.xmi - Eclipse
File  Edit  Navigate  Search  Project  Run  Sample Reflective Editor  Window  Help

Package E  Metamode                          MD2ASchema-simplified.xmi
cwm-relational-package                       platform:/resource/md2auml-profile/model/MD2ASchema-simplified.xmi
  src                                          MD2A Schema HCM Project
  JRE System Library [JavaSE-1.6]                Fact Employees
  Plug-in Dependencies                           Fact Person
  META-INF                                         Fact Attribute NumberOfPersons
  model                                            Fact Attribute SumOfFullTimeEquivalent
    HCM-logical-schema.xmi                       Dimension PersonalData (PD)
    orgomg.ecore                                 Dimension JobCategory (JC)
md2auml-2-cwmrelp                                Dimension Time (T)
  run-config                                     Dimension WorkingTime (WT)
  trace                                          Dimension OrganisationChart (OC)
  transforms                                     Base PD_EmployeeNumber
    md2aTOcwmrelp.qvto                           Base JC_JobDescription
md2auml-profile                                  Base JC_JobCategory
  src                                            Base JC_JobMainGroup
  JRE System Library [jre7]                      Base T_Day
  Plug-in Dependencies                           Base T_Week
  META-INF                                       Base T_Month
  model                                          Base T_Year
    MD2ASchema-simplified.xmi                    Base WT_WorkingTime
    MD2ASchema.xmi                                 OID WorkingTime
    md2aumlprofile.ecore                           Descriptor Employees Working Time
    md2aumlprofile.ecorediag                       Dimension Attribute Regular Working Time
    md2aumlprofile.genmodel                        Dimension Attribute Actual Working Time
  build.properties                                 Dimension Attribute Days of Absence
  plugin.properties                              Base OC_Chair
  plugin.xml                                     Base OC_Institute
                                                 Base OC_Department
```

Fig. 2. Partial Conceptual Schema of the Case Study

5.2 Logical Data Warehouse Design and Transformation

The relevant part of the CWM meta-model is illustrated in Fig. 3. It offers concepts for modeling relational data structures. Star and snowflake schema are well known alternatives for the description of data warehouse databases. The MD²A specification in [17] and [6] focuses on the derivation of star schemas from the conceptual schema. The developers of MD²A defined corresponding transformation rules in QVT relational language. The relevant rules which are implemented in the prototype are briefly introduced and explained by the application to the case study. The resulting logical schema is shown in Fig. 4.

Dimension2Table: Each Dimension is mapped to a Table object which is denoted by "D_" concatenated with the Dimension's name. An additional column and a constraint for the primary key are added to the table. The table 'D_WorkingTime,' for example, is derived from the dimension 'WorkingTime' and contains a primary key constraint, 'PK_WorkingTime' at column 'ID_WorkingTime.' Subsequently it triggers the transformation Base2Column.

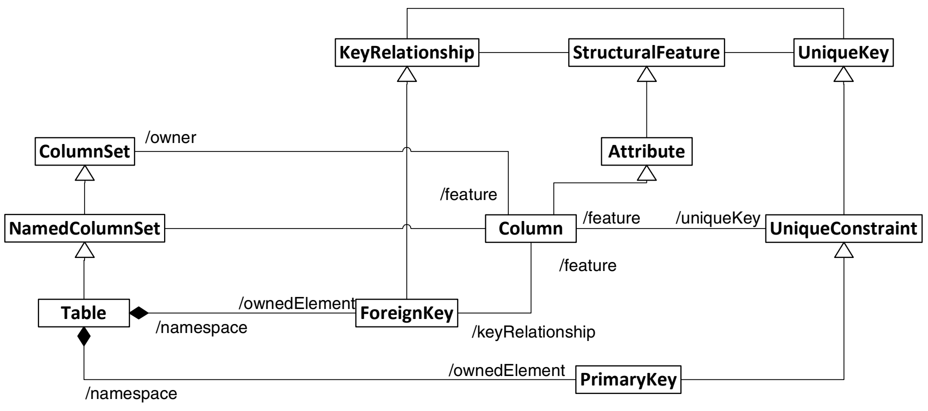
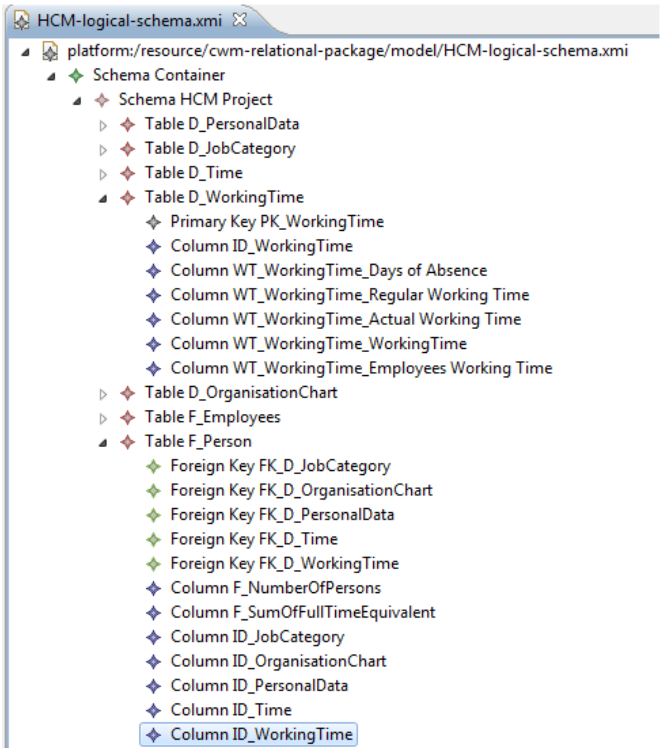


Fig. 3. Part of the Relational Package of the CWM [6]

Base2Column: For each Base, all referenced DimensionAttributes are mapped to columns. For example, the DimensionAttributes 'Regular Working Time' and 'Days of Absence' result in the columns 'WT_WorkingTime_Regular,' 'WT_Working Time_Actual' and 'WT_WorkingTime_Days of Absence.'



HCM-logical-schema.xmi

- platform:/resource/cwm-relational-package/model/HCM-logical-schema.xmi
  - Schema Container
    - Schema HCM Project
      - Table D_PersonalData
      - Table D_JobCategory
      - Table D_Time
      - Table D_WorkingTime
        - Primary Key PK_WorkingTime
        - Column ID_WorkingTime
        - Column WT_WorkingTime_Days of Absence
        - Column WT_WorkingTime_Regular Working Time
        - Column WT_WorkingTime_Actual Working Time
        - Column WT_WorkingTime_WorkingTime
        - Column WT_WorkingTime_Employees Working Time
      - Table D_OrganisationChart
      - Table F_Employees
      - Table F_Person
        - Foreign Key FK_D_JobCategory
        - Foreign Key FK_D_OrganisationChart
        - Foreign Key FK_D_PersonalData
        - Foreign Key FK_D_Time
        - Foreign Key FK_D_WorkingTime
        - Column F_NumberOfPersons
        - Column F_SumOfFullTimeEquivalent
        - Column ID_JobCategory
        - Column ID_OrganisationChart
        - Column ID_PersonalData
        - Column ID_Time
        - Column ID_WorkingTime

Fig. 4. Resulting Logical Schema of the Case Study

Fact2Table: Each Fact is mapped to a Table. For each referenced Dimension a Column is created and the transformation Aggregation2Foreignkey is called. The example in Fig. 4 illustrates this for the fact table 'F_Person' with its foreign key constraints and the corresponding columns. To reference the dimension table 'D_JobCategory,' the foreign key constraint 'FK_D_JobCategory' at column 'ID_JobCategory' is contained. Afterwards each FactAttribute of the fact object is mapped to a column object and is associated with the fact table. According to this, the fact table 'F_Person' contains the facts 'F_NumberOfPersons' and 'F_SumOfFullTimeEquivalent.'

6 Discussion of the Case Study Results

In this chapter we discuss the results of the MD²A application to our case study. We will discuss the results as follows: (i) functional requirements, (ii) non-functional requirements, and (iii) organizational requirements.

(i) At first, we have to ask whether the resulting logical schema is appropriate to satisfy the functional requirements of our users, stated in chapter 3. It is obvious that the requirements ***F1*** and ***F2*** can be answered with the resulting logical schema in Fig. 4. Therefore, we can state that in our case the MD²A approach transformed a semantically valid conceptual schema into a semantically equivalent logical schema, without manual intervention and loss of analytical capabilities.

(ii) The second point of our discussion concerns the non-functional requirement ***NF1***. It was stated that specific reports have to be delivered under certain time constraints. In the following we focus on logical design factors and not on hardware performance or capacity utilization. We assume that a report fulfilling the requirement ***F1*** has to be delivered in less than one minute and that this cannot be held by the given conceptual schema and the resulting implementation. The analysis (***F1***) utilizes the dimension 'D_WorkingTime.' In the current form this dimension has three dimension attributes 'planned working time,' 'actual working time' (considering part time contracts) and the 'days of absence.' These attributes also track the changes over time (i.e. slowly changing dimension of type 2 [15]). This means that for each change in an instance of a dimension attribute a new row is added to the dimension table. In our case the dimension attribute 'days of absence' changes much more quickly over time than the other attributes. Depending on the size of the dimension and the fact table, response time could be improved by separating this dimension attribute into its own dimension table. This concept of rapidly changing dimensions is described in [15]. As argued in section 2, such a restructuring of the multidimensional schema should be considered in the logical schema. The most obvious solution is to adjust the logical schema manually. But these changes will get lost by regeneration of the logical structures. Adjusting the conceptual model will lead to an additional dimension for the working time with no functional requirement as background. This contradicts the idea of a conceptual model focusing on business requirements and facilitating communication (cf. section 2). Hence, within a more complex project all response time requirements will be realized through additional dimensions within a conceptual model. Another option could be the parameterization of the transformation or the annotation of the conceptual schema targeted at deriving optimized logical structures. But neither the conceptual meta-model nor the transformation rules of MD²A support these features. These arguments also hold for the other structural optimizations such as materialized views or junk dimensions as mentioned in section 2.

(iii) Finally, we evaluate the organizational requirements (***O1***, ***O2***) within the case. The first requirement was the prototype driven development of the data warehouse system. This requirement is considered as supported by the MD²A approach, because the time between the deployment of the results and further discussion of open issues between designer and user is reduced through automatic transformations. But this is only the case for the early phases of a DW project, when optimizations on the logical level are not necessary. The organizational requirement ***O2*** was that the conceptual schema would very likely be changed in the production phase and that this should be supported by the approach. The MD²A approach can be used to manipulate the existing conceptual schema and then generate the logical schema and the actual code from it. But this fails for two reasons: First, when regenerating a logical schema from the changed conceptual schema all manual modifications (e.g. materialized views, junk dimensions, and rapidly changing dimensions) to the design will get lost. Second, in a production phase of a data warehouse system, the multidimensional data structures will hold analytical data. Depending on the architecture of the DW, a full-reload of a star schema via the ETL process could be more or less difficult, or even impossible, when the data is not redundantly held in another database. But even so, the problem is only shifted to another part of the DW architecture.

7 Conclusion and Future Work

Based on the insights of the presented case study we have seen that applying the Model Driven Architecture to the development of multidimensional schemas is feasible and useful. The MD²A approach provides a solid base, but further research is necessary to gain broader praxis relevance. As result of our work we can conclude that the MD²A approach is well suited for DW projects that encompass only the early design stages. The approach gives the opportunity to quickly implement prototypes and so shorten the time between analyzing requirements and discussing their implementation with the end user. In that situation the automatic generation of multidimensional schemas enhances the productivity of DW development. But there is a number of projects in practice which share the same characteristics, especially in later phases, with the presented case study:

- dynamic changing functional requirements during production phase
- necessity of restructuring the logical schema due to non-functional requirements
- difficult or impossible complete data reload in production phase

Due to the loss of logical optimizations, the possible need to manually adjust the physical data structures, and missing features to support changing functional requirements during production phase, we can state that the MD²A approach is not well suited for projects sharing the mentioned characteristics. This also is the case for other multidimensional MDA approaches (cf. [2-5]). Neither of them considers logical optimizations or maintenance tasks during production phase closely. In our opinion, these are the main features which should be supported by a model driven multidimensional design approach in praxis.

To close this gap, future work has to be done in the fields of generating optimized data structures and handling varying functional requirements at runtime. In our ongoing work we plan to evaluate the parameterizing of transformations in order to automatically generate optimized logical schema. An alternative solution could be the tracing of manual adjustments at the logical level and merge them to new logical schema versions. For the case of changing functional requirements at production phase we plan an evolutionary concept. This means the incorporation of such changes at conceptual level and their transformation to extensions of the existing physical tables in a way that does not require existing analytical data to be reloaded or recalculated.

References

1. Inmon, W.H.: Building the data warehouse, 4th edn., pp. 29–70. Wiley, Indianapolis (2005)
2. Fernandes, L.A., Neto, B.H., Fagundes, V., Zimbrao, G., de Souza, J.M., Salvador R.: Model-Driven Architecture Approach for Data Warehouse. In: 2010 Sixth International Conference on Autonomic and Autonomous Systems, pp. 156–161 (2010)
3. Gluchowski, P., Kurze, C., Schieder, C.: A Modeling Tool for Multidimensional Data using the ADAPT Notation. In: Proceedings of the 42nd Hawaii International Conference on System Sciences, pp. 1–10 (2009)
4. Zepeda, L., Cecena, E., Quintero, R., Zatarain, R., Vega, L., Mora, Z., Clemente, G.G.: A MDA Tool for Data Warehouse. In: Proceedings of the 2010 International Conference on Computational Science and Its Applications, pp. 261–265. IEEE Computer Society (2010)
5. Choura, H., Feki, J.: MDA Compliant Approach for Data Mart Schemas Generation. In: Bellatreche, L., Mota Pinto, F. (eds.) MEDI 2011. LNCS, vol. 6918, pp. 262–269. Springer, Heidelberg (2011)
6. Mazón, J.-N., Trujillo, J., Serrano, M., Piattini, M.: Applying MDA to the development of data warehouses. In: Proceedings of the 8th ACM International Workshop on Data Ware-Housing and OLAP, pp. 57–66. ACM, Bremen (2005)
7. Luján-Mora, S., Trujillo, J.: A Data Warehouse Engineering Process. In: Yakhno, T. (ed.) ADVIS 2004. LNCS, vol. 3261, pp. 14–23. Springer, Heidelberg (2004)
8. Soler, E., Trujillo, J., Fernández-Medina, E., Piattini, M.: A set of QVT relations to transform PIM to PSM in the Design of Secure Data Warehouses. In: Second International Conference on Availability, Reliability and Security, pp. 644–654 (2007)
9. Mazón, J.-N., Trujillo, J., Lechtenbörger, J.: A set of QVT relations to assure the correctness of data warehouses by using multidimensional normal forms. In: Proceedings of the 25th International Conference on Conceptual Modeling, pp. 385–398. Springer, Tucson (2006)
10. Glorio, O., Trujillo, J.: An MDA Approach for the Development of Spatial Data Warehouses. In: Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery, pp. 23–32. Springer, Turin (2008)

11. Mazón, J.-N., Pardillo, J., Trujillo, J.: Applying transformations to model driven data warehouses. In: Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery, pp. 13–22. Springer, Krakow (2006)
12. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, pp. 3–10. ACM, Arlington (2006)
13. Golfarelli, M., Rizzi, S., Pagliarani, C.: Data Warehouse Design. In: Modern Principles and Methodologies, pp. 43–60. McGraw-Hill, New York (2009)
14. Golfarelli, M., Rizzi, S.: A methodological framework for data warehouse design. In: Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP, pp. 3–9. ACM, Washington, D.C. (1998)
15. Kimball, R., Ross, M.: The data warehouse toolkit. The complete guide to dimensional modeling. Wiley, New York (2002)
16. Object Management Group: MDA Guide Version 1.0.1, `http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf`
17. Luján-Mora, S., Trujillo, J., Song, I.-Y.: A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering 59, 725–769 (2006)
18. Object Management Group: Common Warehouse Metamodel (CWM) Specification, `http://www.omg.org/spec/CWM/1.1/PDF/`
19. Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, `http://www.omg.org/spec/QVT/1.1/PDF/`

Integrating ETL Processes from Information Requirements

Petar Jovanovic[1], Oscar Romero[1], Alkis Simitsis[2], and Alberto Abelló[1]

[1] Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{petar,oromero,aabello}@essi.upc.edu
[2] HP Labs, Palo Alto, CA, USA
alkis@hp.com

Abstract. Data warehouse (DW) design is based on a set of requirements expressed as service level agreements (SLAs) and business level objects (BLOs). Populating a DW system from a set of information sources is realized with extract-transform-load (ETL) processes based on SLAs and BLOs. The entire task is complex, time consuming, and hard to be performed manually. This paper presents our approach to the requirement-driven creation of ETL designs. Each requirement is considered separately and a respective ETL design is produced. We propose an incremental method for consolidating these individual designs and creating an ETL design that satisfies all given requirements. Finally, the design produced is sent to an ETL engine for execution. We illustrate our approach through an example based on TPC-H and report on our experimental findings that show the effectiveness and quality of our approach.

1 Introduction

Organizations share their common Data Warehouse (DW) constructs among users of different skills and needs, involved in different parts of the business process. Information requirements coming from such users may consider different analytical perspectives; e.g., Sales is interested in analyzing suppliers data, while Finance analyzes different data like cost or net profit. Complex business models, often make these data intertwined and mutually dependent. Taking into account dynamic enterprise environments with constantly posed information requirements, we need a means for dealing with the complexity of building a complete target schema and supporting extract-transform-load (ETL) process from the early design phases. In addition, due to typical maintenance tasks of such constructs, a great challenge is to provide the designer with the means for dynamic and incremental building of such designs considering real business needs.

In this paper, we focus on ETL design and present our approach to the incremental consolidation of ETL processes, each created to satisfy a single business requirement. For new projects, we create the ETL design from scratch based on a given set of requirements. If an ETL process already exists, we build upon

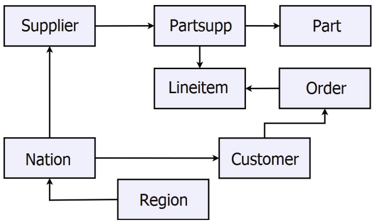


Fig. 1. TPC-H Schema

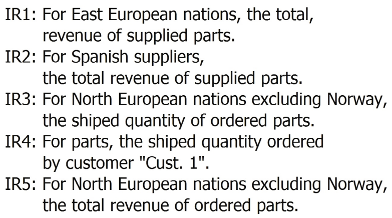

IR1: For East European nations, the total,
revenue of supplied parts.
IR2: For Spanish suppliers,
the total revenue of supplied parts.
IR3: For North European nations excluding Norway,
the shiped quantity of ordered parts.
IR4: For parts, the shiped quantity ordered
by customer "Cust. 1".
IR5: For North European nations excluding Norway,
the total revenue of ordered parts.

Fig. 2. Information Requirements

it and extend it according to new or changed requirements. For these tasks, we propose the *CoAl* algorithm. As 'coal' is formed after the process and extreme compaction of layers of partially decomposed materials (src. Wikipedia), *CoAl* processes partial ETL designs, each satisfying a single business requirement, and consolidates them into a unified design satisfying the entire set of requirements. The algorithm is flexible and applies various equivalence rules to align the order of ETL operations for finding the appropriate matching part among different input ETL designs. At the same time, it accounts for the cost of ETL designs, searching for near-optimal solutions. At the end, the solution suggested by *CoAl* is sent to an ETL engine for execution. Hence, we provide a novel, end-to-end, requirement-driven solution to the ETL design problem. Our experiments show the effectiveness and usefulness of the proposed method.

Contributions. In particular, our main contributions are as follows.

- We present our approach to the incremental integration of new information requirements into new or existing ETL designs.
- We introduce a novel consolidation algorithm, called *CoAl*, that deals with both structural and content comparison of ETL designs, identifies the maximal matching area among them, and finally, taking into account the cost, produces an ETL design satisfying all requirements.
- We show a set of experiments showing the effectiveness and quality of *CoAl*.

Outline. Section 2 introduces a running example used throughout the paper. Section 3 presents the ETL design consolidation problem, describes equivalence rules, and formalizes operation comparisons. Section 4 presents the *CoAl* algorithm and Section 5 reports on our experimental findings. Finally, Sections 6 and 7 discuss related work and conclude the paper, respectively.

2 Running Example

To illustrate our approach, we use a scenario based on the TPC-H schema [2]. Figure 1 shows an abstraction of the TPC-H schema. Assuming a set of five requirements (IR1, ..., IR5) over the TPC-H schema, as shown in Figure 2, we describe how we automatically produce a design that fulfills all five requirements.

First, we create an ETL design for each of these requirements. In the literature there are methods for dealing with such a task (e.g., [14]). Having a design

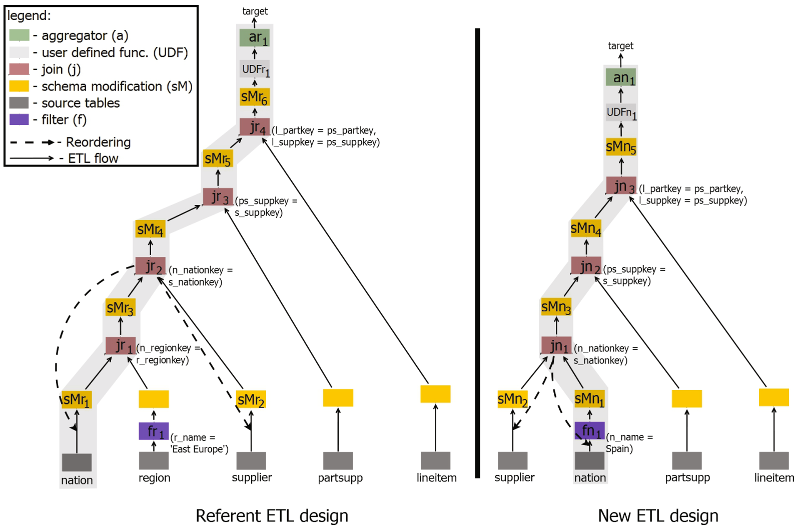

legend:
- aggregator (a)
- user defined func. (UDF)
- join (j)
- schema modification (sM)
- source tables
- filter (f)
- Reordering
- ETL flow
target
ar1
UDFr1
sMr6
jr4 (l_partkey = ps_partkey, l_suppkey = ps_suppkey)
sMr5
jr3 (ps_suppkey = s_suppkey)
sMr4
jr2 (n_nationkey = s_nationkey)
sMr3
jr1 (n_regionkey = r_regionkey)
sMr1
fr1 (r_name = 'East Europe')
sMr2
nation
region
supplier
partsupp
lineitem
Referent ETL design
an1
UDFn1
sMn5
jn3 (l_partkey = ps_partkey, l_suppkey = ps_suppkey)
sMn4
jn2 (ps_suppkey = s_suppkey)
sMn3
jn1 (n_nationkey = s_nationkey)
sMn2
sMn1
fn1 (n_name = Spain)
New ETL design

Fig. 3. ETL designs satisfying IR1 and IR2

per requirement at hand, in this paper, we focus on integrating the individual ETL designs into a design that satisfies all requirements. Considering Figure 3, we define the *referent* ETL design as the integrated ETL design for a number of requirements already modeled (we start from IR1) and the *new* ETL design as the design for a requirement not integrated yet (IR2). In terms of graphical notation, the gray bottom rectangles represent data sources, whereas the other boxes represent operations. The design for IR1, say G1, contains four join operations, jr_k, $k{=}1\ldots4$. The design for IR2, G2, has three joins jn_l, $l{=}1\ldots3$. Both designs contain other operations like filters, and so on.

Observe that the two designs have a number of common operations, like for example those on the paths involving the source *nation* (shaded paths in Figure 3). For both performance and maintainability purposes, we need to create an alternative, equivalent design having the minimum number of overlapping operations. Figure 4 (left) shows an alternative ETL design that satisfies both IR1 and IR2 requirements and the common computation is realized only once.

Once the designs satisfying IR1 and IR2 have been integrated, we iteratively proceed with the remaining requirements, IR3, IR4, and IR5, until we consolidate all five ETL designs into one (assuming that all five designs share operations; otherwise, a design is not merged with the others). For this example, the ETL design that satisfies all five requirements is shown in the right part of Figure 4.

For the sake of presentation we discuss here functional requirements, but our approach works seamlessly for non-functional requirements too. For example, a requirement regarding availability might lead to a fault-tolerant design that

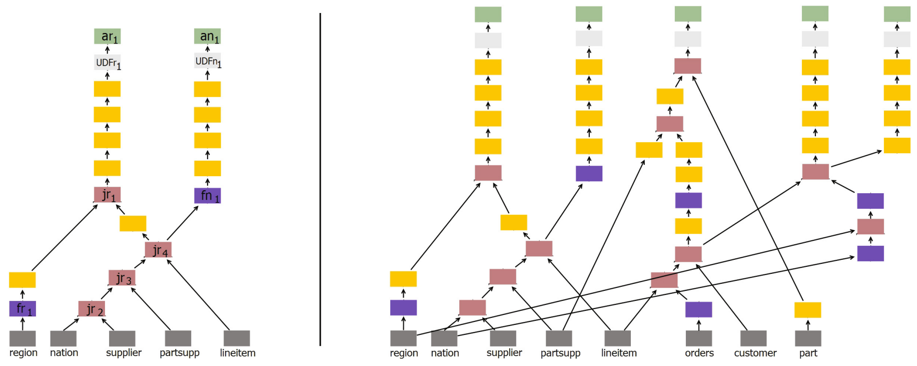


Fig. 4. ETL designs satisfying IR1 and IR2 (left) and all five requirements (right)

uses replication. Such a design would involve a splitter and a voter operations –to create and merge back the replicas, respectively– whereas the flow fragment between these two operations would have been replicated by a factor –for example– of 3 (triple-modular redundancy). In such cases, the design integration proceeds using the same techniques we describe next by means of the running example.

3 The Design Consolidation Problem

In this section, we describe the problem of consolidating ETL designs satisfying single requirements. We first discuss the challenges that need to be solved and then, we formally present the theoretical underpinnings regarding design equivalence and operation comparisons.

3.1 Goals and Challenges

Typically, an ETL design is modeled as a directed acyclic graph. The nodes of the graph are data stores and operations, and the graph edges represent the data flow among the nodes.

Intuitively, for consolidating two ETL designs, a referent G_1 and a new G_2 designs, we need to identify the maximal overlapping area in G_1 and G_2. Therefore, we proceed as follows:

1. First, we identify the common source nodes between G_1 and G_2. In terms of our running example, let us assume that G_1 satisfies IR1 and G_2 satisfies IR2. The common sources for the two designs are *nation*, *supplier*, *partsupp*, and *lineitem* (see Figure 3).
2. For each source node, we consider all paths up to a target node and search for common operations in both designs. Starting with the *nation* source node, we identify the paths up to the target in both designs (shaded paths in Figure 3). In these paths, we search for common operations that could be consolidated into a single operation in the new design.

Deciding which operations can be consolidated and how is not an easy task. If two operations, each placed in a different design, can be matched, then we have a *full match*. For example, the join operation jn_3 in the new design G_2 fully matches the join operation jr_4 in the referent design G_1 (see also Figure 3). If two operations, one in the referent design and the other in the new design, partially overlap, then we have a *partial match*. For example, if jn_3 involved only predicate $ps\_partkey = l\_partkey$, then the referent design would partially overlap, since in that case jr_4 would be more specific and thus would provide only a subset of the necessary results set.

For being able to guarantee full or partial matching, we should also look at the operations performed before the ones considered for the matching; i.e., we need to check the input paths of each operation considered for matching. For example, we cannot consolidate operations jr_2 and jn_1 until we ensure that their predecessors have been already fully matched too. In order to enable better matching, we also consider design restructuring by moving operations before or after those considered for matching. This task is performed by our *CoAl* algorithm described in detail in Section 4.

Before presenting *CoAl*, we first describe two theoretical aspects that set the foundations of our method. In 3.2, we show how to reorder operations within the same design, in order to facilitate the search for full or partial matchings. In 3.3, we show how partial and full matches may be identified between operators.

3.2 Equivalence Rules

Reordering operations within an ETL design may be desirable for several reasons; e.g., for improving performance by pushing selective operations early in the flow. Here, we focus on design restructuring with the goal of favoring operation matching between two different designs.

In order to change the structure of a design, we need to ensure that the change, called transition, is valid and leads to a semantically equivalent design. For this reason, all possible transitions obey to a set of equivalence rules, which guarantees the equivalence of designs after a transition has taken place. We consider some of the transitions previously proposed in the context of ETL optimization [15]. These transitions include: *swap*, *distribute*, and *factorize*. Swap (swp) interchanges the position of two adjacent unary operations. Factorize (fct) and distribute (dst) represent the factorization and distribution of unary operations over an adjacent n-ary one. It is proven that these transitions are sound and produce equivalent designs, as long as some conditions based on the schemata of operations hold. For example, two unary operations o_1 and o_2 cannot be swapped if o_2 has as a parameter an attribute generated by o_1. For further details and a complete list of these conditions, we refer the interested reader to [15]. In addition, we use another two transitions: *association* (asc), which refers to the associativity rule of n-ary operations of the same kind (e.g., joins), and *n-ary distribution* (distr), which refers to the distributive rule between n-ary operators (e.g., join and union), all with their well-known properties.

Table 1. Equivalence rules for the running example

oper.	**f**	**sM**	**j**	$\cup$	**a**	**UDF**	**SK**
f	$\surd_{swp}$	$\sim_{swp}$	$\surd_{dst/fct}$	$\surd_{dst/fct}$	$\sim_{swp}$	$\surd_{swp}$	$\surd_{swp}$
sM	$\surd_{swp}$	$\times$	$\surd_{dst/fct}$	$\surd_{dst/fct}$	$\sim_{swp}$	$\surd_{swp}$	$\surd_{swp}$
j	$\surd_{dst/fct}$	$\sim_{dst/fct}$	$\surd_{asc}$	$\times$	$\sim_{dst/fct}$	$\sim_{dst/fct}$	$\sim_{dst/fct}$
$\cup$	$\surd_{dst/fct}$	$\surd_{dst/fct}$	$\surd_{distr}$	$\surd_{asc}$	$\times$	$\surd_{dst/fct}$	$\sim_{dst/fct}$
a	$\sim_{swp}$	$\sim_{swp}$	$\sim_{dst/fct}$	$\times$	$\times$	$\sim_{swp}$	$\sim_{swp}$
UDF	$\sim_{swp}$	$\sim_{swp}$	$\sim_{dst/fct}$	$\surd_{dst/fct}$	$\sim_{swp}$	$\sim_{swp}$	$\sim_{swp}$
SK	$\sim_{swp}$	$\sim_{swp}$	$\sim_{dst/fct}$	$\surd_{dst/fct}$	$\sim_{swp}$	$\sim_{swp}$	$\times$

Table 1 shows the applicability of equivalence rules for an example set of operations (we explain them in 3.3). Although, for the sake of presentation, we list here a limited set of operations (as those needed for the running example), the transitions work for a much broader set of operation as discussed in the literature (e.g., [15,17]). The table reads as follows. For each cell we present how the operation of the column can be rearranged and pushed down the adjacent operation of the row. A tick ($\surd$) means that the unconditional equivalence rule(s) exists between these operations. The additional label(s) besides this symbol refer to which transitions are allowed. If there is a conflict and no equivalence rule can be applied over operations, the cell is crossed ($\times$). Furthermore, in the cases of partial conflicts the cell is marked with ($\sim$) and has an appropriate label. This happens when certain equivalence rules can be applied only if certain conditions hold. In all cases, *CoAl* considers only valid transitions based on the equivalence rules.

For example, note that reordering sMr_4 in the referent design of the running example is not allowed, since it projects out *s_suppkey* included in the predicate of jr_3. As another example, a filter and an aggregator can be swapped, assuming that the input schema of filter does not have an attribute contained in the grouping attributes of the aggregate. In addition, it is possible to dst/fct aggregate over join if afterwards the specified set of functional dependencies that ensures the equivalence of such transition holds [20]. Other operations behave similarly.

3.3 Operation Comparisons

Next, we describe how we determine whether between two operations, say o_{ref} (placed in the referent design) and o_{new} (placed in the new design) there exists either full or partial or no match.

As we discussed before (see 3.1), two operations can be consolidated if they match and if their input data flows also coincide. Even if they do not coincide at first, after finding the matching, either full or partial, we try design restructuring based on the equivalence rules until we meet this condition (if it is possible). We discuss this in the next section. Here, we describe comparison of two operations o_{ref} and o_{new}, without considering their input flows.

Figure 5 illustrates the four possible outcomes of operation comparison.

(1) The compared operations are equal: $o_{ref} = o_{new}$. Then, we consolidate the two operations as a single one in the integrated design.

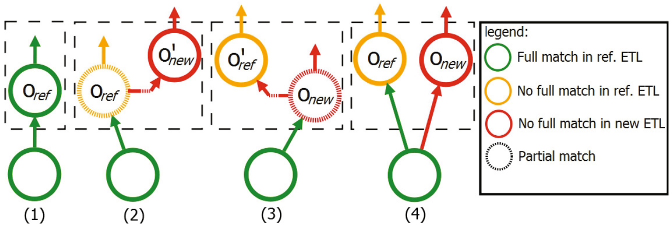


Fig. 5. Integration of the operations

(2) The results of o_{new} can be obtained from the results of o_{ref}. Then, both operations can be partially collapsed as depicted in Figure 5. Hence, the output of o_{new} can be computed from the output of o_{ref} (i.e., $o_{ref} \prec o_{new}$) and thus, it partially benefits from the transformations already performed by o_{ref}. Also, the consolidation of the partially matched operation o_{new} may involve a transformation of this operation for obtaining the original output data. For example, if jr_4 in Figure 3 involved only the predicate $ps\_partkey = l\_partkey$ we could then only identify partial matching between jn_3 and jr_4 and the consolidation of these operations would require an extra operation to filter data according to the remaining predicate ($ps\_suppkey = l\_suppkey$).

(3) The results of o_{ref} can be obtained from the results of o_{new} ($o_{new} \prec o_{ref}$).

(4) Finally, it may happen that neither o_{new} can benefit from o_{ref} nor the opposite. Then, the two operations cannot be consolidated. In such cases, we use a *fork* in the already matched ETL subset, as shown in Figure 5(4). (Note that the fork is implemented as a copy-partitioning operation in the physical design.)

In general, each operation is characterized by its *input* (I) and *output* (O) schemata (see also [15]).We also consider the *semantics* (S) involved in the computation performed by this operation. To express the wide complexity of ETL flows we can define the semantics of their operations as *programs* with corresponding *precondition* (Pre) and *postcondition* (Post) predicates. Accordingly, we can formally represent an ETL operation o as $\mathbf{o}(I, O, S, Pre, Post)$.

- $\mathbf{o}_1(I_1, O_1, S_1, Pre_1, Post_1) = \mathbf{o}_2(I_2, O_2, S_2, Pre_2, Post_2)\ iff\ I_1 = I_2 \wedge O_1 = O_2 \wedge Pre_1 \equiv Pre_2 \wedge Post_1 \equiv Post_2$;
- $\mathbf{o}_1(I_1, O_1, S_1, Pre_1, Post_1) \prec \mathbf{o}_2(I_2, O_2, S_2, Pre_2, Post_2)\ iff\ \exists \mathbf{o}_3(I_3, O_3, S_3, Pre_3, Post_3) : I_1 = I_2 \wedge O_1 = I_3 \wedge O_3 = O_2 \wedge Pre_1 \equiv Pre_2 \wedge Post_3 \equiv Post_2 \wedge Post_1 \Rightarrow Pre_3$;

The definition of a specific operation semantics (S) along with the corresponding postconditions and preconditions predicates and their implications ($\Rightarrow$) are provided by template definitions for operations. Hence, any given ETL design uses instances of these operations that inherit such properties from their generic template definitions. This process is straightforward and a detailed discussion on this topic falls out of the scope of this paper.

Next, we show formal definitions for operation comparison for the operations shown in Table 1. Due to space considerations, we do not elaborate here on other operations, but the process is similar.

- **Filter -** $\mathbf{f}_{\psi}(R)$
  For comparing filter operations, besides the input schema, we also check the comparison of included predicates ψ. The comparison of the equivalence or logical implication of these predicates ($\psi_1 \Leftarrow \psi_2$) can be facilitated by generic reasoners. We compare filter operations as follows:
  - $\mathbf{f}_{\psi_1}(R) = \mathbf{f}_{\psi_2}(S)$ iff R=S $\wedge$ $\psi_1 \equiv \psi_2$;
  - $\mathbf{f}_{\psi_1}(R) \prec \mathbf{f}_{\psi_2}(S)$ iff R=S $\wedge$ $\psi_1 \Leftarrow \psi_2$;
- **Schema Modification -** $\mathbf{sM}_{a1,a2,..,an}(R)$
  For comparing schema modifications, besides the input relations, we also compare the attributes that are modified. Therefore, we compare schema modification operations as follows:
  - $\mathbf{sM}_{a1,..,an}(R) = \mathbf{sM}_{b1,..,bm}(S)$ iff R=S $\wedge$ {a1,..,an} = {b1,..,bm};
  - $\mathbf{sM}_{a1,..,an}(R) \prec \mathbf{sM}_{b1,..,bm}(S)$ iff R=S $\wedge$ {a1,..,an} $\supset$ {b1,..,bm};
- **Join -** $R\ \mathbf{j}_{\psi}\ S$
  To compare joins, we take into account the commutative property that applies over the inputs of a join. As with filter, we compare the corresponding join predicates. Thus, we compare joins as follows:
  - $P\ \mathbf{j}_{\psi_1}\ Q = R\ \mathbf{j}_{\psi_2}\ S$ iff ((P=R $\wedge$ Q=S) $\vee$ (P=S $\wedge$ Q=R)) $\wedge$ $\psi_1 \equiv \psi_2$;
  - $P\ \mathbf{j}_{\psi_1}\ Q \prec R\ \mathbf{j}_{\psi_2}\ S$ iff ((P=R $\wedge$ Q=S) $\vee$ (P=S $\wedge$ Q=R)) $\wedge$ $\psi_1 \Leftarrow \psi_2$;
- **Union -** R $\cup$ S
  To compare unions, we only compare their input relations, as they do not have any additional parameters defined. Here, we also consider the commutative property. Thus, we compare unions as follows:
  - $P \cup Q = R \cup S$ iff ((P=R $\wedge$ Q=S) $\vee$ (P=S $\wedge$ Q=R));
- **Aggregator -** ${}_{g1,..,gm}\mathbf{a}f1(A1'),..,fk(Ak')\ (R)$
  To compare aggregators, besides the input relations, we also compare the grouping attributes regarding equality or functional dependency between them ($g_i \rightarrow t_i$). Currently, we consider the set of aggregation functions to be equal. However, this can be extended considering the class of expandable aggregation functions, discussed in [4]. Thus, we compare them as follows:
  - ${}_{g1,..,gn}\mathbf{a}...\ (R) = {}_{t1,..,tm}\mathbf{a}...\ (S)$ iff R=S $\wedge$ m=n $\wedge$ $\forall i = 1..m, g_i{=}t_i$;
  - ${}_{g1,..,gn}\mathbf{a}...\ (R) \prec {}_{t1,..,tm}\mathbf{a}...\ (S)$ iff R=S $\wedge$ m$\leq$n $\wedge$ $\forall i = 1..m, g_i = t_i \vee g_i{\rightarrow}t_i$;
- **User Defined Function (UDF) - UDF**(R)
  A udf is expressed as ${}_{f:I\rightarrow O}\mathbf{o}(R)$. For comparing udfs we consider their behavior over the input records (r), as follows:
  - ${}_{f_1:I_1\rightarrow O_1}\ \mathbf{o}_1$(R) = ${}_{f_2:I_2\rightarrow O_2}\ \mathbf{o}_2$(S) iff $R = S \wedge \forall r \in R : f_1(r) = f_2(r)$;
  - ${}_{f_1:I_1\rightarrow O_1}\ \mathbf{o}_1$(R) $\prec$ ${}_{f_2:I_2\rightarrow O_2}\ \mathbf{o}_2$(S) iff $\exists_{f_3:I_3\rightarrow O_3}\ \mathbf{o}_3$(P):$R = S \wedge O_1 = I_3 \wedge O_3 = O_2 \wedge \forall r \in R : f_3(f_1(r)) = f_2(r)$;
- **Surrogate Key Assignment (SK) - SK**(R, S)
  Surrogate key assignment (SK) is a typical ETL operation that
  joins the incoming data (R) with a lookup dimension table (S) and replaces the pair "source of data, primary key"($value$) with a unique identifier for DW called "surrogate key"(sk). If a surrogate key does not exist for the pair "source, primary key", then a new surrogate key is generated, typically by a function producing values like $max(SK) + 1$. The comparison is as follows:

- $\mathbf{SK}(R_1, S_1) = \mathbf{SK}(R_2, S_2)$ *iff* $R_1 = R_2 \wedge (\forall t \in R_1 : (\exists t' \in S_1, t[value] = t'[value] \wedge \exists t'' \in S_2, t[value] = t''[value]) \rightarrow t'[sk] = t''[sk])$;

Due to specific semantics of the SK transformations, the above comparison does not actually test equality of two SK transformations, but their ability to be consolidated. Therefore, we define that two SK transformations can be consolidated iff there is no conflict between their lookup tables, i.e., iff the SK values can be found either in one or in none of the tables.

4 Consolidation Algorithm

The *CoAl* algorithm looks for all matching opportunities between operators from the referent and new designs and at the end, it produces a consolidated design. For each source node, we explore its paths up to a target following a *topological order* of the nodes in the design. At each iteration of the algorithm, we only match two operations (one from each design), and we only add this match to the final result if and only if all previous nodes in the path have been fully matched. To ensure this, we proceed using the equivalence rules between the operations at hand and taking into account the performance cost of the design.

Returning to the running example that shows how we consolidate the designs for IR1 and IR2 (see Figure 3), we identify a full match between jr_2 and jn_1 operations. However, their input paths have not been fully matched yet. One solution to handle this is to check if we can push jr_2 and jn_1 down to their respective sources (*supplier* and *nation*). This is possible because on the one side jr_2 may move over sMr_2, sMr_3, jr_1, and sMr_1, while on the other side operation jn_1 may move over sMn_1, sMn_2, and fn_1, and we may still produce equivalent designs that fulfill our constraint.

As we discussed, only when a full match is not possible (either directly or after reordering of operations), we search for a partial match. Partial matches finish our exploration in the considered branch, as we do not fulfill our constraint: fully match of the input paths is required in order to keep exploring the branch. Hence, if at the end a complete match is not found –i.e., the new ETL cannot be completely subsumed by the referent ETL– we explore the partial matchings identified and estimate their costs. The cheapest solution according to the cost model considered (discussed later in this section) is chosen for integration.

Formally, *CoAl* starts with two ETL designs, the referent and the new, and iterates following a topological order of the ETL operations, which guarantees the following two invariants:

(I_1): At each iteration, only one pair of operations can be partially or fully matched.
(I_2): A new match is added to the set of already matched operations iff the input flows of the operations involved in the new match have been fully matched in previous iterations.

These invariants have some interesting consequences. Two matched operations are eventually consolidated in the output, integrated design if the designs they

```
inputs: $G_1$, $G_2$, output: $G_{int}$
 1. matchingsQueue := matchLeafs($G_1$, $G_2$);
 2. alternativeList := $\emptyset$;
 3. while (matchingsQueue is not empty) do
    (a) currentMatchings($G_1\prime$,$G_2\prime$) := dequeue(matchingsQueue);
    (b) newOperationsForMatching(LOps_ref, LOps_new) := explore($G_1\prime$,$G_2\prime$);
    (c) if (LOps_new is empty) then
         i. insert($G_1\prime$,$G_2\prime$, $\emptyset$, 0) into alternativeList;
    (d) foreach pair($O_{new}$ from LOps_new, $O_{ref}$ from LOps_ref)
          i. if ($O_{new}$ fully matches $O_{ref}$) then
              A. $G_1\prime\prime$ =: reorder($G_1\prime$); $G_2\prime\prime$ =: reorder($G_2\prime$);
              B. enqueue(matchingsQueue, {$G_1\prime\prime$,$G_2\prime\prime$}, {$O_{new}$,$O_{ref}$});
         ii. else if ($O_{new}$ partially matches $O_{ref}$) then
              A. $G_1\prime\prime$ =: reorder($G_1\prime$); $G_2\prime\prime$ =: reorder($G_2\prime$);
              B. insert($G_1\prime\prime$, $G_2\prime\prime$, {$O_{new}$, $O_{ref}$}, Cost($G_1\prime\prime$, $G_2\prime\prime$)) into alternativeList ;
    (e) if (no matching found) then
         i. insert($G_1\prime$, $G_2\prime$, $\emptyset$, Cost($G_1\prime$, $G_2\prime$)) into alternativeList;
 4. if (alternativeList is not empty)
    (a) $G_{int}$ := integrate(cheapestAlternativeMatching);
 5. Return $G_{int}$;
```

Fig. 6. Pseudocode for *CoAl*

belong to can be reordered so that their children are fully matched. Since we are looking for the maximal overlapping area between the two designs, we can guarantee that any two operations that fully match can be immediately added to the output. The proof based on contradiction is straightforward.

Suppose that o_{ref1} and o_{ref2} are two operations from the referent design and o_{new1} and o_{new2} are two operations from the new design. Let us assume that o_{ref1} fully matches with o_{new1}, from now $pair_1$, and o_{ref2} fully matches with o_{new2}, from now $pair_2$. Both belong to the maximal overlapping area between both designs and there is no other full match left to identify. If the order to add them to the output matters, it means that one of these pairs, say $pair_1$, should be added to the result before $pair_2$. But this can only happen if no equivalence rules can be applied between the corresponding operators in each design (i.e., between o_{ref1} and o_{ref2} in the referent design and between o_{new1} and o_{new2} in the new design). In such a case, and knowing that they belong to the maximal overlapping area, $pair_1$ can be moved according to the equivalence rules down in both designs, so that all their input data flows are fully matched and consequently, they will be added to the output in the next iteration. After $pair_1$ has been integrated, we use another finite set of equivalence rules for pushing $pair_2$ down and fulfill (I_2). Thus, in the next iteration it will be also added to the output. Relevantly, this is also the proof that our algorithm will eventually finish. Due to this property we define rule R_1:

(R_1): When looking for matchings, the first two operations to be compared from the referent and the new designs are those that fulfill (I_2).

Although it favors the cheapest solutions (i.e., that do not require any reordering), R_1 nevertheless does not eliminate better solutions that may appear.

CoAl comprises four steps (see Figure 6): i) *search for the next operations to match*; ii) *compare the next operations*; iii) *reorder input designs* if a match has been found; and iv) *integrate the alternative matching* with the lowest estimated cost. The three first are executed in each iteration of the algorithm, whereas the last one is executed only once, when no match is pending.

Through the algorithm we maintain two structures. First, a priority queue that contains fully matched areas that may be further extended with new matching operations. Each queue element contains the list of matching operations together with the input ETL structures specifically reordered for such matchings. Second, a list for keeping all alternative matching combinations ending up in a partial matching found through the algorithm, along with the estimated costs of such matchings. The algorithm starts by matching the source nodes of the referent and the new designs (step 1). The comparison of the source nodes is based on the parameters that characterize them: source type, source name, and extracted fields. The steps of *CoAl* are as follows.

Search for the next operations to match. We identify the operations to be compared next (step 3b). We start with comparing operations according to (R_1). If there is no full match, the algorithm identifies all operations that can be reordered, by applying equivalence rules, and pushes them down according to (I_2) and hence it identifies different possibilities for comparing. As a result, two sets of operations to be compared (LOps_new and LOps_ref) are produced. In terms of the running example, for the paths starting from the matching source *nation* (see Figure 3), in the referent design we identify the set: (jr_1, jr_2, and sMr_1) and in the new design: (jn_1, sMn_1, and fn_1).

Compare the next operations. We then produce the cartesian product of these two sets (step 3d). For each pair, we proceed as explained in subsection 3.3 depending on the result of the comparison: (a) we can identify a *full match* (equality) (step 3(d)i); (b) a *partial match* (step 3(d)ii) or (c) *no match* (step 3e).

Reorder the input designs. If *CoAl* finds a (full or partial) match between two operations, then it tries reordering of the input designs to guarantee (I_2) (steps 3(d)iA and 3(d)iiA). Considering the running example, when we find a full match between joins jr_2 and jn_1, the algorithm pushes them down to the sources *nation* and *supplier*. *CoAl* then adds the match found to the integrated design and depending on the type of match found it proceeds as follows.

– For a full match, it enqueues back to priority queue the two designs (possibly reordered) to further extend the matching in next iterations (step 3(d)iB).
– For a partial match, it estimates the cost of such a solution and then adds it, along with its cost, to the list of integration alternatives (step 3(d)iiB).
– Finally, if there is no match, this alternative, along with its estimated cost, is also added to the list of potential integration alternatives (step 3(e)i).

Regarding a cost estimation model, *CoAl* is not tied to a specific cost model; in fact, it is extensible to any given cost model. Example cost models for ETL designs can be found in the literature (e.g., [16]).

The matching process ends when the algorithm finishes with all possible paths and the comparison among their operations (i.e., when there are no more elements in the priority queue). Alternatively, the algorithm terminates when a complete matching of the new design is identified (step 3c). This extreme case happens only when the new design is completely subsumed by the referent one and thus the cost of such an alternative (i.e., the referent design itself) is 0.

Integrate an alternative match. After the iterations finish, *CoAl* checks if there is any alternative matching. It checks the list of all possible alternatives and chooses the one with the lowest estimated cost. Then, it continues the design consolidation with that alternative (step 4a).

Finally, *CoAl* returns the consolidated design.

5 Evaluation

This section describes our prototype and reports on our experimental findings.

***CoAl* in *GEM*.** Our work revolves around *GEM*, which is a prototype for the creation of multidimensional (MD) schemata and the respective ETL design based on a given set of business requirements. In a nutshell, starting from a set of requirements expressed in a proprietary XML-like form, we semi-automatically construct the resulting MD schemata and ETL designs using Semantic Web technology for inferring the necessary mappings [14]. The outcome of this process is a conceptual MD and an ETL designs. The conceptual ETL design is encoded in an XML-like format, namely xLM, previously proposed in [19]. Our method produces one ETL design per business requirement and then, we use *CoAl* for consolidating the results into a unified ETL process.

As a next step, our prototype translates the conceptual ETL design into a physiological ETL model, expressed again in xLM, which then may be executed in an ETL engine. Figure 7 shows a physical rendition for the running example. For now, *GEM* is connected to an open source ETL engine (PDI, a.k.a. Kettle [1]). For the connection, we translate xLM to the engine-specific XML form for storing ETL metadata, and thus, we are able to import a design into the engine and execute it. Our design choice of use an XML-like encoding was made for achieving a greater extensibility, since many modern ETL engines use XML

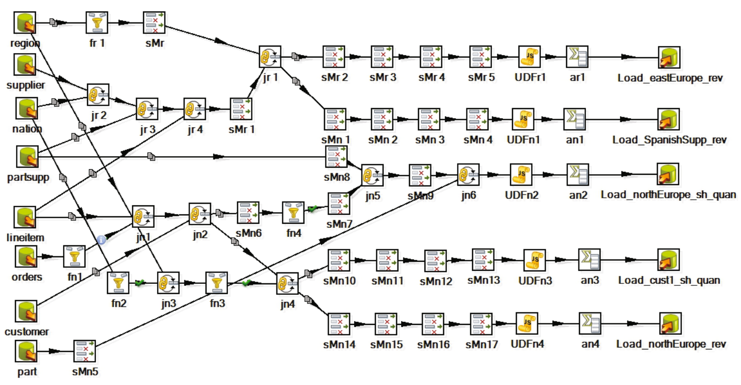


Fig. 7. Physical ETL design satisfying (IR1 - IR5)

encoding to import/export ETL metadata. Thus, *GEM* may connect to any of them, assuming that the correct XSLT to tool-specific XML parser is provided.

Experimental Methodology. We constructed designs based on the TPC-H [2] schema and queries (information requirements). We first used *GEM* to build designs corresponding to individual requirements and then, we launched *CoAl* to consolidate these designs. We considered all order permutations of the provided designs. Here, due to space considerations, we present our representative results for six TPC-H queries: Q3, Q5, Q7, Q8, Q9, and Q10. For each permutation, we first started by consolidating two requirements, and then we incrementally added the other four.

Scrutinizing *CoAl*. Next we report on our experimental findings.
Search Space. As shown in Figure 8(right), for a naive search, the search space grows with the ♯requirements. For input designs of an average size of 28 operations, the number of states considered starts from 1.2k for 2 requirements and go up to 9.7k for 6 requirements. At the same time, the time needed to complete the search grows exponentially with the ♯requirements –see Figure 8(left)– starting from 60sec for 2 requirements and go up to 7.7ksec for 6 requirements. Hence, it is obvious that we need to prune the state space.

For that, we used the (R_1) rule (see Section 4). For evaluating the effectiveness of (R_1), we performed the same set of experiments with and without the rule. This is shown in Figure 8: the red bars represent the naive search and the blue bar shows the results of applying the (R_1) rule into our search. The improvement is obvious in terms of both space and time.

As another experiment, we studied the behavior of the internal characteristics of *CoAl*. Figure 9 shows how ♯matches, ♯maxTransitions (this relates to the (I_2) invariant), ♯firstMatches (R_1-effect), and ♯solutions are affected by the size of the problem. While the number of matches increases with the number of requirements, both the numbers of solutions and reorganizations drop as we encounter additional requirements. At first, *CoAl* aggressively matches different designs, but as the incrementally integrated design matures and the design space is covered, there are lesser novel, valid moves. This is also verified by the ♯visited states (not shown in the graph) that increases with the ♯requirements.

Quality of our solutions. Figure 10 presents our findings regarding the quality of solutions provided by *CoAl* with respect to optimal designs, which were

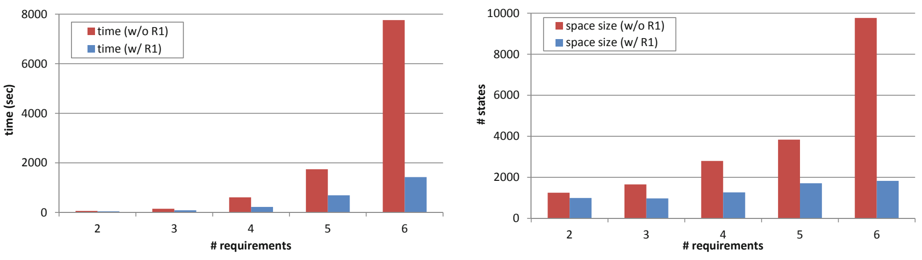


Fig. 8. Search space exploration

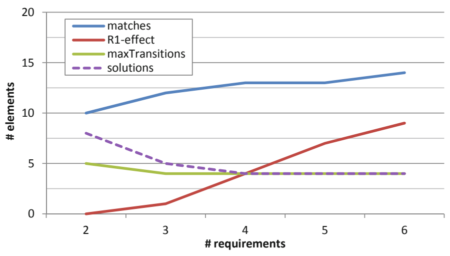


Fig. 9. *CoAl* characteristics

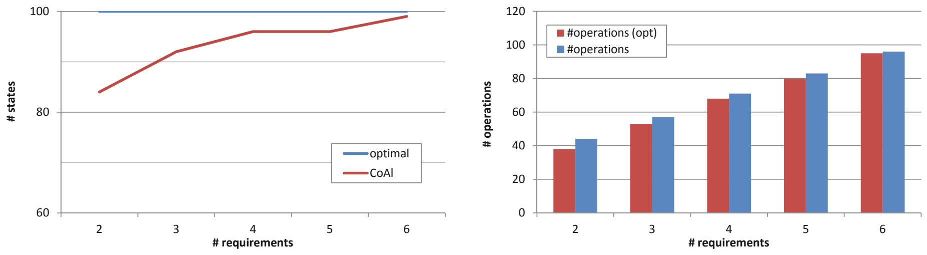


Fig. 10. Quality of *CoAl*

manually constructed. Figure 10(left) shows a comparison based on a combination of design metrics that measure the coverage of the optimal cases by the respective designs. Interestingly, the quality of *CoAl* increases with the ♯requirements. Figure 10(right) reports on an individual metric, namely ♯operations. In all cases, the ♯operations in designs produced by *CoAl*, follows the same pattern as in the respective optimal cases. Moreover, *CoAl* matches all data stores (not shown in the figure). It is worth noting that the time needed for finding the optimal case was 3-4x larger than the respective time needed for getting the designs automatically. In addition, *CoAl* produces equivalent designs; i.e., designs that when executed produce the exact same results.

6 Related Work

ETL. Previous work on ETL has studied modeling and optimization issues. Regarding modeling, there are two directions: the use of ad-hoc formalisms (e.g., [18]) and standard modeling languages (e.g., [3,11,12]). These approaches do not describe how the ETL design adapts to change of requirements. Past work has also tackled the problem of optimizing ETL designs for a variety of objectives (e.g., performance, fault-tolerance, etc.) without showing how to deal with business requirements [15,17].

Query optimization. Both traditional query optimization [8] and multi-query optimization approaches [9] focus on performance and consider a different subset of operations than those typically encountered in ETL. Also, database optimizers do not work well for operations with 'black-box' semantics [17]. Our equivalence rules, however, are based on transitions that have been proved to work for a wider range of operations [15](e.g., arbitrary user functions, data mining transformations, cleansing operations, etc.).

Data mappings and data exchange. Data mapping specifications aim at bridging the heterogeneities between source and target schemas by *mapping* the relationships between schemas [10]. The data exchange problem aims at restructuring data structured under one source schema in terms of a given target schema [7]. However, current algorithms and tools generating automatic data mappings (e.g., [5,6,13]) either cannot tackle grouping and aggregation or overlook complex transformations like those with black-box semantics.

7 Conclusions

We have presented *CoAl*, our approach to facilitate the incremental consolidation of ETL designs based on business requirements. *CoAl* identifies different possibilities for consolidation and suggests near-optimal designs taking into account their processing cost too. Our method can be used either at the early stages of an ETL project for creating the ETL design or at later stages, to facilitate the burdensome process of adapting an ETL design to evolving requirements. *CoAl* is integrated in our prototype tool called *GEM*, which connects to an ETL engine for the actual execution of the produced designs. Our experiments show that *CoAl* successfully automates the design process, a task that is largely infeasible to be performed manually in a timely fashion.

Our future plans include the optimization of our method by exploiting heuristics based on the observation of past execution results for a variety of designs.

References

1. Pentaho Data Integration, `http://kettle.pentaho.com/`
2. TPC-H, `http://www.tpc.org/tpch/spec/tpch2.14.0.pdf`
3. Akkaoui, Z.E., Zimányi, E.: Defining ETL worfklows using BPMN and BPEL. In: DOLAP, pp. 41–48 (2009)
4. Cohen, S., Nutt, W., Sagiv, Y.: Containment of Aggregate Queries. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 111–125. Springer, Heidelberg (2002)
5. Dessloch, S., Hernández, M.A., Wisnesky, R., Radwan, A., Zhou, J.: Orchid: Integrating schema mapping and etl. In: ICDE, pp. 1307–1316. IEEE (2008)
6. Fagin, R., Haas, L.M., Hernández, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema Mapping Creation and Data Exchange. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 198–236. Springer, Heidelberg (2009)
7. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 207–224. Springer, Heidelberg (2002)

8. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database systems - the complete book, 2nd edn. Pearson Education (2009)
9. Kalnis, P., Papadias, D.: Multi-query optimization for on-line analytical processing. Inf. Syst. 28(5), 457–473 (2003)
10. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS, pp. 233–246. ACM (2002)
11. Luján-Mora, S., Vassiliadis, P., Trujillo, J.: Data Mapping Diagrams for Data Warehouse Design with UML. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 191–204. Springer, Heidelberg (2004)
12. Mazón, J.N., Trujillo, J.: An MDA Approach for the Development of Data Warehouses. In: DSS, pp. 41–58 (2008)
13. Mecca, G., Papotti, P., Raunich, S.: Core schema mappings. In: SIGMOD Conference, pp. 655–668. ACM (2009)
14. Romero, O., Simitsis, A., Abelló, A.: *GEM*: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 80–95. Springer, Heidelberg (2011)
15. Simitsis, A., Vassiliadis, P., Sellis, T.K.: Optimizing etl processes in data warehouses. In: ICDE, pp. 564–575 (2005)
16. Simitsis, A., Wilkinson, K., Castellanos, M., Dayal, U.: QoX-driven ETL design: reducing the cost of ETL consulting engagements. In: SIGMOD, pp. 953–960 (2009)
17. Simitsis, A., Wilkinson, K., Dayal, U., Castellanos, M.: Optimizing ETL workflows for fault-tolerance. In: ICDE, pp. 385–396 (2010)
18. Vassiliadis, P., Simitsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. In: DOLAP, pp. 14–21 (2002)
19. Wilkinson, K., Simitsis, A., Castellanos, M., Dayal, U.: Leveraging Business Process Models for ETL Design. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 15–30. Springer, Heidelberg (2010)
20. Yan, W.P., Larson, P.Å.: Performing group-by before join. In: ICDE, pp. 89–100 (1994)

Automatic Transformation of Multi-dimensional Web Tables into Data Cubes

Norah Alrayes and Wo-Shun Luk

School of Computing Science, Simon Fraser University, BC, Canada
{naa27,woshun}@sfu.ca

Abstract. The similarities between data cubes and multi-dimensional tables have long been noted. Routinely, OLAP reporting tools produce multi-dimensional tables from data cubes. In this paper, we develop a scheme that does the reverse transformation, automatically, so that one may produce charts directly from multi-dimensional tables using standard OLAP data visualization tools. In the process, we develop several new techniques for table processing: (i) extraction of non-overlapping hierarchies from a table; (ii) extraction of metadata from the table title via natural language processing; and (iii) integration of tables in a table series, and integration of tables with common dimensions. Experiments were conducted on some 800 summary tables from Statistics Canada, and our success rate was greater than 90% for each component that was tested.

Keywords: Data Cube, Information Extraction, Multi-dimensional Tables.

1 Introduction

A data cube is a multi-dimensional database, where the members of each dimension form a hierarchy. Each cell in the cube is labeled by a member from each hierarchy. Inside a non-empty cell are a number of numeric values, which are called *measures*. These measures may, or may not, be arranged as a hierarchy. Data cube provides a convenient data model which allows the user to access data by naming the member of each dimension hierarchy, and to roll-up/drill-down on specific dimensions by moving up to the ancestors/descendants of the current members. Through the slice-and-dice operation, a data cube can shrink into another equivalent data cube with fewer dimensions, by simply 'aggregating away' some selective dimensions. This is usually called the OLAP (On-line Analytic Processing), or dimensional, approach to data analysis.

Early on, the similarities between a data cube and a statistical table have been recognized [1], by showing how the data and headers of a statistical table may be transformed into a data cube. In [2], a table is defined such that table headers are partitioned into a number of groups, or dimensions as viewed in the OLAP terminology. Obviously, not all tables are suitable for modeling as a data cube. Many simple tables are straightly one-dimensional, in the sense that all column names are members of a single hierarchy. As a result we called tables with more than one dimension *multi-dimensional* tables. Multi-dimensional tables tend to be high quality data tables, published on the Web not only by statistical agencies, but also by public organizations such as universities, e.g., enrolment reports. In fact, reports generated by OLAP tools are precisely multi-dimensional tables. As well, the de facto standard query language on data cubes is called MDX, or multi-dimensional expression.

In this paper, we develop a scheme that automatically transforms a multi-dimensional table into a data cube, complete with its schema. The derived schema must satisfy some properties in order to achieve our objectives. Our basic objective is to enable users to query the table by a well-established query language, or to visualize the data in some chart. Thus, the schema must satisfy the rule of summarizability[3], which is a design principle for data cube. To us, this rule means that a dimension hierarchy must be non-overlapping, i.e., a header in the dimension hierarchy must not have more than one parent header. At a more advanced level, by transforming a table into a data cube, we hope to present a new way of integrating multiple tables into one or integrating tables with existing structured databases. To accomplish table integration, we need not only the names of the dimensions as they are specified in the table, but also the domains associated with them.

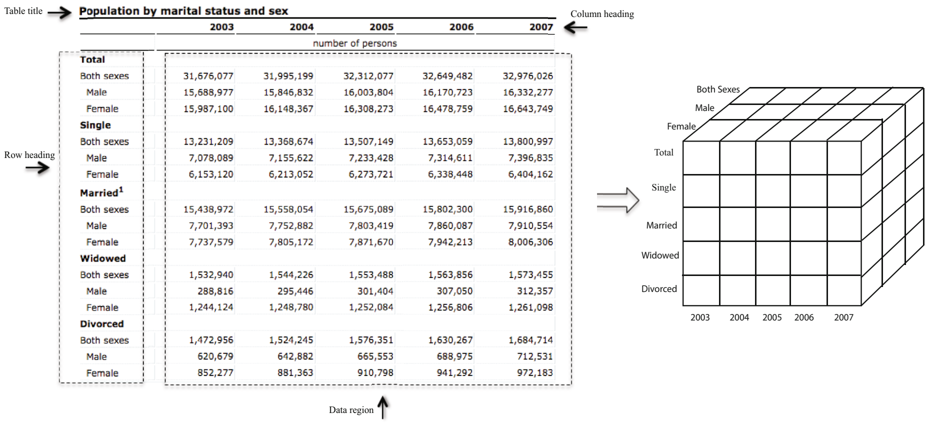
There has been considerable research on extracting semantic information from multi-dimensional tables, e.g., [4], [5], [6], [7], [8]. Many propose to store the semantic information in a proprietary format, while others propose to store in a standard format, i.e., RDF, associated with Semantic Web. None of them resort to the table title for extraction of metadata. While the Semantic Web as a technology is getting more attention, OLAP technology has been around much longer. All of these papers are concerned with extracting from a single table. Issues about table integration are not addressed.

The rest of this paper is organized as follows. In Section 2, a table model is proposed, with specifications of various table components. Section 3 is about how a table is automatically transformed into a data cube, with emphasis on three issues: constructing non-overlapping hierarchies, locating a name for a dimension, and table integration. Section 4 presents our experimental results. Section 5 is the conclusion and future work.

2 Table to Data Cube

Our model of multi-dimensional table consists of four parts: table title, column heading, row heading and data region, as shown in the example in Fig. 1, where the table is divided into these four regions, each indicated by an arrow. A *table title* summarizes the information stored in the table. It usually consists of two components: (a) description of the data stored in the cells of the table, and (b) the names of the dimensions. Usually these elements are connected together with a preposition such as ‘by’, or punctuation ‘,’, as shown as an example in Fig. 1. Most of the names for dimensions are present in the table title, although some common ones, such as years, are often omitted from there. A multi-dimensional table has two kinds of headings: the *row heading* and the *column heading*. The text string contained in a cell in either heading is called a *header*. For large tables, these headings are not symmetric, due to the normal page layout of a web page. Most viewers are more used to scrolling vertically on a web page than scrolling horizontally. As a result, when there are more than three dimensions on the table, dimensions with fewer members and shorter headers, e.g., time dimension, are often located in the column heading. Information about the data, e.g., measurement units, is also often found in the column heading too. The *data region* is populated with data cells that contain numeric data items. A data cell is indexed horizontally by a row header and vertically by a column header. For example the numeric data '6,153,120' is directly associated with the column header '2003' and a row header 'Female'. An experienced human reader, however, will recognize that an additional header 'Single' should also be included because the row comes under the header 'Single' which is visually and semantically distinct from the row header 'Female', especially given that there is a hint of the marital status as a dimension in the table titles.

The goal of this research is to develop a scheme to transform a table like the one on the left in Fig. 1 to an equivalent cube like the one on the right.



Population by marital status and sex

	2003	2004	2005	2006	2007
	number of persons				
Total					
Both sexes	31,676,077	31,995,199	32,312,077	32,649,482	32,976,026
Male	15,688,977	15,846,832	16,003,804	16,170,723	16,332,277
Female	15,987,100	16,148,367	16,308,273	16,478,759	16,643,749
Single					
Both sexes	13,231,209	13,368,674	13,507,149	13,653,059	13,800,997
Male	7,078,089	7,155,622	7,233,428	7,314,611	7,396,835
Female	6,153,120	6,213,052	6,273,721	6,338,448	6,404,162
Married[1]					
Both sexes	15,438,972	15,558,054	15,675,089	15,802,300	15,916,860
Male	7,701,393	7,752,882	7,803,419	7,860,087	7,910,554
Female	7,737,579	7,805,172	7,871,670	7,942,213	8,006,306
Widowed					
Both sexes	1,532,940	1,544,226	1,553,488	1,563,856	1,573,455
Male	288,816	295,446	301,404	307,050	312,357
Female	1,244,124	1,248,780	1,252,084	1,256,806	1,261,098
Divorced					
Both sexes	1,472,956	1,524,245	1,576,351	1,630,267	1,684,714
Male	620,679	642,882	665,553	688,975	712,531
Female	852,277	881,363	910,798	941,292	972,183

Fig. 1. Transformation of a Table to a Data Cube

3 Table Processing

Multi-dimensional tables published by large organizations, especially national agencies, are often meticulously prepared, with consistent naming conventions and visual format, such that a human can easily extract the information in a multi-dimensional way. However, it is still a non-trivial task for a program to do the same. In [9], an algorithm is developed which is capable of partitioning the headers of a multi-dimensional table into multiple groups, so that the headers that belong to a dimension are grouped together. It largely relies on the visual clues installed in the table by the table designer for the benefit of the human reader. While a program can be trained to see what a human can see, the visual clues alone will not be sufficient to extract the metadata from the table required to build a schema for the data cube. In this section, instead of going over the entire table transformation scheme, we will focus on three aspects as highlights of our scheme: constructing non-overlapping hierarchies, assigning labels to dimensions, and table integration.

3.1 Constructing Non-overlapping Hierarchies

To see why we insist on non-overlapping hierarchies, consider the following table about the student majors of the Faculty of Applied Sciences:

Table 1. Enrolment, Faculty of Applied Science

Student Majors	**2008**	**2009**
CS Majors	45	52
ENG Majors	25	27

The design of the dimension 'Student Majors' on the first column is ambiguous, considering the fact that there is a likelihood of joint majors. As a result, it is not possible to compute the enrolment from the entire faculty for each year. This table does not have summarizability on the dimension 'Student Majors', according to [3]. The proper design of dimension will include all categories such that each student belongs to one, and only one, category. Clearly, an experienced table designer would not make this mistake. However, there are three situations when our system may run into overlapping hierarchies:

1. **A hierarchy presented in the table is an integration of two or more hierarchies.** This is likely for any data cube other than 2-dimensional ones because the table designer must arrange all necessary headers in two hierarchies, one in either column or row heading. In the table in Fig. 1, two dimension hierarchies, i.e., Both Sexes and Marital Status, are squeezed into the row heading. The resulting hierarchy will have the same member at the leaf level, e.g., male and female.
2. **Not all members of a hierarchy are presented in the hierarchy on table.** Sometimes, the table designer would like to highlight only a subset of members of a hierarchy. Consider a segment of the table on educational attainment in Fig. 2 from US Census data [10], which consists of column heading plus one associated row in the data region (the segment is folded because it does not fit into the width of the paper). The row heading, which is not shown here, consists of headers for the dimension 'Detailed Years of School'. The second part of this column heading depicts an incomplete race hierarchy. According to a table on race in the US, which does satisfy the summarizability rule, Non-Hispanic white is a descendant member of White. Asian has a descendant member Hispanic Asian, which is also a descendant member of Hispanic (of any race). Presumably, it is the intention of the table designer to highlight a selective set of races, but it would be wrong to conclude from the table that there are only five members in the race.

All Races	**Males**	**Females**	**25 to 34 years old**	**35 to 54 years old**	**55 years old and over**
201,543	97,220	104,323	41,584	83,796	76,163

White	**Non-Hispanic White**	**Black**	**Asian**	**Hispanic (of any race)**
163,979	139,146	23,364	9,723	26,672

Fig. 2. A Segment of a Table – Column Heading and Associated Cells

3. **The multi-dimensional table is a composite table.** Some tables are very similar, as they share exactly the same, say, row (or column) heading. For the sake of comparison, the table designer may decide to integrate these tables so that all column headers for these tables are included in the same row heading. Consider the column heading in Fig. 2 again. The column headers may be partitioned into three groups, by gender, by age group, and by race. One can look upon the table as an amalgamation of three separate tables, each with the same row heading but a different group of column headers. Note that these dimensions do not add to the dimensionality of the data cube, unlike the case for the table in Fig. 1. There, every numeric value inside a cell of the cube is indexed by a header in all three dimensions. Here, it is indexed by only two headers, i.e., a header from a dimension in the row heading, which is 'Detailed Years of School', and a member from one of three tables. There is nothing about the schooling of the male population in the age group of 55 years old and over. This is a form of table integration: integration of 3 tables together as a *composite* table. A human can easily understand, but there are no visual clues for the program, unless the program adds up the associated numeric values. For example, 97,220 + 104,323 = 201,543, which means 'All Races' is a parent header of headers 'Males' and 'Females'. The same is true with age-related headers. More about this summarizability test later.

For the rest of this sub-section, we consider how to construct a non-overlapping hierarchy out of potentially overlapping ones. Initially, there are two hierarchies, one from each heading. They will be derived by observing the visual presentations, e.g., font style, font size, and rows with or without empty cells. These are visual clues installed by the table designer for the benefit of the human reader. For brevity, we will skip the procedure to extract the hierarchy from the headings. We use the technique of dimension factoring to the overlapping hierarchies arising from the first case, and the summarizability test for those from the remaining cases.

Dimension Factoring

We will first present a common way of integrating two hierarchies into one, and then we show how to separate them.

Consider the hierarchies in Fig. 3 where Hierarchies A and B are independent hierarchies. A common way to integrate them is to attach one of the hierarchies as a descendant hierarchy to each leaf node of the other. This is how the Hierarchy X is created. This is also how the row heading of the table in Fig. 1 is constructed. There are in total 15 entries, judging by the data region in the 2nd column from the left. The corresponding headers can be seen as the Cartesian product of these two sets {Both sexes, Female, Male} and {Total, Single, Married, Widowed, Divorced}. Mathematically, they are orthogonal to each. Consequently, the product should be factored into two independent dimensions, and each of the 15 entries may be represented by 2 headers, one from each set.

We now present a procedure to detect a non-overlapping hierarchy. It is slightly more general than simply factoring out the nodes which should have been part of another separate hierarchy. We first examine the leaf nodes (header) of the given hierarchy. If at least two headers are identical, then it is an overlapping hierarchy. We then search for a set of nodes which has the largest set of descendants that are common to all of them. The splitting of the hierarchy will take place along these nodes, which become the leaf nodes of the reduced hierarchy; their descendants will become a separate hierarchy.

As an example, consider the row heading of the table in Fig. 3. Initially, the hierarchy contains many identical headers in the leaf node. During the processing of hierarchy, we identify the headers ‘Single’, ‘Married’, etc. which have identical nodes as descendants, i.e., {Total, Female, Male}. Consequently, the descendants are split off to become a separate hierarchy, and headers such as ‘Single’ and ‘Married’ become of the leaf nodes of the original hierarchy.

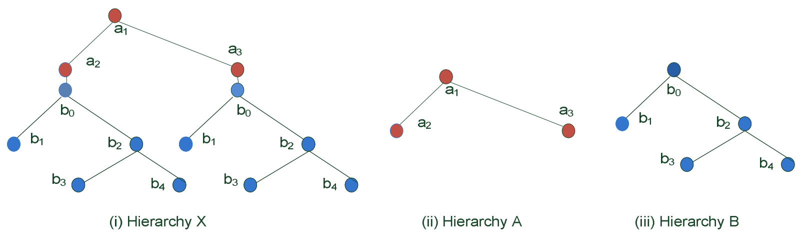


Fig. 3. Composition of Hierarchy A and B

Summarizability Test and Table Decomposition

To detect hierarchies that do not satisfy the summarizability rule as well as disintegrate a composite table, we introduce a *summarizability test*. Given a set of *p* headers of a certain dimension hierarchy at the same level (in appearance), say H={h_1, …, h_p}, we apply the test to H to determine if there is any header that is actually a parent of other headers in H. Suppose we want to know if h_1 is a parent of h_2 and h_3. This is true if:

$$\text{Measure}(h_1, c_2, \ldots, c_n) = \text{Measure}(h_2, c_2, \ldots, c_n) + \text{Measure}(h_3, c_2, \ldots, c_n),$$

where Measure is the numeric value inside the cell identified by the coordinate, and c_j, $2<=j<=n$ is any header in the j^{th} dimension. The header h_1 is then called the *dimensional summary header* of h_2 and h_3. The summarizability test is to identify every dimension summary header in H, and its descendant headers in H. The test procedure is exhaustive, but quite straightforward. One should sort the numbers in M and, for each number in a decreasing order, calculate all possible sums of all numbers below.

With this test result, we will be able to decide how to decompose this composite table. If there is only a summary header for the remaining headers, no disintegration is warranted for this dimension, since the hierarchy is already a non-overlapping one. Otherwise, the data cube will be sliced into different sub-cubes, along the dimension. Each sub-cube will have a summary header together with its constituent members on that dimension. For example, there will be three sub-cubes as the result of this spin-off exercise. The headers for the dimension in each sub-cube will be: {All Races, Male, Female}, {All Races, 25 to 40 years old,…}, and {White, Non-Hispanic White,…}. The sub-cube with the last set of headers should be marked as one that does not satisfy the summarizability rule. We should perform this test for all headers in every dimension of the data cube so that finally each sub-cube has non-overlapping hierarchy.

Note that the summarizability test is not applicable when the aggregate function is not summation, but this situation rarely occurs.

3.2 Dimension Labeling

Once the headers have been partitioned into a number of dimensions, the next step is to annotate each dimension with a label, which will properly describe the collection of headers therein. More importantly, the label should uniquely identify a domain, such that dimensions from different data cubes with the same label share the same domain. As a consequence, the data cubes that have some common dimension name are integratable. We will return to this table integration issue in the next section. Since our labeling process depends on the table title, the table in question must be intact. In particular, a composite table must not be split until after this labeling process, in order that the table title properly describes the table.

To start off, we will attempt to identify all dimensions containing time-related headers, as this is the most common dimension by far. For the rest of the dimensions, we will solve an optimization problem: given a set of headers $H = \{h_1, \dots, h_m\}$, we produce a set of candidate labels, i.e., $C = \{c_1, \dots, c_p\}$, based on H, and then search for a c_i, $1<=i<=p$, which is the most appropriate label for H. That is, each label in C is given a score measuring the semantic distance between the label and H as a whole. The label that is closest to H will be the choice.

The candidate set is formed with labels from the two main sources: the table title, and the dimension summary header for H. The latter is located right above the headers in either column or row headings. It is distinguished from the headers by font type/size and/or indentation. The dimension summary header is ignored when it is an aggregate word only, such as total. Neither the table title nor the dimension summary header is likely to be a sentence. Many of them resemble noun phrases. Therefore, we need to partition them to appropriate sizes in order to find a label to be matched against the headers in H. Chunking any sentence/phrase, in our case the table title and the dimension summary header, depends on the part of speech tagging of words in the sentence. We refer to "Statistical parsing of English sentences" [11] to divide the candidates into appropriate noun phrases, verb phrases and words. For example, for the table title in Fig. 1, the chunking process will result in 3 chunks: population, marital status, sex. We may use these chunks as candidate labels to be included in C. Due to the restriction of WordNet, we include in C all single WordNet synset from the chunks. C will include the following labels: population, marital, status, sex. Note that if any candidate label is chosen as the optimal choice, it is the chunk containing the candidate label that is to be presented. For example if 'marital' is chosen as the optimal choice, the dimension label will be 'marital status'.

To the matching process between C and H, we take an indirect approach. The headers in H could be phrases or short sentences and there could be many of them. Consequently, they could also be too diverse semantically. When the headers are phrases or short sentences, we do not know which word of the header is more dominant. Therefore, we do not think it is appropriate to match the headers directly to the candidate set. Instead, we identify a list of words, A = {a_1,…, a_p} from the WordNet taxonomy that are ancestors to some subsets of headers in H. It is hoped that this pre-processing will result in a smaller set than H, and a reduced level of semantic diversity among the members in A. Thus A becomes a proxy for H, and every label in C will match against A instead of H. We have conducted some experiments on a small number of tables, and our results justify our indirect approach. By introducing pre-processing, the precision is increased from 60% to 90%.

To proceed with this pre-processing, we first represent each header in the dimension within the WordNet taxonomy. It is reasonable to assume that headers in the same dimension should share a common ancestor. Two important factors must be considered when choosing a common ancestor for the dimension headers: the size of the coverage (*c*), where *c* is the set of headers in the dimension that retrieves this ancestor, and height (*h*), which is the median distance between the common ancestors of the headers in *c*. The greater the height, the more tenuous is the ancestor-descendant relationship. Conversely, the larger the coverage, the more relevant is the ancestor to the header set as a whole. For each ancestor matching at least 25% of the headers in the set, we define the common ancestor score to be $|c|/h$. The list of ancestors thus derived is sorted according to their common ancestor score; the low scoring ancestors will be dropped from the list. The remaining ancestors form the set of ancestors A, as the proxy for H.

In practice, we need to adjust the above procedures slightly to cope with the restrictions imposed by WordNet. Because some headers in the dimension are phrases or short sentences, it can be difficult to collect the common ancestor for these headers, since we cannot be sure which word is most closely related to the other headers in the set. WordNet contains single words and some general purpose/common phrases. Since the headers are mostly phrases defined by the table designer and they are not common or general purpose WordNet synset, for WordNet to retrieve them we need to divide each dimension header to the appropriate WordNet synset. Also some ancestors may be retrieved more than once for the same group, for the reason given above. Therefore, we choose the shortest height between them.

We now proceed with the matching between the ancestors in A and the candidate labels in C. This is done by making use of the semantic score presented in "WordNet-based semantic similarity measurement" [12] [13]. We compute the semantic similarities between the ancestors in A and the candidate set, beginning with the highest scoring ancestor and the candidates that appear most of the time after the prepositions in the table title. To measure the score, each candidate is partitioned to a single WordNet synset. The score of the candidate is assigned as the maximum similarity score of any WordNet synset belonging to the candidate and the ancestor. The candidate that matches one of the ancestors and has the highest score is selected as a label.

When the candidates are single WordNet synsets, we measure the semantic similarity between each ancestor and each candidate according to Wu and Palmer [14], as shown in (1), where *LCA* is the least common ancestor depth in WordNet taxonomy between two WordNet synsets, i.e., the ancestor and the candidate, d_{a_i} is the depth for the ancestor, and d_{w_i} is the depth for the candidate word.

$$Sim = LCA \,/\, (d_{a_i} + d_{w_i}) \tag{1}$$

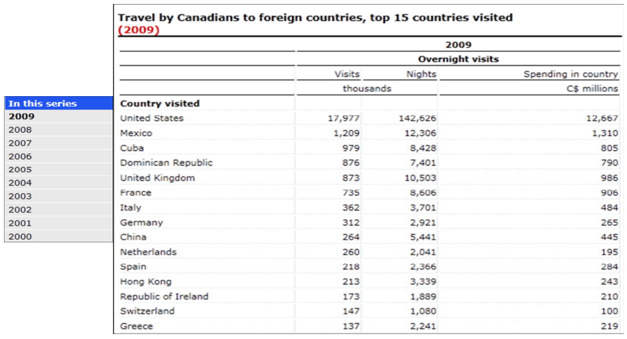
3.3 Table Integration

Here we address two common and practical ways for effective integration of tables: integrating tables in a table series, and domain integration.

Table Series

It is a common practice to organize tables in a table series, where the tables contain table links to other related table(s). (In fact, EXCEL has the same feature). These tables are derived from a larger table, which presumably contains too much information to be properly displayed as one single web table. Normally, in recognition that a multi-dimensional table is transformable into a multi-dimensional array, each table is a slice of the cube across one selected dimension. Consequently, these tables share roughly the same table title, column heading, and row heading; however data inside the data cells are different. Each table is accessible by moving to the associated header of that dimension in a drop-down box, or equivalently clicking the associated button, as shown in the table in Fig. 4.

Integration of tables in a table series is straightforward. It is just the reverse of the process of splitting a large table into a number of tables across a selected dimension. Each table in the series will be processed in exactly the same way, though care must be taken to remove the references in the table to the associated members in that dimension, e.g., the string ‘2009’ under the title in Fig. 4, because there is another reference to this year in the column heading.



In this series
2009
2008
2007
2006
2005
2004
2003
2002
2001
2000

Travel by Canadians to foreign countries, top 15 countries visited
(2009)

	2009		
	Overnight visits		
	Visits	Nights	Spending in country
	thousands		C$ millions
Country visited			
United States	17,977	142,626	12,667
Mexico	1,209	12,306	1,310
Cuba	979	8,428	805
Dominican Republic	876	7,401	790
United Kingdom	873	10,503	986
France	735	8,606	906
Italy	362	3,701	484
Germany	312	2,921	265
China	264	5,441	445
Netherlands	260	2,041	195
Spain	218	2,366	284
Hong Kong	213	3,339	243
Republic of Ireland	173	1,889	210
Switzerland	147	1,080	100
Greece	137	2,241	219

Fig. 4. A Table in a Table Series

Domain Integration

Sometimes, tables that are not tightly coupled together, as in a table series, may still be integrated if they share some common dimensions in their schema. A case in point is the composite table in Fig 2, which is formed by integrating three tables that have the identical dimension, ‘Detailed Years of School’. It is not enough to claim that two dimensions are identical just because they have the same name. Conversely, dimensions with different names may turn out to be identical. To us, two dimensions are the same if they share the same domain.

A *domain* consists of a set of values from which a dimension draws its members. In this sense, a domain of a dimension is similar to a domain for an attribute in the relational model. Domain integration is about identifying dimensions that share the same domain. Domain integration will establish the join connectivity of cubes with common domains. Domain integration plays an important part in table processing too. For example, suppose a member of a dimension has an abbreviation, e.g., N.B., and a member of another dimension is New Brunswick. If it is found that the two dimensions share the same domain, the former member will be reverted back into its full name, since the latter is syntactically closest to the abbreviation.

To achieve domain integration, a list of domains and related information is kept during the table processing to keep track of the domains that have been processed. Whenever a dimension is derived (with or without a name), an attempt is made to match the dimension with one of the dimensions on the list. If a match occurs, the union of the members of the two dimensions replaces the members of the dimension already on the list. If either of the matched dimensions has been labeled, the other one will be given the same label.

4 Experimental Results

Our system was tested against the summary tables in the Statistics Canada website[15]. They are freely accessible, and belong to one of the few national statistical agencies that continue to publish in HTML, although they are available in PDF, and recently in Excel spreadsheet. More importantly, these tables are quite different from all other non-summary tables because they cover a wide range of topics and often pack together data from a multitude of non-summary tables. Consequently, they are more appropriate for our extractor for testing purposes than the non-summary tables published by government agencies including Statistics Canada. Our test dataset contains some 800 randomly selected tables. They are domain-independent and cover such topics as education, construction, household, travel, etc. To measure the results for each process proposed in our paper, we count the total number of dimensions in the tables, the number of tables that need to be integrated, and the total number of domains that need to be extracted.

Table 2. Experimental Results

Process name	**Number of components**	**Success rate**
Deriving the dimension and assigning the dimension label	2446	91%
Table integration	119	92%
Domain integration	50	96%

To confirm that our method works for different government agencies that have HTML multidimensional tables (in English), we tested our algorithms in a small number of tables in English from Statistics Austria and Statistics Finland, and retrieved good results.

5 Conclusion and Future Work

We have presented a scheme to allow one to view and manipulate a multi-dimensional table as if it were a data cube for querying and data visualization. In addition, one can integrate tables that share common dimensions and treat them as one big virtual cube. We have also made contributions to the state of the art of table processing. We introduce the summarizability test, which will allow a program to discover child-parent relationships among a group of headers, in the absence of any visual clues. We develop a technique to analyze a table title as a natural language entity and extract vital metadata about the table.

Currently, we are in the process of expanding the coverage of the tables our system can analyze. Many, if not most, tables are not published in the HTML form. They are either part of a PDF document or, increasingly, are published individually as spreadsheets. We hope to apply our system to large sets of multi-dimensional tables from a variety of sources.

References

1. Shoshani, A.: OLAP and statistical databases: similarities and differences. In: Fifth International Conference on Information and Knowledge Management, Rockville, Maryland (1996)
2. Wang, X.: Tabular abstraction, editing, and formatting, University of Waterloo, Ph.D. Thesis (1996)
3. Lenz, H.J., Shoshani, A.: Summarizability in OLAP and Statistical Data Bases. In: Ninth International Conference on Scientific and Statistical Database Management, Olympia, WA, USA, pp. 132–143 (1997)
4. Embley, D.W., Lopresti, D.P., Nagy, G.: Notes on Contemporary Table Recognition. In: 7th Int. Workshop on Document Analysis Systems, pp. 164–175 (2006)
5. Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., Pollak, B.: Towards Domain-Independent Information Extraction from Web Tables. In: WWW 2007, Banff, Alberta, Canada (2007)
6. Pivk, A.: Automatic ontology generation from web tabular structures. AI Communications 19, 83–85 (2006)
7. Tanaka, M., Ishida, T.: Ontology Extraction from Tables on the Web. In: International Symposium on Applications on Internet, pp. 284–290 (2006)
8. Seth, S., Jandhyala, R., Krishnamoorthy, M., Nagy, G.: Analysis and Taxonomy of Column Header Categories for Web Tables. In: DAS 2010, Boston, MA, USA (2010)
9. Luk, W., Leung, P.: Extraction of Semantics From Web Statistical Tables. In: IEEE/WIC/ACM International Workshop on Semantic Web Mining and Reasoning, Beijing, China (2004)
10. US Census Bureau, Education Attainment (Table 3) (2011), `http://www.census.gov/hhes/socdemo/education/data/cps/2011/tables.html`
11. Northedge, R.: Code Project: Statistical parsing of English sentences (2011), `http://www.codeproject.com/Articles/12109/Statistical-parsing-of-English-sentences`

12. Simpson, T., Dao, T.: WordNet-based semantic similarity measurement, Source code (January 2010), http://wordnetdotnet.googlecode.com/svn/trunk/Projects/Thanh/
13. Simpson, T., Dao, T.: Code Project: WordNet-based semantic similarity measurement (2010), http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx?msg=2776502
14. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics, pp. 133–138 (1994)
15. Statistics Canada, http://www.statcan.gc.ca/start-debut-eng.html

Differentiated Multiple Aggregations in Multidimensional Databases

Ali Hassan[1], Franck Ravat[1], Olivier Teste[2], Ronan Tournier[1], and Gilles Zurfluh[1]

[1] Université Toulouse 1 Capitole
[2] Université Toulouse 3 Paul Sabatier
[1,2] IRIT (UMR 5505), 118 Route de Narbonne, F-31062 Toulouse Cedex 9, France
{hassan,ravat,teste,tournier,zurfluh}@irit.fr

Abstract. Many models have been proposed for multidimensional data warehouse modeling and most consider a same function to determine how measure values are aggregated according to different data detail levels. We provide a conceptual model that supports (1) multiple aggregations, associating to the same measure a different aggregation function according to analysis axes, and (2) differentiated aggregation, allowing specific aggregations at each detail level. Our model is based on a graphical formalism that allows controlling the validity of aggregation functions (distributive, algebraic or holistic). We also show how conceptual modeling can be used, in an R-OLAP environment, for building lattices of pre-computed aggregates.

Keywords: Data warehouse, Conceptual modeling, Aggregate lattice, multiple aggregations, Aggregation functions.

1 Introduction

Multidimensional databases (MDB) are used by decision makers to analyze data within their organizations [7]. To query efficiently data, these systems represent the analyzed data from analysis indicators (i.e. measures grouped into facts) as points in the multidimensional space [4]. Each dimension having various granularity/detail levels. Decision makers visualize extracts of the MDB with two-dimensional "slices" of the cube, i.e. multidimensional tables (MT) [5] and can interact with these extracts with manipulation operations [15].

A classical MBD supports only the calculation of a measure using the same aggregation function while performing drilling or rotating operations (i.e. changing the analyzed slice of the cube). For example, if we consider sales amounts, these can be calculated as the sum of the products sold by cities and years (Fig. 1-a). When drilling from cities to countries, the new amounts are calculated using the same aggregation function (also sum in Fig. 1-b). When the user wishes to change the aggregation function between two slices of the manipulated cube, the classical MBD aggregates all-ready aggregated data and has no way of guaranteeing the validity of the newly aggregated data (for example, averages of averages are usually erroneous).

This paper aims at allowing non-uniform aggregations during user manipulations. To ensure the validity of such aggregations, we define **differentiated multiple**

aggregations. Our proposal aims at developing a multidimensional model flexible enough for designing cubes with aggregation functions according to different levels.

Case Study. A diploma delivery jury. Here, decision makers (jury members) deliver diplomas by analyzing the marks of students. Students are split into groups and the academic year has two semesters. Each semester consists of Teaching Units (TU) and each TU is comprised of several courses. Each course is associated with a coefficient that represents the importance of the course in the TU, which itself is linked to an ECTS (European Credit Transfer System). Each semester has the same amount of ECTS.

Fig. 2-a shows the conceptual star schema [14] of the MDB of our case study. This MDB analyzes the Marks (measure) by Courses and Students (dimensions). A course is characterized by a course number (C_Id), a teaching unit number (TU_Id) and a semester. Each student has a student number (S_Id) and a group number (G_Id).

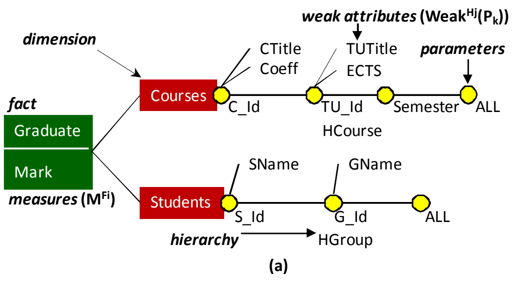
SUM(SALES.Amount)		**DATE**			
		Year	2009	2010	2011
STORE	**City**				
	Toulouse		100	120	115
	Bordeaux		110	100	105
	Barcelona		90	115	100

(a)

SUM(SALES.Amount)		**DATES**			
		Year	2009	2010	2011
STORE	**Country**				
	France		210	220	220
	Spain		90	115	100

(b)

Fig. 1. (a) and (b) Uniform aggregation in slices of a cube



AVG(Graduate.Mark)		**Courses**			
		Semester	S1		
		TU_Id	U1		U2
		C_Id	M1	M2	M3
Students	**S_Id (SName)**				
	St1(Tom)		14	10	12
	St2(Sara)		8	10	9

(b)

Fig. 2. (a) Diploma delivery case study; (b) Average marks by course by students in a MT

Illustration of the Problem. This schema analyzes average marks by courses and by students (Fig. 2-b). Obtaining the average mark by TU in this multidimensional environment requires aggregating the average marks by courses in accordance with the function associated with the measure Mark (AVG). But this operation gives a result that does not correspond to examination modalities: an average mark by TU should be calculated from the course marks and taking into account the coefficient of each course (equation 1). Similarly, for average marks by semester, the ECTS of each TU (equation 2) has to be taken into account.

$$AVG\_TU = \frac{\sum Mark * Coeff}{\sum Coeff} \tag{1}$$

$$AVG\_Semester = \frac{\sum AVG\_TU * ECTS}{\sum ECTS} = \frac{\sum \left(\frac{\sum Mark * Coeff}{\sum Coeff} \right) * ECTS}{\sum ECTS} \tag{2}$$

Therefore, classical approaches that consider a single aggregation function for all modeled aggregation levels in the star schema suffer from several limits:

- **Variability of the aggregation function.** Traditionally, models do not allow the use of aggregation functions that vary along dimensions or hierarchical levels. In our example, the aggregation function changes between the levels C_Id (courses), TU_Id (teaching units) and the semester level.
- **Shortcomings of basic functions.** When aggregating data across hierarchical levels, in our example, we use non-standard aggregation functions which use complementary data other than measure values (i.e. coefficients *Coeff*, weights *ECTS*).
- **Aggregation constraints.** Aggregation functions belong to three different categories [4]: **distributive** functions can calculate aggregated values of the selected granularity level from the values already aggregated at the lower level (e.g. yearly amounts can be calculated by summing monthly values); **Algebraic** functions can calculate aggregated values from stored intermediate results (for example, the average of an amount per year can be calculated from the sum of the amounts and the count of occurrences from a month level); Finally, **holistic** functions cannot be calculated from intermediate results. In this case, aggregated values must be calculated from the elementary values of the lowest granularity level (e.g. RANK).

In addition to these functions categories, constraints on how to make the calculation of distributive or algebraic functions may exist. In our example, the average per semester is calculated from the average per TU as shown in the first expression of (2).

The objective of this paper is to propose a multidimensional model sufficiently expressive to support this type of aggregation.

The rest of this paper is organized as follows: section 2 reviews related work. Section 3 defines our conceptual multidimensional model followed by extensions for differentiated multiple aggregations, then we present the associated graphical formalism. Section 4 shows the logical R-OLAP model of our star schema and its optimization relations. We detail our experiments in section 5 and the last section concludes this work and states some research perspectives.

2 Related Work

There are typically two approaches for modeling multidimensional databases. The first is based on the data cube (or hypercube) metaphor according to which the MDB is represented by cubes. The second is known as multidimensional modeling, where the MDB is described by a star schema or constellation [7]. Our work falls in the second category. A cube is based on an equivocal separation between the structure elements and the values [16]: modeling analysis axes is not very expressive especially due to the difficulty for representing the hierarchical organization of the data. It is also limited for representing constellations of facts with shared dimensions.

Several surveys of the domain [2,17,11] and comparative studies [13,1,10,15,12,8] exist. Most of the existing proposals consider that a measure is associated with only one aggregation function for all aggregation levels. This function calculates the same aggregation for all combinations of all modeled parameters.

The YAM^2 model [1] is the only model that supports a different aggregation function with each dimension. But this model does not allow changing the function within the hierarchical levels. In [13] the authors can link several aggregation functions to a single measure but each one will be used for all dimensions and all hierarchical levels.

Regarding commercial tools, "Business Objects" uses a single aggregation function for each measure. By contrast, "Microsoft Analysis Services" offers the possibility that a "custom rollup" can be applied in a hierarchy in several ways [6]:

- By using unary operators to solve the aggregation problem over a particular type of hierarchy (parent-child attributes hierarchy). These hierarchies are built from a single attribute with a reflexive join relationship on the attribute itself (i.e. a join on the dimension table itself).
- By using MDX scripts, either directly or by using the attribute property "CustomRollupColumn" which indicates a column where MDX scripts are stored.

Although concerning aggregation functions, they are not related to a specific dimension or an aggregation level. They are related to a member (an instance) of an aggregation level in a hierarchy (i.e. a line in the dimension table). Therefore, applying this "custom rollup" to an aggregation level requires repeating it for all the instances of that level with storage problems and reduced performance [6]. Moreover, binding a "custom rollup" with a specific instance can cause difficulties when updating data.

Our aim is to remove this limit by designing a conceptual model for representing differentiated multiple multidimensional aggregates. By *multiple* we mean that the same measure can be aggregated by several aggregation functions according to analysis axes and by *differentiated* we mean that these aggregations may vary, depending on the chosen aggregation level.

In addition to the aggregation functions classification [4] (distributive, algebraic and holistic), there are other classifications:

- From a summerizability point of view, aggregation functions are classified in two groups [1]: (1) "Transitive" that guarantees summerizability, (2) "Non-Transitive" which implies that aggregations must always be calculated from the base level.
- From a measure point of view, aggregation functions are of three types [13]: (1) for additive data, (2) for snapshot data that can be used for average calculations, (3) for constant data, i.e. data that can only be counted.

All these proposals as well as aggregation functions classifications assume that the measure aggregation can be calculated from the base level. Our goal is to add the means to consider the opposite case (when the measure cannot be aggregated from the base level) using *aggregation constraints.*

3 Conceptual Data Model

3.1 Classical Concepts

Let us define $\mathcal{N}$, F and D such as: $\mathcal{N} = \{n_1, n_2, \ldots\}$ a finite set of non-redundant names; $F = \{F_1, \ldots, F_n\}$ is a finite set of facts, $n \geq 1$; and $D = \{D_1, \ldots, D_m\}$ is a finite set of dimensions, $m \geq 2$.

Definition 1. A *fact*, denoted F_i, $\forall i \in [1..n]$, is defined by (n^{Fi}, M^{Fi}), where: $n^{Fi} \in \mathcal{N}$ is the name that identifies the fact; and $M^{Fi} = \{m_1, ..., m_{pi}\}$ is a set of *measures*.

We define the measure set as $M = \bigcup_{i=1}^{n} M^{F_i}$.

Definition 2. A *dimension*, denoted D_i, $\forall i \in [1..m]$, is defined by (n^{Di}, A^{Di}, H^{Di}), where: $n^{Di} \in \mathcal{N}$ is the name that identifies the dimension; $A^{Di} = \{ a_1^{D_i}, ..., a_{r_i}^{D_i} \}$ is the set of the *attributes of the dimension*; and $H^{Di} = \{ H_1^{D_i}, ..., H_{s_i}^{D_i} \}$ is a set of *hierarchies*.

Hierarchies organize the attributes of a dimension, from the finest graduation (root parameter) to the most general graduation (extremity parameter, "All"). Thus a hierarchy defines the valid navigation paths on an analysis axis.

We define the attribute set $A = \bigcup_{i=1}^{m} A^{D_i}$ and the hierarchy set $H = \bigcup_{i=1}^{m} H^{D_i}$.

Definition 3. A *hierarchy*, denoted H_j (abusive notation of $H_j^{D_i}$, $\forall i \in [1..m]$, $\forall j \in [1..s_i]$) is defined by $(n^{Hj}, P^{Hj}, \prec^{Hj}, Weak^{Hj})$, where:

- $n^{Hj} \in \mathcal{N}$ is the name that identifies the hierarchy,
- $P^{Hj} = \{ p_1^{H_j}, ..., p_{q_j}^{H_j} \}$ is a set of attributes called *parameters*, $P^{Hj} \subseteq A^{Di}$,
- $\prec^{Hj} = \{(p^{Hj}_x, p^{Hj}_y) \mid p^{Hj}_x \in P^{Hj} \wedge p^{Hj}_y \in P^{Hj}\}$ is an antisymmetric and transitive binary relation between parameters. Remember that the antisymmetry means that $(p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k2}) \wedge (p^{Hj}_{k2} \prec^{Hj} p^{Hj}_{k1}) \Rightarrow p^{Hj}_{k1} = p^{Hj}_{k2}$ while the transitivity means that $(p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k2}) \wedge (p^{Hj}_{k2} \prec^{Hj} p^{Hj}_{k3}) \Rightarrow p^{Hj}_{k1} \prec^{Hj} p^{Hj}_{k3}$.
- $Weak^{Hj} : P^{Hj} \rightarrow 2^{A^{D_i} \setminus P^{H_j}}$ is an application that associates to each parameter a set of dimension attributes, called *weak attributes* (2^N represents the power set of N).

We define parameter sets $P^{D_i} = \bigcup_{j=1}^{Si} P^{H_j}$ and $P = \bigcup_{i=1}^{m} P^{D_i} = \bigcup_{i=1}^{m} \bigcup_{j=1}^{Si} P^{H_j}$.

Lemma 1. For each dimension D_i, all its attributes are exclusively either parameters or weak attributes, $P^{Di} \cap W^{Di} = \varnothing$ and $P^{Di} \cup W^{Di} = A^{Di}$.

3.2 Extensions for Differentiated Multiple Aggregations

We enrich the multidimensional model by the following extensions:

- **Differentiated aggregation.** The aggregation function is associated with one measure and one parameter. This kind of aggregation allows a specific aggregation over each level of granularity.
- **Multiple aggregations.** This is a simplified representation instead of a repeated use of the same differentiated function over several levels of granularity. The same

aggregation is performed over each level of granularity of a dimension. The function is associated with one measure and a dimension. Therefore, it is important to note that several aggregation functions can be associated to a same measure; one for each analysis axis.

- **General aggregation.** This function is associated only with a measure without taking into account neither parameter nor dimension. This is a simplified representation instead of a repeated use of the same multiple function over several dimensions. This is equivalent to aggregation functions in classical models.
- **Execution order.** it is possible to have different aggregation functions, one for each dimension. These functions are generally not commutative. Therefore, it is necessary to plan in the MDB an execution order for the use of the functions between the different dimensions involved in a same analysis.

Let $\mathcal{F} = \{f_1, f_2, ...\}$ be a finite set of aggregation functions.

Definition 4. A *multidimensional schema*, denoted S, is defined by (F, D, Star, Order, Aggregate), where:

- $F = \{F_1, ..., F_n\}$ is the set of facts, if $|F| = 1$ then the multidimensional schema is called a *star schema* while if $|F| > 1$ it is a *constellation schema*,
- $D = \{D_1, ..., D_m\}$ is the set of dimensions,
- Star: $F \rightarrow 2^D$ is a function that associates each fact to a set of dimensions according to which it can be analyzed (note that 2^D represents the power set of D).
- Order: $M \rightarrow 2^{D \times \mathbb{N}^*}$ is a function that binds to each dimension (regarding each measure) an execution order used for aggregating the measure. The aggregation function of the dimension with the smallest order is the highest priority. If the aggregation functions of two dimensions are commutative, then both dimensions will have the same order.
- Aggregate: $M \rightarrow \mathcal{F} \times 2^D \times 2^{H \times P} \times \mathbb{N}^-$ associates each measure to an aggregation function and a specific aggregation level. Aggregate defines the different types of aggregation functions supported by our model:
  - *General aggregation*: 2^D and $2^{H \times P}$ are not used ($2^D = \varnothing$ and $2^{H \times P} = \varnothing$).
  - *Multiple aggregation*: $2^{H \times P}$ is not used ($2^{H \times P} = \varnothing$). Here, the function is used to aggregate the measure over the entire considered dimension.
  - *Differentiated aggregation*: the function aggregates the measure between a considered parameter and the parameter directly above it in the same hierarchy.

$\mathbb{N}^-$ is to constraint aggregations by indicating a specific level from which the considered aggregation must be calculated. An unconstrained aggregation will be associated with 0 while a constrained aggregation will be associated with a negative value to force the calculation from a chosen level lower than the considered level.

Lemma 2. Aggregation functions ensure the full *coverage* of multidimensional schemas. Thus there does not exist any parameter (i.e. aggregation levels) for which the aggregation function to be applied is unknown.

$$\forall i \in [1..n], \forall m_k \in M^{F_i}, \exists f \in \mathcal{F}, \exists x \in \mathbb{N}^-,$$
$$\begin{cases} (f, \{\ \}, \{\ \}, x) \in Aggregate(m_k) \\ \forall D_j \in Star(F_i) \,|\, (f, \{D_j\}, \{\ \}, x) \in Aggregate(m_k) \\ \forall H_S \in H^{D_j}, \forall p_q \in P^{D_j} \setminus \{All^{D_j}\} \,|\, (f, \{D_j\}, \{(H_s, p_q)\}, x) \in \mathrm{Aggregate}(m_k) \end{cases}$$

Less formally, the *coverage* of the schema is carried out in several ways:

— By using a general aggregation function,
— By using a multiple aggregation function for each dimension,
— By using a differentiated aggregation function for each aggregation level,
— By combining multiple aggregation functions with differentiated ones. Each dimension having no multiple function must have a differentiated function for each aggregation level (i.e. parameter).

3.3 Graphical Formalisms

The following example, illustrated in Fig. 3, is defined formally by (F, D, Star, Order, Aggregate) where:

— $F = \{F_{Graduate}\}$, where the fact is defined by $F_{Graduate}$ = ('*Graduate*', {Mark}).
— $D = \{D_{Courses}, D_{Students}\}$, where the dimensions are defined by:

- $D_{Courses}$ = ('*Courses*', $\{a_{C\_Id}, a_{Coeff}, a_{CTitle}, a_{TU\_Id}, a_{ECTS}, a_{TUTitle}, a_{Semester}, ALL^{DCourses}\}$, $\{H_{HCourse}\}$) with $H_{HCourse}$ = ('*HCourse*', $\{a_{C\_Id}, a_{TU\_Id}, a_{Semester}, ALL^{DCourses}\}$, $\{(a_{C\_Id}, a_{TU\_Id}), (a_{TU\_Id}, a_{Semester}), (a_{Semester}, ALL^{DCourses})\}$, $\{(a_{C\_Id}, \{a_{Coeff}, a_{CTitle}\}), (a_{TU\_Id}, \{a_{ECTS}, a_{TUTitle}\})\}$), and
- $D_{Students}$ is specified similarly.

— Star : $F \to 2^D$ | Star($F_{Graduate}$) = $\{D_{Courses}, D_{Students}\}$
— Order : $M \to 2^{D \times \mathbb{N}^*}$| Order(Mark) = {(Courses, 1), (Students, 2)}
— Aggregate : $M \to \mathcal{F} \times 2^D \times 2^{H \times P} \times \mathbb{N}^-$ |
Aggregate (Mark) = { (AVG(Mark), {Students},{ },0) [1],
(AVG_W[2] (Mark, Coeff), {Courses}, {(HCourse, C_Id)},-1) [3],
(AVG_W(Mark, ECTS), {Courses}, {(HCourse, TU_Id)}, -1),
(AVG(Mark), {Courses}, {(HCourse, Semester)}, -1)}.

We introduce a 2 level graphical formalism to facilitate the understanding of the MDB schema:

— **Structural Schema.** The structural schema is used to display the multidimensional elements (facts, dimensions and hierarchies) hiding aggregation mechanisms. This global view (see Fig. 2) is defined by the function *Star*. The graphical formalism is based on [3,15].

[1] Note that there is no constraint on the aggregation.
[2] AVG_W is the function that computes a weighted average.
[3] The aggregated values are computed from the values at the level directly below the one considered.

— **Aggregation schema.** For each measure $m_k \in M^{Fi}$, an aggregation schema is obtained using the functions *Order* and *Aggregate*. Fig. 3 details the aggregation mechanisms involved in the *Mark* measure analysis (multiple, differentiated and general aggregations, constraints of aggregation and execution order) but shows simply the structural elements directly related to the *Mark* measure.

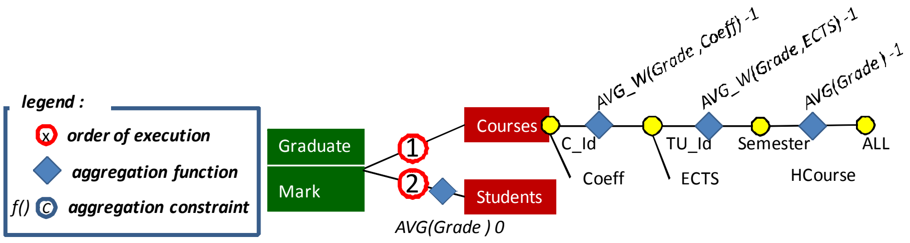


Fig. 3. Graphical notation extensions (Student dimension is not completely displayed)

As illustrated in the above figure, the execution order is symbolized by numbers on the edges that connect the fact to dimensions while the aggregation functions are modeled by diamonds. The positions of the diamonds depend on the type of function:

— A general function is represented by a diamond on the fact (none in our example),
— A multiple aggregation function is on the edge connecting facts to dimensions,
— A differentiated aggregation function is a label on the edge linking two parameters.

4 Relational-OLAP (R-OLAP) Logical Model

Current multidimensional schema implementations use mainly the relational approach [7]. This approach has many advantages such as reusing proven data management mechanisms and the ability to manage very large volumes of data.

4.1 R-OLAP Star

In this relational context, the MDB is translated into relations [7]. Applied to our example, the R-OLAP schema is the following:

```
COURSE    (C_Id,Coeff,CTitle,TU_Id,ECTS,TUTitle,Semester)
STUDENTS  (S_Id,SName,G_Id,GName)
GRADUATE  (C_Id#,S_Id#,Mark)
```

The aggregation functions are stored in the database engine. We use a metaschema (not detailed here due to lack of space) to describe the multidimensional schema (facts, dimensions and hierarchies) corresponding to the R-OLAP relations that store the analysis data. It also describes the different aggregation functions and the possible aggregation constraints.

4.2 Optimized Star

Conceptual modeling allows structuring hierarchically the analysis axis (dimension) graduations (parameters). These hierarchies are exploited for pre-computing the aggregations required by decision makers to navigate and to perform analyses in the multidimensional space (using OLAP). Traditionally, these pre-aggregations are modeled by a *lattice* of pre-computed aggregates [4,2] where:

– each node represents a pre-computed aggregate and
– each edge represents a path for computing aggregates. If the aggregation function used is *distributive* or *algebraic*, the aggregate can be calculated from the directly lower aggregate, while if it is *holistic*, the calculus is from the base relation [4].

The flexibility introduced in the conceptual model impacts the lattice. The differentiated and multiple aggregation functions involve using different aggregation computations for each edge of the lattice (Fig. 4), contrary to the traditional approach which usually considers only a single aggregation function.

When multiple paths are possible, the less costly path is preferred. The cost function (not detailed here) favors the most effective computation time [9]. However, the use of different aggregation functions on each edge of the lattice makes the cost estimate more complex than in usual lattices.

Furthermore, some paths or edges are invalid; therefore, these can be eliminated to reduce the lattice size. This pruning is possible using the execution order. In our example, we cannot apply the function AVG_W(Mark, Coeff) on the *Courses* dimension after AVG(Mark) on the *Students* dimension. Thus, average marks of TU for each group (TU_Id_G_Id) cannot be calculated from the average marks of courses by groups (C_Id_G_Id) as this would give erroneous results. Therefore, the edge between C_Id_G_Id and TU_Id_G_id can be deleted.

Moreover, constraints (the specific level from which the considered aggregation must be calculated) associated with the aggregation functions have repercussions on the lattice. Edges with a symbol (crosses in a circle in Fig. 4) come from these constraints which require calculating the node from another specific node. It is then forbidden to calculate an upper node using transitivity from lower nodes as it would be in a classical schema. Thus the computing paths are blocked as soon as such an edge is encountered; e.g. the node Semester_ALL^{Students} is calculable from the direct lower node Semester_G_Id; using transitivity, it is also calculable from the lower node Semester_S_Id. However, the edge resulting from the constraint of the function AVG_W(Mark, ECTS) which operates on the edge (Semester_S_Id, TU_Id_S_Id) blocks the calculation transitivity.

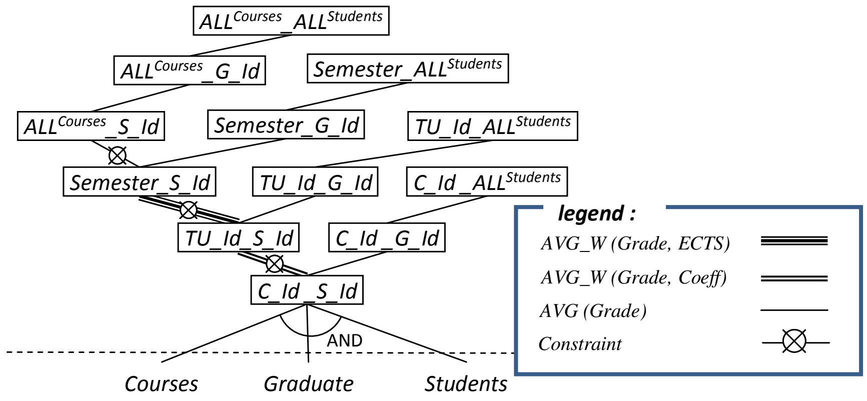


Fig. 4. Optimization lattice

5 Experiments

To demonstrate the feasibility of our approach, we have produced a prototype using Oracle 12*g* DBMS. We have implemented the aggregation functions (1) and (3) described in section 1. For this, a generic aggregation function was implemented:

- An Oracle object type (class) was used to implement the four routines of the interface ODCIAggregate that are ODCIAggregateInitialize, ODCIAggregateIterate, ODCIAggregateMerge and ODCIAggregateTerminate. These methods correspond to internal operations that each aggregation function performs (respectively Initialize, Iterate, Merge and Terminate).
- Then, our aggregation function AVG_W was created to compute a weighted mean. This function takes one parameter (TYPE ty_weighted_data AS OBJECT (value NUMBER, weight NUMBER)) composed of the data to aggregate and the weight.

Our own aggregation function was tested using three SQL queries that simulate user navigation in the multidimensional space. Note that the SQL Queries are generated using an interface where the user manipulates only multidimensional concepts. Thus, the complexity of the logical structure of the MDB is hidden.

Firstly, the decision maker visualizes biannual average marks of students (the SQL query on the R-OLAP schema is Q1 in Fig. 5. Secondly, in order to have a detailed view of students who have failed, the user carries out a drilling operation to get average marks of the student by Teaching Units (Q2 in Fig. 5). Thirdly, to find the courses that students will have to take the exam again, another drilldown views the average marks by courses (Q3 in Fig. 5).

Fig. 6 shows the execution time (ms) according to the number of processed tuples for each of the three queries. The time of the two first queries increase regularly with the number of tuples while the third query is almost stable because the number of tuples was low and relatively constant in our experiments. These first results are intended to show the feasibility of our approach. They are encouraging as we do not observe notable changes in the query execution behavior in a data warehouse either built in a traditional way or on the base or our proposals of differentiated multiple aggregations.

Q1
```
SELECT Semester, S_Id,
  AVG_W(ty_weighed_data (Mark, ECTS)) AS Mark
FROM ( SELECT Semester, ECTS, TU_Id, D1.S_Id,
  AVG_W(ty_weighed_data (Mark, Coeff)) AS Mark
  FROM Graduate F, Students D1, Courses D2
  WHERE F.S_Id = D1.S_Id
     AND F.C_Id = D2.C_Id
  GROUP BY Semester, ECTS, TU_Id, D1.S_Id
GROUP BY Semester, S_Id;
```

Q2
```
SELECT TU_Id, S_Id, Mark
FROM ( SELECT TU_Id, S_Id,
  AVG_W(ty_weighed_data(Mark, Coeff)) AS Mark
  FROM Graduate F, Students D1, Courses D2
  WHERE F.S_Id = D1.S_Id
  AND F.C_Id = D2.C_Id
  GROUP BY TU_Id, S_Id)
WHERE Mark < 10;
```

Q3
```
SELECT C_Id, S_Id, Mark
FROM Graduate F, Students D1, Courses D2
WHERE Mark < 10
AND F.S_Id = D1.S_Id
AND F.C_Id = D2.C_Id;
```

Fig. 5. SQL Queries

However, our experiments will be extended to define the contours of our approach. Moreover, we must adapt existing lattice computing algorithms; this is very promising since our model allows pruning as we mentioned in the previous section.

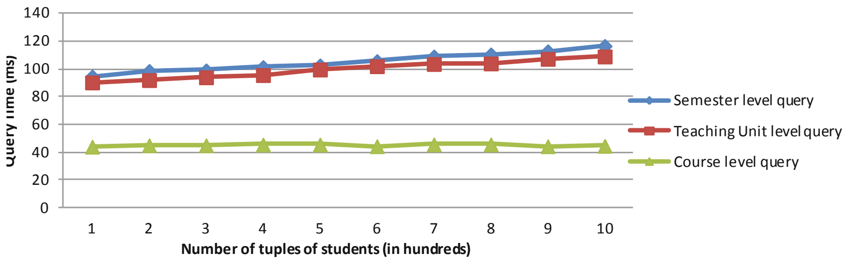


Fig. 6. Query execution results

6 Conclusion and Future Work

This paper defines a conceptual multidimensional data model flexible enough to allow the designer to specify differentiated and multiple aggregations. *Multiple*, as the same measure can be aggregated by several aggregation functions according to analysis axes and *differentiated* as these aggregations may vary, depending on the aggregation level. The model can combine a measure with different aggregation functions according to parameters. Furthermore, the model is expressive enough to check function calculations validity. At the logical level, the implementation can be optimized by a lattice of pre-computed aggregates, where invalid edges can be pruned.

We plan to extend our experimentations using different sets of tests to draw the outline of our approach (performance, storage volumes, complexity of the model, etc). We also plan to study OLAP manipulation operators on our model.

References

1. Abelló, A., Samos, J., Saltor, F.: YAM2: A multidimensional conceptual model extending UML. Information Systems 31(6), 541–567 (2006)
2. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 26(1), 65–74 (1997)
3. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual Design of Data Warehouses from E/R Schemes. In: Intl. Conf. HICSS 1998, vol. 7, pp. 334–343 (1998)
4. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: Intl. Conf. ICDE 1996, pp. 152–159 (1996)
5. Gyssens, M., Lakshmanan, L.V.S.: A Foundation for Multi-Dimensional Databases. In: Intl. Conf. VLDB 1997, pp. 106–115 (1997)
6. Harinath, S., Zare, R., Meenakshisundaram, S., Carroll, M., Guang-Yeu Lee, D.: Professional Microsoft SQL Server Analysis Services 2008 with MDX. Wiley Publishing, Indianapolis (2009)
7. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, USA (1996)
8. Jaecksch, B., Lehner, W.: The Planning OLAP Model - A Multidimensional Model with Planning Support. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 14–25. Springer, Heidelberg (2011)
9. Kotidis, Y., Roussopoulos, N.: DynaMat: A Dynamic View Management System for Data Warehouses. In: Intl. Conf. SIGMOD 1999, pp. 371–382 (1999)
10. Lujàn-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering 59, 725–769 (2006)
11. Mazón, J.N., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modelling. Data & Knowledge Engineering 68, 1452–1469 (2009)
12. Oliveira, R., Rodrigues, F., Martins, P., Moura, J.P.: Extending the Dimensional Templates Approach to Integrate Complex Multidimensional Design Concepts. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 26–38. Springer, Heidelberg (2011)
13. Pedersen, T.B., Jensen, C., Dyreson, C.: A foundation for capturing and querying complex multidimensional data. Information Systems 26, 383–423 (2001)
14. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Graphical Querying of Multidimensional Databases. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 298–313. Springer, Heidelberg (2007)
15. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. International Journal of Data Warehousing and Mining 4, 17–46 (2008)
16. Torlone, R.: Conceptual Multidimensional Models. In: Multidimensional Databases: Problems and Solutions, ch. 3, pp. 69–90. IGI Publishing Group (2003)
17. Vassiliadis, P., Sellis, T.K.: A Survey of Logical Models for OLAP Databases. SIGMOD Record 28(4), 64–69 (1999)

Genetic Algorithms-Based Symbolic Aggregate Approximation[*]

Muhammad Marwan Muhammad Fuad

Department of Electronics and Telecommunications
Norwegian University of Science and Technology (NTNU)
NO-7491 Trondheim, Norway
marwan.fuad@iet.ntnu.no

Abstract. Time series data appear in a broad variety of economic, medical, and scientific applications. Because of their high dimensionality, time series data are managed by using representation methods. Symbolic representation has attracted particular attention because of the possibility it offers to benefit from algorithms and techniques of other fields in computer science. The symbolic aggregate approximation method (SAX) is one of the most important symbolic representation techniques of times series data. SAX is based on the assumption of "high Gaussianity" of normalized time series which permits it to use breakpoints obtained from Gaussian lookup tables. The use of these breakpoints is the heart of SAX. In this paper we show that this assumption of Gaussianity oversimplifies the problem and can result in very large errors in time series mining tasks. We present an alternative scheme, based on the genetic algorithms (GASAX), to find the breakpoints. The new scheme does not assume any particular distribution of the data, and it does not require normalizing the data either. We conduct experiments on different datasets and we show that the new scheme clearly outperforms the original scheme.

Keywords: Time Series Mining, Symbolic Aggregate Approximation, Genetic Algorithms.

1 Introduction

A time series is a collection of observations at intervals of time points. These data appear in a broad variety of economic, medical, and scientific applications. Indexing, search and retrieval are the main topics of research in time series mining. Due to the numerous applications in which time series are involved, and the large size of time series databases, speed has always been the principal focus of all the methods and algorithms that deal with this type of data.

Time series data mining handles several tasks such as classification, clustering, similarity search, motif discovery, anomaly detection, and others. One key to perform these tasks efficiently and effectively is to use suitable indexing structures.

[*] This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie-Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

However, the high dimensionality of time series can cause indexing structures to fail to handle these data. One of the best solutions to deal with the high dimensionality of time series is to utilize a dimensionality reduction technique, also called a representation method, which helps represent the time series at a lower dimensional space, and then to use an indexing structure on this reduced space. The different tasks can then be processed in this reduced, lower dimensional space.

Using this scheme may result in two side effects; *false alarms* and *false dismissals*. False alarms are data objects that belong to the response set in the reduced space, but do not belong to the response set in the original space. False dismissals are data objects that the search algorithm excluded in the reduced space, although they are answers to the query in the original space. In order to guarantee no false dismissals the distance between the data objects in the reduced space should underestimate the distance in the original space. This condition is known as the *lower-bounding lemma.* [2]

A *tight* transformation is, by definition, one in which the distance defined on the reduced space is as close as possible (but always smaller) to the distance in the original space. Such a property increases the pruning power of the method thus decreases post-processing time.

There have been different suggested representation methods in the literature, to mention a few; *Discrete Fourier Transform* (DFT) [2] and [3], *Discrete Wavelet Transform* (DWT) [5], *Singular Value Decomposition* (SVD) [11], *Adaptive Piecewise Constant Approximation* (APCA) [8], *Piecewise Aggregate Approximation* (PAA) [7] and [21], *Piecewise Linear Approximation* (PLA) [16], *Chebyshev Polynomials* (CP) [4], etc.

Among representation methods of time series, symbolic representation of time series has several advantages which have particularly interested researchers in this field of computer science. One of its main advantages is that symbolic representation allows researchers to benefit from the ample symbolic algorithms known in the text-retrieval and bioinformatics communities [13].

There have been many suggestions to represent time series symbolically. But in general, most of these methods suffered from two main inconveniences [14]: the first is that the dimensionality of the symbolic representation method is the same as that of the original space, so there is no virtual dimensionality reduction. The second drawback is that although distance measures have been defined on the reduced symbolic spaces, these distance measures are poorly correlated with the original distance measures defined on the original spaces.

One of the most widely-known symbolic representation methods of time series is SAX. SAX is based on an assumption that normalized time series have a highly Gaussian distribution. This assumption permits SAX to determine the locations of breakpoints by using Gaussian lookup tables. This in turn enables SAX to use pre-computed distances.

In this work we show that this assumption of Gaussianity oversimplifies the problem and may result in very high values of error when performing time series mining tasks. We present a new method to determine the breakpoints. The new method is based on the genetic algorithm. This new method does not presume any particular distribution of the time series. We show experimentally how the new method outperforms the original one.

2 Background

2.1 The Symbolic Aggregate Approximation (SAX)

The *Symbolic Aggregate approXimation* method (SAX) [13] stands out as probably the most powerful symbolic representation method of time series. The main advantage of SAX is that the similarity measure it utilizes, called MINDIST, uses statistical lookup tables. This makes SAX easy to compute with an overall complexity of $O(N)$.

SAX is based on an assumption that normalized time series have “highly Gaussian distribution”, so by determining the breakpoints that correspond to a particular alphabet size, one can obtain equal-sized areas under the Gaussian curve. SAX is applied as follows:

1- The time series are normalized.
2- The dimensionality of the time series is reduced by using PAA [7], [21].
3- The PAA representation of the time series is discretized.

This is achieved by determining the number and location of the breakpoints. The number of breakpoints is related to the desired alphabet size (which is chosen by the user); i.e. *number_of_breakpoints = alphabet_size -1*. Their locations are determined, as mentioned above, by using Gaussian lookup tables. The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of PAA that lies within that interval is discretized by that symbol. The last step of SAX is using the following similarity measure:

$$MINDIST(\hat{S},\hat{R}) \equiv \sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}\left(dist(\hat{s}_i,\ \hat{r}_i)\right)^2} \tag{1}$$

Where n is the length of the original time series, N is the number of segments, $\hat{S}$ and $\hat{R}$ are the symbolic representations of the two time series S and R, respectively, and where the function $dist(\ )$ is implemented by using the appropriate lookup table.

We also need to mention that the similarity measure used in PAA is:

$$d(S,R) = \sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}\left(\overline{s_i} - \overline{r_i}\right)^2} \tag{2}$$

Where n is the length of the time series, N is the number of segments.

It is proven in [7], [21] that the above similarity measure is lower bounding of the Euclidean distance applied in the original space of time series. This results in the fact that MINDIST is also lower bounding of the Euclidean distance, because it is lower bounding of the similarity measure used in PAA. This guarantees that PAA produces no false dismissals.

There are other versions and extensions of SAX [9], [18], [19], [20], [17]. These versions use it for other applications or to apply it to index massive datasets, or compute MINDIST differently. However, the version of SAX that we presented earlier, which is called *classic* SAX, is the base of all these versions and extensions and it is actually the most widely-known one.

2.2 The Genetic Algorithms (GAs)

The *Genetic Algorithms* are an optimization and search technique based on the principles of genetics and natural selection [12]. GA has the following elements: a population of individuals, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring [15]. GA creates an environment in which a population of individuals, representing solutions to a particular problem, is allowed to evolve under certain rules towards a state that minimizes, in terms of optimization, the value of a function which is usually called the *fitness function*.

There has been a great amount of research on the genetic algorithms and their applications in many fields of science and engineering. In the following we present a description of the simple, classical GA. The first step of GA is defining the problem variables and the fitness function. The range of the variable values can be constrained or unconstrained. A particular configuration of variables produces a certain value of the fitness function and the objective of GA is to find the configuration that gives the "best" value of the fitness function. This configuration represents the best solution to the problem. This solution can be optimal or close-to-optimal.

GA starts with a collection of individuals, also called *chromosomes*, each of which represents a possible solution to the problem at hand. This collection of randomly chosen chromosomes constitutes a population whose size $popsize$ is chosen by the algorithm designer. This step is called *initialization*. The variables of the problem can be encoded using different schemes the most famous of which is real-valued encoding. In this scheme a candidate solution is represented as a real-valued vector in which the dimension of the chromosomes is constant and equal to the dimension of the solution vectors [1]. This dimension is denoted by nbp. The fitness function of each chromosome is evaluated. The next step is *selection*. The purpose of this procedure is to determine which chromosomes are fit enough to survive and possibly produce offspring. This is decided according to the fitness function of the chromosome in that the higher the fitness function is the more chance is has to be selected for mating. There are several selection methods such as the *roulette wheel selection*, *random selection*, *rank selection*, *tournament selection*, and others [15]. The percentage of chromosomes selected for mating is denoted by $srate$. *Crossover* is the next step in which offspring of two parents are produced to enrich the population with fitter chromosomes. There are several approaches to perform this process, the most common of which is *single-point* crossover and *multi-point* crossover.

While crossover is the mechanism that enables the GA to communicate and share information about fitter chromosomes, it is not sufficient to efficiently explore the search space. *Mutation*, which is a random alteration of a certain percentage $mrate$ of chromosomes, is the other mechanism which enables the GA to examine unexplored regions in the search space. It is important to keep a balance between crossover and mutation. High crossover rate can cause converging to local minima and high mutation rate can cause very slow convergence.

Now that a new generation is formed, the fitting function of the offspring is calculated and the above procedures repeat for a number of generations *NrGen* or until a stopping criterion terminates the algorithm.

3 Genetic Algorithms-Based SAX (GASAX)

The heart of SAX, as we saw in Section 2, is the assumption that normalized time series have a highly Gaussian distribution. This is an intrinsic part of the method because it permits localizing the breakpoints, which, in turn, allows SAX to use pre-computed distances, which is the main advantage of SAX over other methods.

Nevertheless, this assumption is a very general one that does not take into account the dataset to which SAX is applied. The direct result of this assumption of Gaussianity is the weak, even very weak, performance of SAX on certain datasets (as we will show later in Section 4).

Another direct result of this assumption of Gaussianity is the requirement of normalizing the time series before applying SAX. This requirement can restrict the application of SAX in certain cases. We give the example of streaming time series that undergo trends. Correct normalizing of such times series requires that all the data be available before the normalization process.

In this paper we popose a different scheme to localize the breakpoints. The new scheme does not presume any distribution of the time series. It does not even require normalizing the data as it can be applied to normalized and non- normalized time series. The main point here is that this scheme does not presume either that the normalized time series are non-Gaussian. It simply localizes the breakpoints without any particular assumption of the data distribution, whether these data are normalized or not.

Our proposed scheme, GASAX, processes in the same manner illustrated in Section 2.1, but instead of using Gaussian lookup tables; it uses the genetic algorithms to localize the breakpoints.

As we saw in Section 2.2, applying the genetic algorithms requires choosing an appropriate fitness function. There are several fitness functions that can be applied with our method. Time series data mining handles several tasks, one of the most important of which is classification. In a classification task we have categorical variables which represent classes, and the task is to assign class labels to the dataset according to a model learned from a learning phase on a training data where the classes are known. So we can choose the classification error as a fitness function; i.e. we aim at minimizing the classification error and find the breakpoints, for each value of the alphabet size, which minimize the classification error. Another possible fitness function that can be used with GASX is the average tightness (the rate of the similarity measure using GASAX with a certain configuration of the breakpoints to the Euclidean distance in the original space. See Section 1 for the definition of tightness). This tightness can be calculated on each pair of time series in the dataset or on a randomly chosen sample of the time series in the dataset. Since most GA problems are formulated as minimization problems, we can choose the fitness function in this case to be the function that minimizes the difference between the similarity measure using GASAX (with a certain configuration of the breakpoints) and the Euclidean distance in the original space. Other fitness functions can also be used with GASAX in addition to the two functions mentioned above.

However, because computational cost is an issue to be considered when applying the genetic algorithms, we used the classification error as a fitness function in our experiments because we wanted to test the worst-case application of our method.

In our problem each chromosome is a vector that represents potential locations of the breakpoints for a particular alphabet size, so the optimal solution constitutes the locations of the breakpoints, for that particular alphabet size, which minimize the classification error. The alphabet size, as in the case with the original SAX, is chosen by the user.

While genetic algorithms usually define the different control parameters (crossover rate, mutation rate, etc) before starting the algorithm and these parameters remain unchanged, GASAX proposes an approach in which some of these parameters are modified dynamically during the GASAX run.

The variables of GASAX are constrained by the following condition:

$$\text{if} \qquad \alpha < \beta \Rightarrow l_\alpha < l_\beta \qquad \forall \alpha, \beta \in \Sigma \tag{3}$$

where Σ is the alphabet used, l_α, l_β are the locations of α, β, respectively. This condition is obvious for one chromosome, but taking into account the different crossover and mutation processes, this condition should hold under all such possible processes that the chromosome may undergo, so the above condition should be modified to become:

$$\text{if} \qquad \alpha < \beta \Rightarrow \max l_\alpha < \min l_\beta \qquad \forall \alpha, \beta \in \Sigma \tag{4}$$

where *max* and *min* are taken for all the chromosomes. This condition means that for all the chromosomes, the right-most location of one character in the alphabet should not exceed the left-most location of the character that comes directly after that first character. Condition (4) cannot be modified during the GASAX run.

The mutation rate is modified dynamically during the GASAX run. This means that GASAX starts with a relatively small value of the mutation rate and if the fitness function remains unchanged for several generations, this indicates that GASAX is stuck in a local minimum, so the mutation rate is increased for the next generations. This is particularly important for the first generations.

4 Experiments

We compared our method GASAX against SAX on the datasets available at [10], which is the same archive on which the original SAX was tested. This archive makes up between 90% and 100% of all publicly available, labeled time series data sets in the world, and it represents the interest of the data mining/database community, and not just one group [6].

We tested our method in a classification task based on the first near-neighbor (1-NN) rule using leaving-one-out cross validation. This means that every time series is compared to the other time series in the dataset. If the 1-NN does not belong to the same class, the error counter is incremented by 1.

In the experiments we conducted we had to normalize the time series for comparison reasons only, since SAX, as indicated earlier, can only be applied to normalized time series, but GASAX does not require normalization.

The two methods SAX and GASAX were tested for all values of the alphabet size on which SAX is defined (3 through 20). However, GASAX does not require predefined lookup tables so it can practically be used for any value of the alphabet size.

As indicated in Section 3, for each value of the alphabet size, we train GASAX on the training sets, using the classification error as a fitness function, to obtain the optimal values of the breakpoints for that alphabet size. Then these breakpoints are used to find the classification error of the testing sets of the corresponding dataset. There is no training phase for SAX and it is applied directly to the testing sets.

For GASAX, the population size (the number of chromosomes) was between 12 and 16, depending on the size of the dataset. These chromosomes were encoded using a real-valued encoding scheme. We used two stopping criteria with GASAX: the first is the number of generations, which was set to 100, and the second is the value of the fitness function, which was set to 0. GASAX terminated and exited as soon as one of these criteria was met. These criteria are in fact strict and could have been relaxed.

Concerning the first one, for example, in many cases the algorithm gave good solutions after 20 or 30 generations, or even earlier. The second criterion could have also been relaxed to a value appropriate to the dataset in question. We know from experience that the classification error of some datasets is high for any time series representation method used and it is low for others. However, we wanted to test GASAX under strict conditions which do not assume any prior knowledge about the dataset tested, so GASAX is based completely on the knowledge acquired at run time only. Table 1 summarizes the symbols used in the experiments together with their corresponding values.

Table 1. The symbol table together with the corresponding values used in the experiments

popsize	Population size	12-16
NrGen	Number of generations	100
mrate	Mutation rate	0.2
srate	Selection rate	0.5
nbp	Number of parameters	varies

Figure 1 shows some of the results we obtained. As we can see from the figure, GASAX clearly outperforms SAX for all values of the alphabet size and for all the datasets presented. We obtained similar results when we conducted these experiments on the other datasets. Dataset (Coffee) was particularly adapted to GASAX. As we can see, the classification error for this dataset was 0 in 14 of the 18 values of alphabet size. This also shows the utility of using the 0 value of the fitness function as a stopping criterion, because, interestingly, GASAX terminated very quickly on this dataset as it found the optimal values of the breakpoints after few generations only. Having found that GASAX gave a 0 value of the classification error on this dataset, we wanted to conduct further experiments on this dataset, so we ran GASAX several times. We found that the optimal solution was not unique, and the classification error was also 0 for other values of the breakpoints for certain values of the alphabet size.

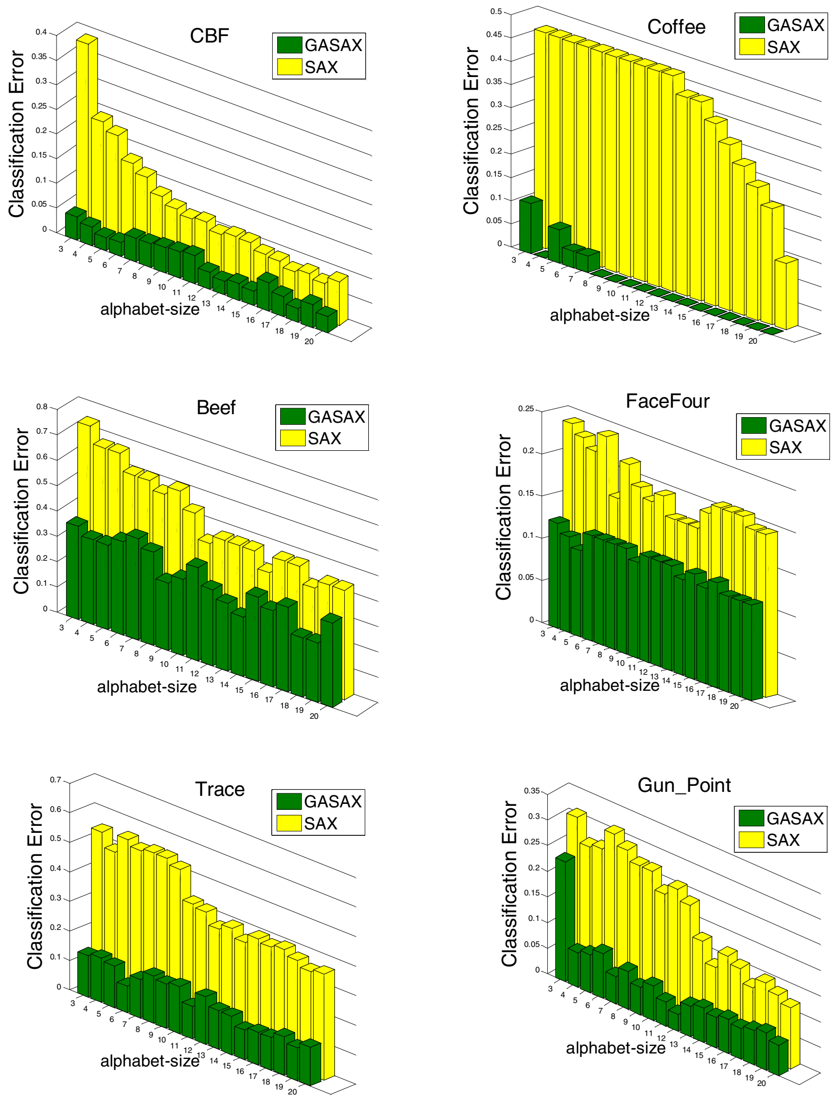


Fig. 1. Comparison between the classification errors of GASAX and SAX on datasets (CBF), (Coffee), (Beef),(FaceFour),(Trace) and (Gun_Point), for alphabet size between 3 and 20

Table 2. The minimum and maximum values of the normalized data, the breakpoints, and the training errors of datasets (CBF) and (Coffee) for alphabet size 3 through 10 using GASAX

Dataset	MinMax Values	Alphabet Size	Breakpoints	Training Error
CBF	[-2.317 3.245]	3	[0.979 1.16]	0.033
		4	[0.855 1.017 1.825]	0
		5	[0.824 1.029 1.062 1.848]	0
		6	[0.696 0.780 0.909 1.057 1.59]	0
		7	[0.337 0.626 0.925 1.053 1.928 2.833]	0
		8	[-1.144 0.244 0.582 0.736 0.876 0.975 1.582]	0
		9	[-1.826 -1.679 -1.487 0.537 0.714 0.848 0.945 1.559]	0
		10	[-1.907 -1.512 -1.448 -1.133 -0.979 0.692 0.718 0.898 1.357]	0
Coffee	[-2.064 2.177]	3	[0.792 0.798]	0.107
		4	[-0.337 0.887 0.906]	0
		5	[-0.434 0.557 0.731 0.762]	0
		6	[-0.535 0.39 0.728 0.84 0.937]	0
		7	[-0.862 -0.276 -0.153 0.158 0.938 0.979]	0
		8	[-0.979 -0.878 0.014 0.485 0.778 0.818 0.925]	0
		9	[-0.625 -0.096 0.642 0.717 0.846 0.853 0.898 0.932]	0
		10	[-0.627 0.026 0.106 0.394 0.564 0.821 0.855 0.962 1.127]	0

In Table 2 we show the breakpoints obtained by applying our method GASAX on some of the datasets presented in Figure 1 in addition to the corresponding classification error on the training sets (because of space restrictions, we present the results of two datasets only and for alphabet size between 3 and 10). The figures are rounded to 3 decimal places. The breakpoints of SAX are presented later in Table 3.

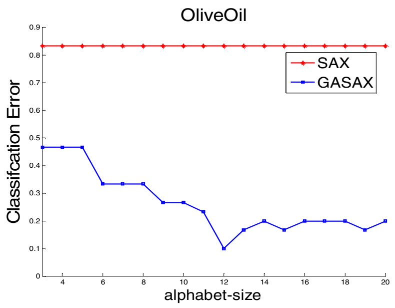


Fig. 2. A comparison between the classification error of GASAX and that of SAX on the testing set of (OliveOil).

As indicated earlier, the breakpoints of SAX are the same for all datasets and they are obtained from the lookup tables of the Gaussian distribution. We also show the minimum and maximum values of the normalized data of those datasets

We can see from Figure 1 and Table 2 that these datasets do not follow a Gaussian distribution and that the breakpoints, taking into account the minimum and maximum values of the normalized time series, are in general very different from those used in SAX.

The results of a certain dataset (OliveOil) were particularly interesting. In data mining, the use of a random classifier (one that classifies time series completely at random) results in an error that is equal to $\frac{NbrClass-1}{NbrClass}$, where $NbrClass$ is the number of classes of that dataset. This error value of the random classifier constitutes the worst-case value that no classification method should exceed. (OliveOil) has 4 classes, so the classification error of a random classifier on that dataset is 0.750. Surprisingly, the classification error of SAX on (OliveOil) is 0.833. This error is not only very high, but it even exceeds that of a random classifier. Another strange result is that using SAX on that dataset yielded the same classification error for all values of the alphabet size. After applying GASAX to that dataset the classification error decreased to almost half that of SAX, and much less for some values of the alphabet size (it dropped to 0.1 for alphabet size=12) as we can see in Figure 2 which shows the classification error on the testing set of (OliveOil) for both SAX and GASAX.

Table 3. Comparison between the classification error of SAX and that GASAX on the testing of (OliveOil)

Method	**MinMax Values**	**Alphabet Size**	**Breakpoints**	**Testing Error**
SAX	[-1.0 3.719]	3	[-0.43 0.43]	0.833
		4	[-0.67 0 0.67]	0.833
		5	[-0.84 -0.25 0.25 0.84]	0.833
		6	[-0.97 -0.43 0 0.43 0.97]	0.833
		7	[-1.07 -0.57 -0.18 0.18 0.57 1.07]	0.833
		8	[-1.15 -0.67 -0.32 0 0.32 0.67 1.15]	0.833
		9	[-1.22 -0.76 -0.43 -0.14 0.14 0.43 0.76 1.22]	0.833
		10	[-1.28 -0.84 -0.52 -0.25 0 0.25 0.52 0.84 1.28]	0.833
GASAX		3	[0.914 0.93]	0.467
		4	[-0.4 0.91 0.93]	0.467
		5	[-0.39 0.43 0.91 0.92]	0.467
		6	[-0.049 -0.034 0.402 1.303 1.307]	0.333
		7	[-0.05 -0.03 0.40 1.3035 1.3065 2.23]	0.333
		8	[-0.048 -0.044 0.01 0.016 0.787 0.794 0.798]	0.333
		9	[-0.47 -0.456 -0.039 -0.036 1.061 1.062 1. 4349 1.4352]	0.267
		10	[-0.445 -0.433 -0.431 -0.115 -0.009 -0.005 0.908 0.935 1.666]	0.267

Table 3 shows the classification error on the testing set of (OliveOil), together with the corresponding breakpoints using GASAX. As mentioned earlier, these breakpoints are obtained by applying GASAX on the training set to obtain the optimal breakpoints, and then we use these optimal breakpoints on the testing set to obtain the classification error. Table 3 also shows the breakpoints given by SAX (which, as indicated earlier, are the same for all datasets). We can easily see that the assumption of Gaussianity of normalized time series oversimplifies the problem, which results in a very high error value for this dataset. In fact, as we can see from Table 3 , the breakpoints of SAX even go out of range staring alphabet size=7 which makes the representation useless. We witnessed similar cases with other datasets too.

5 Conclusion

In this paper we introduced a new scheme, GASAX, which uses the genetic algorithms to determine the optimal locations of the breakpoints, which is the main idea of SAX; one of the most important methods of time series representation methods. We showed how these optimal breakpoints found by GASAX can successfully boost the performance of SAX. The new scheme GASAX does not assume any particular distribution of the time series and does not require that the time series be normalized, unlike SAX which requires normalizing the time series to localize the breakpoints. SAX also assumes that normalized time series have a Gaussian distribution. This assumption, as we illustrated, can result in a high error rate.

We validated the new scheme by conducting classification task experiments on different datasets. The experiments showed that our new algorithm, GASAX, clearly outperforms SAX for all values of the alphabet size, and for all the datasets tested.

SAX uses predefined lookup tables, so it can only be applied for alphabet size on which lookup tables are defined, but GASAX is not based on predefined lookup tables, so it can be applied for any value of the alphabet size.

The training phase of GASAX is performed offline to determine the locations of the breakpoints, so GASAX has exactly the same online speed as that of SAX.

While our direct objective of this work was to use the genetic algorithms on a particular problem of time series representation, our indirect aim was to explore the application of one optimization methodology to this domain of research. We believe that the genetic algorithms, and other optimization methodologies, still have many applications in time series data mining and information retrieval.

References

1. Affenzeller, M., Winkler, S., Wagner, S., Beham, A.: Genetic Algorithms and Genetic Programming Modern Concepts and Practical Applications. Chapman and Hall/CRC (2009)
2. Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search in Sequence Databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)

3. Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K.: Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. In: Proceedings of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland, pp. 490–501 (1995)
4. Cai, Y., Ng, R.: Indexing Spatio-temporal Trajectories with Chebyshev Polynomials. In: SIGMOD (2004)
5. Chan, K., Fu, A.W.: Efficient Time Series Matching by Wavelets. In: Proc. of the 15th IEEE Int'l Conf. on Data Engineering, Sydney, Australia, March 23-26, pp. 126–133 (1999)
6. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. In: Proc. of the 34th VLDB (2008)
7. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. J. of Know. and Inform. Sys. (2000)
8. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Locally Adaptive Dimensionality Reduction for Similarity Search in Large Time Series Databases. In: SIGMOD, pp.151–162 (2001)
9. Keogh, E., Lin, J., Fu, A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In: Proc. of the 5th IEEE International Conference on Data Mining (ICDM 2005), Houston, Texas, November 27-30 (2005)
10. Keogh, E., Zhu, Q., Hu, B., Hao, Y., Xi, X., Wei, L., Ratanamahatana, C.A.: The UCR Time Series Classification/Clustering, Homepage (2011), `http://www.cs.ucr.edu/~eamonn/time_series_data/`
11. Korn, F., Jagadish, H., Faloutsos, C.: Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. In: Proceedings of SIGMOD 1997, Tucson, AZ, pp. 289–300 (1997)
12. Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms with CD-ROM. Wiley-Interscience (2004)
13. Lin, J., Keogh, E., Lonardi, S., Chiu, B.Y.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In: DMKD 2003, pp. 2–11 (2003)
14. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a Novel Symbolic Representation of Time Series. DMKD Journal (2007)
15. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
16. Morinaka, Y., Yoshikawa, M., Amagasa, T., Uemura, S.: The L-index: An Indexing Structure for Efficient Subsequence Matching in Time Sequence Databases. In: Proc. 5th Pacific Aisa Conf. on Knowledge Discovery and Data Mining, pp. 51–60 (2001)
17. Muhammad Fuad, M.M., Marteau, P.-F.: Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables. In: Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C. (eds.) KES 2010. LNCS, vol. 6276, pp. 420–431. Springer, Heidelberg (2010)
18. Shieh, J., Keogh, E.: iSAX. Disk-Aware Mining and Indexing of Massive Time Series Datasets. Data Mining and Knowledge Discovery (2009)
19. Shieh, J., Keogh, E.: *iSAX*: Indexing and Mining Terabyte Sized Time Series. In: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27 (2008)
20. Wei, L., Keogh, E., Xi, X.: SAXually Explict Images: Finding Unusual Shapes. In: ICDM (2006)
21. Yi, B.K., Faloutsos, C.: Fast Time Sequence Indexing for Arbitrary Lp Norms. In: Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt (2000)

A Neural-Based Approach for Extending OLAP to Prediction

Wiem Abdelbaki[1,2], Riadh Ben Messaoud[2], and Sadok Ben Yahia[1]

[1] Faculty of Sciences of Tunis – El-Manar University, Tunisia
[2] Faculty of Economics and Management of Nabeul – Carthage University, Tunisia

Abstract. In the Data Warehouse (DW) technology, On-line Analytical Processing (OLAP) is a good applications package that empowers decision makers to explore and navigate into a multidimensional structure of precomputed measures, which is referred to as a *Data Cube*. Though, OLAP is poorly equipped for forecasting and predicting empty measures of data cubes. Usually, empty measures translate inexistent facts in the DW and in most cases are a source of frustration for enterprise managements, especially when strategic decisions need to be taken. In the recent years, various studies have tried to add prediction capabilities to OLAP applications. For this purpose, generally, Data Mining and Machine Learning methods have been widely used to predict new measures' values in DWs. In this paper, we introduce a novel approach attempting to extend OLAP to a prediction application. Our approach operates in two main stages. The first one is a preprocessing one that makes use of the Principal Component Analysis (PCA) to reduce the dimensionality of the data cube and then generates *ad hoc* training sets. The second stage proposes a novel OLAP oriented architecture of Multilayer Perceptron Networks (MLP) that learns from each training set and comes out with predicted measures of inexistent facts. Carried out experiments demonstrate the effectiveness of our proposal and the performance of its predictive capabilities.

Keywords: Data Warehouse, OLAP, Data Mining, Principal Component Analysis, Machine Learning, Multilayer Perceptron, Prediction.

1 Introduction

Inmon defines a Data Warehouse (DW) as a subject oriented, nonvolatile, integrated, time variant collection of data, in support of management's decisions [1]. A DW is a corner stone in the Business Intelligence (BI) process. It is used to store analysis contexts within multidimensional data structures referred to as *Data Cubes*. Then, data cubes are usually manipulated by using On-line Analytical Processing (OLAP) applications to allow senior managers exploring information and getting BI reportings thought interactive and friendly dashboards.

However, even if a DW should fundamentally contain integrated data [1], generally exploration of cubes disclose a sparse structure within several empty measures. In the DW model, empty measures correspond to inexistent facts generally reflecting either out-of-date events that did not happen, or future events that have not yet occurred and may possibly happen in the future. Empty measures can cause the delivery of irrelevant analysis and be a source of frustration for the enterprise management, especially when strategic decisions need to be taken. We argue that predicting these measures can consolidate the BI reporting and could even provide new opportunities to DW customers by enlarging their dashboard picture and empowering them with knowledge on what should occur if inexistent facts had already happened.

For instance, it will be very useful to a car sale company to predict the potential turnover that a new agency can produce in a new city by the next year's end. This indicator will definitely help the company's management to assess the potential investment. So far, inexistent measures in a data cube may potentially be learned from its neighborhood. Agarwal and Chen affirm that making future decisions over historical data is one important goal of OLAP [2]. However, OLAP is restricted to exploration and not equipped with a framework to empower user investigation of interesting information. In fact, despite the fundamental Cood's statement of goal seeking analysis models (such as "What if" analysis) required in OLAP applications since the early 90's [3], most of today's OLAP products still lack an effective implementation of this feature.

Hence, several studies have been proposing to enhance OLAP applications by using Data Mining and Machine Learning methods in order to respond to various analysis purposes like cube exploration [4] and association rules mining [5]. Recently, new approaches are attempting to extend OLAP to prediction capabilities, in order to anticipate forthcoming events [6,2]. However, as far as we know, none of them is providing DW customers with explicit values instead of inexistent measures.

In this paper, we propose a novel approach for extending OLAP to prediction using a machine learning technique. It aims at producing explicit values for inexistent measures in a data cube using Multilayer Perceptron Networks (MLP) known for their prediction performances [7,8]. However, MLPs are not adapted to perform on multidimensional data such as data cubes. So, we propose a novel MLP architecture suited to multidimensional structure of data. Moreover, volumes of data stored in real-world OLAP cubes are usually huge [9]. The data size overlapped with the multidimensional structure leads generally to highly correlated measures values which generate confusion over MLP during the training process and thus, degrades their generalization capability [10]. So we propose to associate our proposal with a preprocessing stage to provide the neural network with *ad hoc* training sets that contain reduced and decorrelated predictors generated over the original measures.

This paper is organized as follows. In Section 2, we expose a state of the art of works related to predictions in data cubes. In Section 3, we present the main objectives and the general overview of our approach. Section 4 details the methods formalization that we followed in the two stages of our proposal. In Section 6, we carry out experiments investigating the effectiveness of our proposal and the performance of its predictive capabilities. Finally, Section 7 summarizes our findings and addresses future research directions.

2 Related Work

In the recent years, several studies have addressed the issue of extending OLAP to advanced analysis capacities. As they were driven under different objectives (discovery-driven exploration of cubes, cube mining, cube compression, and so on), they are based on various concepts and methodologies. In line with our concern, we focus on those having a close linkage with prediction in DWs.

Table 1. Proposals integrating prediction in data cubes

Proposal	Goal	Optimization	Reduction	Measures	Values
Sarawagi *et al.* [4]	Exploration	+	-	-	-
Palpanas *et al.* [9]	Compression	-	+	-	-
Chen *et al.* [11]	Prediction	+	-	+	-
Chen *et al.* [12]	Compression	-	+	+	-
Bodin-Niemczuk *et al.* [6]	Prediction	+	-	-	-
Agarwal and Chen [2]	Prediction	+	-	+	-
Our approach	Prediction	-	+	+	+

We summarize in Table 1 proposals that have attempted to extend OLAP to prediction. These proposals are detailed according to five main criteria: (1) What is the overall goal of the proposal? (2) Does the proposal include an algorithmic optimization? (3) Does it use a reduction technique? (4) Does it introduce new classes of measures? and (5) Does it provide explicit predicted values of empty measures? We note (+) if the proposal meets the criteria, and (-) if not.

Sarawagi *et al.* proposed to assist DW users when exploring data by detecting exceptions. Their approach is based on a log linear model [4]. In order to compress data cubes, Palpanas *et al.* used the principle of information entropy to build a probabilistic model capable of detecting measure deviations [9]. Their approach predicts low-level measures from high-level precalculated aggregates. Chen *et al.* introduced the concept of *Prediction Cubes* where a score or a probability distribution of measures are fetched beside their original values [11]. Prediction Cubes are then used to build prediction models. Chen *et al.* proposed a new type of multidimensional structures called *Regression Cubes* [12]. They contain compressible measures and each cell of a Regression Cube indicates the general tendency and the variation compared to its original measures. Agarwal and Chen built a new data cube class called *Latent-Variable Cube* [2] characterized by its ability to compute aggregate functions, such as mean and variance, over latent variables detected by a statistical model. Bodin-Niemczuk *et al.* proposed to equip OLAP with a regression tree to predict measures of forthcoming facts [6].

From the above cited references, an outstanding common observation dealing with data dimensionality need to be addressed. In fact, one of the most challenging issues of integrating predictive models into OLAP data cubes concerns the multidimensional structure of data and the usual huge facts' volumetry in DWs. This could be of a negative effect on the prediction performance, which is supposed to provide BI reporting costumers with fast and accurate results in line with OLAP applications. We note that some of the above proposals consider a pre-processing stage to reduce the dimensionality effect on the algorithms performance [9,12]. Some others rather rely on heuristics to optimize implemented algorithms [4,11,2]. In our case, we include a PCA-based preprocessing stage in our approach to reduce the dimensionality of the data cube and generate adequate training sets for the prediction stage.

While some approaches provide approximations of inexisting measures [4,9,6], some others introduce new classes of data cubes within new measures generated over the existing ones [11,12,2]. This is generally argued by the need for additional indicators to enrich the BI reporting. In our approach, we attempt to enlarge the dashboard picture for the decision maker by giving explicit predicted values of inexisting measures. To do so, we base our prediction model on a machine learning approach that uses a novel architecture of MLP networks. In addition, we introduce a new class of data cubes integrating customized measures referring to predictors stored in an external database.

3 Objectives and Overview

Our proposal consists in integrating a predictive model in the OLAP environment in order to enrich the decision-making process. We aim at predicting inexistent measures according to the existent ones while dealing with large data cubes. We give a formalized framework of our proposal with respect to traditional BI customers' needs and integrate in a machine learning algorithm. Our overarching objectives can be summed up in the following points:

1. Generating reduced training sets from the original data cube;
2. Adapting MLPs to the multidimensional structure of data;
3. Predicting an explicit value of an inexistent measure;
4. Assess predicted measures with quality indicators.

As far as we know, despite their proven performances [8,7], MLPs are not yet used for prediction purposes in OLAP cubes. On the other hand, the multidimensional structure and the volume of data cubes are challenging issues when data mining or machine learning frameworks needs to be embedded into OLAP applications. In particular, attempting to integrate the MLPs within OLAP needs to fix these two issues. In fact, they are not designed to deal with a multidimensional structure of data. Moreover, the size of a real-world data cube leads to highly correlated measures values with a lot of noise and redundancy. According to Bishop [10], these characteristics may corrupt the training phase of the MLPs and degrades their generalization capability.

Therefore, the first stage of our approach consists in generating reduced training sets over the original cube comparing to the original data structure in terms of measures and dimensions' attributes. In order to do so, we resort to the Principal Component Analysis (PCA) procedure and exploit its orthogonal transformation to convert the correlated cube attributes into a smaller set of principal components [13]. The obtained components design a linearly combination of the original attributes and concentrate the largest possible variance of the original measures. Recently some studies have been interested in the factorial approach to fulfill prediction tasks, considering the principal components as independent variables of their prediction models [14,15]. We intend to follow this trail as a one-shot backstage preprocessing step that ensures the generation of complementary training sets that preserves the measure variation and the semantics linking attributes and dimensions. On the other hand, neural networks are recognized as a potent prediction tool with research and applications [7]. One of the most used MLPs are multilayer perceptrons with a backpropagation process. This is due to their operation simplicity, their excellent generalization capacity and their ability to approximate any universal function [16]. Thus, for the second stage, we apply an MLP learning process on the generated data sets. In order to overtake the multidimensional aspect of these input data, we propose a novel MLP architecture based on an interconnection of a multiple childre-networks. It involves several training sets in the same learning process without having to merge them and yet generates a unique prediction value for each targeted one.

It is important to note that our approach is not a completion technique and is not supposed to fill all empty measures of a data cube. The main objective of our proposal is to promptly come-out with prediction to any empty measure upon the request of the OLAP user in order to anticipate a decisive indicator such as a future turnover as the example described previously in Section 1.

4 Formalization of Our Proposal

4.1 General Notations

In our framework, we propose to reuse the same notations and definitions provided in [5]. We assume that the user has set an hierarchical level per dimension with l_i categorical attributes per dimension; $1 \leq i \leq d$. Thus, let $\mathcal{C}$ be a data cube having the following proprieties:

- $\mathcal{C}$ has a non empty set of d dimensions $\mathcal{D} = \{D_i\}_{(1 \leq i \leq d)}$;
- $\mathcal{C}$ contains a non empty set of m measures $\mathcal{M} = \{M_q\}_{(1 \leq q \leq m)}$;
- Each dimension D_i contains l_i attributes, i.e., $\mathcal{A} = \{a_1^i, ..., a_t^i, ..., a_{l_i}^i\}$ is the set of dimension attributes D_i.

4.2 Dimensions Reduction

The main purpose of this stage is to generate concentrated, information-preserving training sets over the original data cube. To do so, we intend to exploit the PCA as a reduction technique. However, to respect OLAP dependencies, the data transformation requires to keep on the original variation of the cube measures according to all the concerned dimensions, otherwise, the training sets may promote some dimensions at the expense of others. Therefore, we decompose the cube in a rigorous manner which preserves the multidimensional aspect of measures' variation and respect the semantics connecting each dimension attributes.

Technically, the same slice of a data cube generates two transposed matrices. While applying a numerical analysis technique like PCA, these two matrices provide two different results. To meet the basic idea of our approach, which aims at covering all variations of the measure, we propose to treat separately all extractable matrices from an OLAP slice. To do so, we provide the notion of *Cube-face* that identifies all the possible configurations of a data cube and the notion of *PCA-slice* that captures the variations of the measures according to attributes' semantics of specific dimensions.

Definition 1 (Cube-face). *Let* $\{D_k, D_v, D_{s_1}, \ldots, D_{s_i}, \ldots, D_{s_{d-2}}\}$ *be a non-empty subset of* d *distinct dimensions. We denote by* $Cf(D_k, D_v, (D_{s_1}, \ldots, D_{s_i}, \ldots, D_{s_{d-2}}))$ *a Cube-face of a data cube* $\mathcal{C}$. *A Cube-face is identifiable by the geometrical positions of its dimensions that we call: Key dimension* D_k, *Variant dimension* D_v *and a set of* $(d-2)$ *Slicer dimensions* $\{D_{s_i}\}$; $i \in [1, d-2]$. *The number of extractable cube-faces over a data cube is equal to the number of its geometrical faces.*

Definition 2 (PCA-slice). $T(D_k, D_v, (a^o_{t_1}, \ldots, a^p_{t_i}, \ldots, a^q_{t_{d-2}}))$ *is the PCA-slice obtained by applying the* Slice *operator on* Cf; *with* $a^o_{t_1}$ $a^p_{t_i}$ *and* $a^q_{t_{d-2}} \in D_{s_1}, D_{s_i}$ *and* $D_{s_{d-2}}$, *respectively.*

Note that $T(D_k, D_v, (a^o_{t_1}, \ldots, a^p_{t_i}, \ldots, a^q_{t_{d-2}})) \neq T(D_v, D_k, (a^o_{t_1}, \ldots, a^p_{t_i}, \ldots, a^q_{t_{d-2}}))$.

To reduce the dimension of our cube, we start by identifying all its cube-faces. Then, we extract all the PCA-slices from the cube-faces. Finally, we apply iteratively PCA on each PCA-slice. This operation generates a set of coordinate factors for every single PCA-slice. In order to track the membership of a cell and its corresponding coordinate factors, we store the sets of coordinate factors in relational tables that we refer to as PCA-tables. Thus, each PCA-slice has its corresponding PCA-table and each cell of the original cube is associated with a set of PCA-tables corresponding to its containing PCA-slices. We introduce then the new notion of *PCA-cube* which associates each cell with its correspondent coordinate factors.

Definition 3 (PCA-cube). *A PCA-cube is a data cube which contains a new type of measures, consisting of references to the sets of the coordinate factors associated to original measure. These coordinate factors are stored in an external database.*

Thus, every single cell is associated with a number of predictor sets equal to the number of its containing cube-faces. Hereby, we note that cells belonging to the same PCA-slice and sharing the same attribute along their *variant* dimensions share the same factorial coordinates. This highlights the reduction aspect and contributes to the decrease of the training data size comparing to the original data cube. Moreover, we admit that this reduction stage is a time consuming process, which is just like most conventional OLAP pre-processing phases. Therefore, we believe that it should be executed in backstage on a regular basis by the end of each periodic data loading of the DW.

4.3 Prediction of Inexistent Measures

As mentioned above, our predictive model considers the factorial coordinates as predictors and the measures values as targets. In our case, every PCA-table corresponding to the same PCA-slice serves as an exclusive training sub-sets of its contained measures. To perform the learning process, a naive approach consists in merging all the training sub-sets into a unique large one.

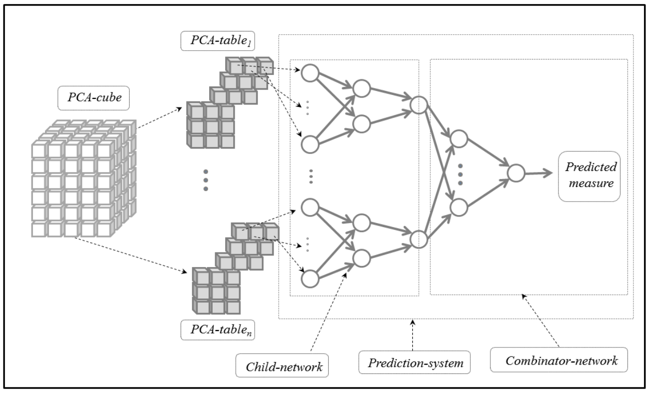


Fig. 1. Our proposed MLP architecture

However, this approach will damage the uniqueness of information provided by each PCA-cube, which represents the variation of the targeted measure over particular dimensions. Thereafter, we propose a novel architecture of feed-forward neural networks capable of separately consider a set of distinct training sub-sets without having to merge them. In such a way, we allow MLP to deal with the challenges of the multidimensional structure. Therefore, we empower the impact of the unique contributions of each network. Therefore, we allow MLP to be embedded into a multidimensional environment.

As shows in Figure 1, our novel neural networks architecture is based on an interconnection of a multiple children-networks. Each child-network is associated with one particular PCA-cube and considers the data referenced by its measure as predictors. In addition, each child-network has a single output neuron taking the values of the targeted classical measure during the learning process. The set of the children-networks output neurons are brought together as inputs of a second network that we will refer to as a combinator-network. It has a single output neuron that provides the predicted value of the inexistent measure at the end of the learning process. In such a way, we allow each PCA-cube to contribute in a unique way in the learning process by feeding separate child-networks and thus allow MLPs to deal with the multidimensional structure.

Let us note $N(D_k, D_v, (D_{s_1}, \ldots, D_{s_i}, \ldots, D_{s_{d-2}}))$ the child-network associated to the cube-face $Cf(D_c k D_v, (D_{s_1}, \ldots, D_{s_i}, \ldots, D_{s_{d-2}}))$.

For each sub-network of our prediction system, we restrict the MLP architecture to three layer networks, including a hidden one, as several theoretical and empirical studies showed that a single hidden layer is sufficient to achieve a satisfactory approximation of any nonlinear function [16]. We also use the gradient back-propagation algorithm [17] that has proved its usefulness in several applications [7,8]. We associate it with the conjugate gradient learning method and the sigmoid activation function.

Algorithm 1. Training algorithm

```
Input: $\mathcal{P}cc, \mathcal{A}[], RMSE\_min$
Output: $child, combinator, RMSE$
foreach $child$ do
    child $\leftarrow initialize(child)$;
    $converged \leftarrow false$ while $converged = false$ do
        while $A[] \neq \emptyset$ do
            $m \leftarrow extract\_measure(A[], \mathcal{P}cc)$;
            $cp \leftarrow extract\_compoment(m)$;
            $propagte(cp, child)$;
            $adjust(child)$;
        if $RMSE(child[]) < RMSE\_min$ then
            $converged \leftarrow true$;
$combinator \leftarrow initialize(combinator)$;
$converged \leftarrow false$ while $converged = false$ do
    while $A[] \neq \emptyset$ do
        foreach $child$ do
            $m \leftarrow extract\_measure(A[], \mathcal{P}cc)$;
            $cp \leftarrow extract\_compoment(m)$;
            $combinator - input[] \leftarrow propagte(cp, child)$;
        $propagte(combinator - input[], combinator)$;
        $adjust(combinator)$;
    if $RMSE(combinator) < RMSE\_min$ then
        $converged \leftarrow true$;
        $RMSE \leftarrow RMSE(combinator)$;
        return $child[], combinator, RMSE$;
```

The learning process of the whole system is provided in Algorithm 1. It starts by the initialization of each child-network by setting-up its appropriate structure, according to its associated cube-faces. Then, each child-network, is trained individually using a randomly selected set of cells $\mathcal{A}$ from the PCA-Cube $\mathcal{P}cc$. As cited above, the cells of these sets contains, in addition to their conventional measures, a reference to their corresponding coordinate factors scores in external PCA-tables. Therefore, a PCA-table is extracted following the treated child-network. Then, following the treated cell, a specific record is extracted and injected in the child-network. After the training of all children-networks, the combinator-network is to be initialized and then trained with obtained outputs of the trained child-networks.

The main contribution of this novel architecture is that it involves different training sets in the same learning process without having to merge them and yet generates one unique prediction for each targeted value through the combinator-network. This fact outlines the contribution of each PCA-cube in the learning process and thus, highlights the particularity of the new OLAP oriented MLP architecture.

5 Experimentation

To validate our approach, we implemented an experimental prototype of our system. We exploited the database *American Community Surveys 2000-2003* [1], after adapting it to context of DW. It is a real-life database of the U.S.A census that concerns samples of the population treated between 2000 and 2003.

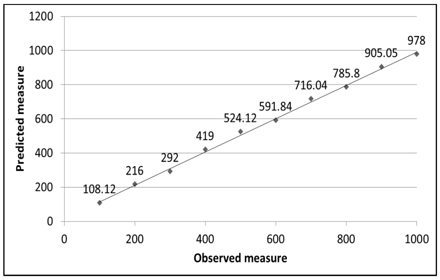


Fig. 2. Prediction quality

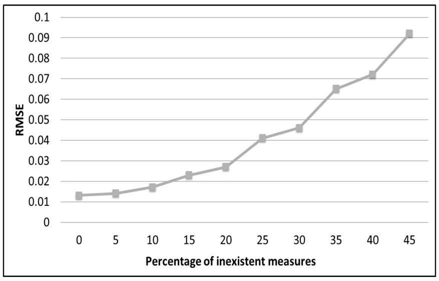


Fig. 3. Proposal performance

5.1 Analysis Context

We consider a 4 dimensions data cube; *Location*, *Origin*, *Education* and *Time*, with 3.8 million facts. The *Location* dimension contains the geographic data of the census. The *Origin* dimension contains information about the racial structure of the U.S.A population. The *Education* dimension contains information about the levels of education attained by the subjects of the census. As for the *Time* dimension, it provides the time information. We aim at predicting the number of people of a certain race, according to their cities and their levels of education in 2003. To do so, we focus on the *person count* measure and select the hierarchical

[1] American Community Surveys is accessible from the official site IPUMS-USA (Integrated Public Use Microdata Series); http://sda.berkeley.edu

levels; *State*, *Race* and *Education*. These levels include, respectively, 51, 10 and 14 dimension attributes. We start by applying our reduction approach, we end up with 6 PCA-cubes, which generate 10, 12, 4, 3, 4 and 3 principal components. For the prediction phase, we use the 10-fold cross-validation technique and the Root Mean Squared Error (RMSE) as a quality indicator. Our experimental study is spread over four experiments to meet the following aspects.

5.2 Prediction Quality

The objective of the first experiment is to investigate the performance of our predictive system in the case of real-life sparse data cube. To do so we elaborated a predictive system that faithfully represents our proposed architecture. We have set the hidden neurons number of each sub-network hidden layer, to one half of the number of its own input. Then we tried to predict a set of random measures that had been not included neither in the reduction nor in the learning process. To properly present the results, we considered measure values that are separated by regular intervals. We presented the resulting curve in the Figure 2. We note that the predicted values have minimum distances from the line *observed measure = predicted measure*. Furthermore, the correlation coefficient of these values is equal to 0.97, which shows a good quality prediction accuracy.

Table 2. Detailed performances of the the child-networks

	N(L,O,(E))	N(L,E,(O))	N(E,O,(L))	N(E,L,(O))	N(O,E,(L))	N(O,L,(E))	
RMSE_Train	0.390	0.045	0.690	0.032	0.098	0.035	0.022
RMSE_Test	0.482	0.068	0.630	0.061	0.183	0.027	0.025

5.3 Performance of Our Proposal

Even if resolving the data cubes sparsity is not part of our main objectives. We find it incessant to investigate the impact of sparsity level of data cubes on the performance of our system. Therefore, in the second experiment, we studied the prediction quality of our system with different percentages of the inexistent facts in the training set. In the resulting curve, presented in Figure 3, we notice that the performance of our proposal gradually decreases when data sparsity increases. Actually, RMSE values evolve in a monotonous manner between a percentage of 0% and 20% of inexistent facts. From 20%, changes in RMSE become important. This may be explained by the absence of a sufficient number of valid instances to support the learning process starting from this rate. Moreover, from a rate of 20%, the system starts to face serious difficulties in capturing the patterns of measures.

Table 3. Detailed performances of the different systems

	Nc	Nc_restricted	N_classical
RMSE_Train	0.022	0.032	0.050
RMSE_Test	0.025	0.039	0.071

5.4 Novel Architecture Contribution

In the third experiment, we investigate the performances of the distinct child-networks involved in our global system. For this purpose, we extracted the distinct values of RMSEs obtained from the different child-networks that form our system. We present these values in Table 2.

We observe that RMSEs values remarkably vary from of a child-network to another. This is justified by the particularity of the data structure. We find that the two child-networks, which provide the largest RMSEs thus the worst quality of prediction, are $N(E, O, (L))$ and $N(L, O, (E))$. We note that these two child-networks consider *Origin* as their *Variant* dimension. Here, we must keep in mind that in our proposal the number of available instances in a training set for a child-network is equal to the number of rows in its correspondent PCA-slice, which is equal to the number of dimension attribute of the *Variant* dimension of that PCA-slice. In our case, *Origin* dimension is the poorer dimension in term of attributes number (10 attributes). Subsequently, child-networks that consider it as their *Variant* dimension have the smallest number of training instance. Inversely, we found that $N(E, L, (O))$ and $N(O, L, (E))$, which consider *Location* as its *Variant* dimension, produces the smallest RMSE among all child-networks. We note that the more instances of learning a system have, the better performance it will provide.

Although these results are obtained in order to detail the performance of our system components, only the results produced by the combiner-network Nc presented in Table 3, which predict the final value, reflect the quality of our global system. Interestingly, the combinator-network surpasses all the child-networks in training and in test phases. Thus, we conclude that combinator-network merge the information provided by the child-networks to provide conclusive values.

Following these results, a logical question arises: How would the system perform if we eliminate the child-networks that which produces the worst results from the analysis? To answer this question, we trained the network Nc_restricted, in which we eliminated $N(E, O, (L))$ and $N(L, O, (E)$ from the analysis. We found that the performance of the global system deteriorates comparing to Nc, as is shown in Table 3. Actually, irrelevant information provided by the two eliminated child-networks, become relevant for the global system. In other words, the values predicted from a cube-face, serves to refine the quality of the global system even if they are not usable in individual manner.

In order to further investigate the contribution of our system. In the fourth experiment, we trained N_classical, which is a classical architecture MLP. This network receives all the coordinate factor associated to a particular cell at once as inputs, in a classical manner, without using the concept of child-network. In other words, this architecture merges all the training sub-sets into a unique large one. This system, provided in Table 3, provided the worst performances among all the studied architectures. This is due to the incapacity of a classical system to capture the measure pattern over the multidimensional architecture even if it receives decorrelated data. This fact highlights further the efficiency of our novel architecture.

6 Conclusion and Perspectives

In this paper, we encouraged the integration of machine learning techniques OLAP environment. The key idea of our proposal is that these sophisticated techniques can be used even with the constraints raised by the important dimensionality and volumetry of data cubes. However, we highlighted that some pre-processing steps have to be considered. In our study, we proposed to feed an MLP network with several sets of reduced data that are generated over the original measures. Nevertheless, these reduced sets have been extracted in a particular manner that preserved their fundamental distributions and semantic proprieties. In order to enhance the unique contribution of each training set, we proposed a novel MLP architecture that involves separate data sets in the same learning process and still provide conclusive values for the targeted measures.

The carried out experimental study showed good prediction performance. Furthermore, it helped to get further insights for the different results obtained over the sub-systems that form our global model.

In future work, we plan to include a framework that explains the reasons of inexistent measures occurrences, similarly to that of [18], which is performed on classical two-dimensional data. We also would like to involve the hierarchical structure of data cubes in our system. This way we could exploit the different levels of aggregation to predict lower/higher level facts. Finally, we believe that modeling a theoretical relation between the reduction and the prediction system can be very useful to optimize our model.

References

1. Inmon, W.H.: Building the Data Warehouse. John Wiley & Sons (1996)
2. Agarwal, D., Chen, B.C.: Latent OLAP: Data cubes over latent variables. In: Proceedings of the 2011 International Conference on Management of Data, SIGMOD 2011, pp. 877–888. ACM, New York (2011)
3. Codd, E.F., Codd, S.B., Salley, C.T.: Providing OLAP (on-line Analytical Processing) to User-analysts: An IT Mandate, vol. 32. Codd & Date, Inc. (1993)
4. Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-Driven Exploration of OLAP Data Cubes. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 168–182. Springer, Heidelberg (1998)

5. BenMessaoud, R., Loudcher-Rabaseda, S.: OLEMAR: an On-Line Environment for Mining Association Rules in Multidimensional Data. Advances in Data Warehousing and Mining, vol. 2. Idea Group Publishing (2007)
6. Bodin-Niemczuk, A., Messaoud, R.B., Rabaséda, S.L., Boussaid, O.: Vers l'intégration de la prédiction dans les cubes OLAP. In: EGC, pp. 203–204 (2008)
7. Haykin, S.: Neural Networks: a Comprehensive Foundation. Prentice Hall International Editions Series. Prentice-Hall (1999)
8. Adya, M., Collopy, F.: How Effective are Neural Networks at Forecasting and Prediction? a Review and evaluation. Journal of Forecasting 17(5-6)
9. Palpanas, T., Koudas, N., Mendelzon, A.: Using Datacube Aggregates for Approximate Querying and Deviation Detection. IEEE Trans. on Knowl. and Data Eng. 17, 1465–1477 (2005)
10. Bishop, C.: Neural Networks For Pattern Recognition. Oxford University Press (1995)
11. Chen, B.C., Chen, L., Lin, Y., Ramakrishnan, R.: Prediction Cubes. In: Proceedings of the 31st International Conference on Very large Data Bases, VLDB 2005, pp. 982–993 (2005)
12. Chen, Y., Dong, G., Han, J., Pei, J., Wah, B.W., Wang, J.: Regression Cubes with Lossless Compression and Aggregation. IEEE Trans. on Knowl. and Data Eng. 18
13. Hotelling, H.: Analysis of a Complex of Statistical Variables into Principal Components. Journal of Educational Psychology 24(7), 498–520 (1933)
14. Tshilidzi, M.: Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques. Information Science Reference - Imprint of: IGI Publishing, Hershey (2009)
15. Wang, Z., Xu, J., Lu, F., Zhang, Y.: Using the Method Combining PCA with BP Neural Network to Predict Water Demand for Urban Development. In: Proceedings of the 2009 Fifth International Conference on Natural Computation, vol. 2, pp. 621–625. IEEE Computer Society, Washington, DC (2009)
16. Hornik, K., Stinchcombe, M., White, H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2(5), 359–366 (1989)
17. Rumelhart, D.E., Mcclelland, J.L.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations (Parallel Distributed Processing) (August 1986)
18. Ben Othman, L., Ben Yahia, S.: Yet Another Approach for Completing Missing Values. In: Yahia, S.B., Nguifo, E.M., Belohlavek, R. (eds.) CLA 2006. LNCS (LNAI), vol. 4923, pp. 155–169. Springer, Heidelberg (2008)

Mobile Activity Recognition Using Ubiquitous Data Stream Mining

João Bártolo Gomes[1], Shonali Krishnaswamy[1], Mohamed M. Gaber[2], Pedro A.C. Sousa[3], and Ernestina Menasalvas[4]

[1] Institute for Infocomm Research (I2R), A*STAR, Singapore
{bartologjp,spkrishna}@i2r.a-star.edu.sg
[2] School of Computing, University of Portsmouth, United Kingdom
mohamed.gaber@port.ac.uk
[3] Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal
pas@fct.unl.pt
[4] Facultad de Informatica, Universidad Politecnica Madrid, Spain
emenasalvas@fi.upm.es

Abstract. Mobile activity recognition focuses on inferring the current activities of a mobile user by leveraging the rich sensory data that is available on today's smart phones and other wearable sensors. The state of the art in mobile activity recognition research has focused on traditional classification learning techniques. In this paper, we propose the Mobile Activity Recognition System (MARS) where for the first time the classifier is built on-board the mobile device itself through ubiquitous data stream mining in an incremental manner. The advantages of on-board data stream mining for mobile activity recognition are: i) personalisation of models built to individual users; ii) increased privacy as the data is not sent to an external site; iii) adaptation of the model as the user's activity profile changes. In our extensive experimental results using a recent benchmarking activity recognition dataset, we show that MARS can achieve similar accuracy when compared with traditional classifiers for activity recognition, while at the same time being scalable and efficient in terms of the mobile device resources consumption. MARS has been implemented on the Android platform for empirical evaluation.

1 Introduction

The integration of small wireless sensors into objects of everyday life allows to create a non-intrusive sensory data rich environment. Thus, the miniaturisation and cost reduction of sensor hardware and mobile devices has led to the emergence of research into mobile AR [1]. In several existing studies wearable sensors are used by people while performing their daily activities [1,6], while others additionally use sensors embedded into tools and utensils in an apartment, which allows the analysis of more fine grained activities [8].

Mobile AR is usually formulated as a classification problem, where supervised machine learning is used to interpret sensed data into activities [1,7]. The learning process normally goes through the following stages: i) data collection, where sensor data is collected over a specified period of time from one or more mobile users, with the users typically labelling/annotating their activities; ii) data transfer, where the collected data is transferred to and collated in a centralised repository; iii) learning/model building, where the AR classification model is trained and tested using the collected data; iv) model deployment, where the learnt model is deployed on-board the mobile device for identifying and classifying activities from sensory data. These state of the art mobile AR approaches from ubiquitous sensors have been shown to achieve high recognition rates [7]. This may give the impression that the general problem of AR has been solved successfully. However, in existing approaches the obtained models are static, are built off-line in an external (to the mobile device) environment and little attention is given to issues such as personalisation of generic models and privacy.

To address these issues in this paper, we propose the Mobile Activity Recognition System (MARS) that learns the classification model on-board the mobile device itself through ubiquitous data stream mining in an incremental manner. The proposed system (MARS) has been implemented on the Android platform to evaluate its feasibility.

The rest of the paper is organised as follows. The following Section reviews the related work. Section 3 presents the definition of mobile AR as classification problem, which is followed by a detailed description of the existing open challenges in Section 4. The proposed Mobile Activity Recognition System (MARS) is presented Section 5. The experimental setup and results are discussed in Section 6. Finally, in Section 7, conclusions of this work and future work are presented.

2 Related Work

AR from sensor data is a popular research field that has contributed with several high recognition rate approaches. Many of these use supervised machine learning algorithms, such as Decision Trees [1], Artificial Neural Networks, Hidden Markov Models, Naive Bayes, K-Nearest Neighbour or Support Vector Machines. For an extensive review of supervised learning approaches for AR please refer to [7]. Here we focus our review on works that perform mobile AR from sensor data.

One of the most cited publications on activity recognition in pervasive computing [1] deployed five small biaxial accelerometers worn simultaneously on different body positions in order to distinguish 20 activities of interest. The data was collected from 20 subjects that annotated it themselves without researcher supervision or observation. From the learning algorithms tested, C4.5 decision trees showed the best performance with an overall accuracy rate of 84%. Such technique is considered to be slow to train but quick to run. Therefore, the authors suggest that a pre-trained decision tree should be able to recognise user activities in real-time on a 2004 top-end mobile device. Moreover, it is reported that some activities are recognised with subject-independent training data while others seem to require subject-specific training data.

In a recent paper, [6] proposes and experimentally evaluates a system that uses phone-based accelerometers to perform mobile AR. Data was collected from

29 subjects as they performed their daily activities such as walking, jogging, climbing stairs, sitting, and standing. This works shows how a smart phone (Android) can be used to perform activity recognition, simply by carrying it in a fixed position (front pants leg pocket). The results show that most activities are recognised correctly over 90% of the time. Still, the collected data is transferred to an Internet-based server where a static model is generated off-line. Again the issues of personalisation or privacy are not addressed but in the future work section it is mentioned that an improvement of the proposed system would be to generate the model on-board. Nevertheless, to the best of our knowledge such improvement has not yet been proposed.

The reviewed approaches built static classification models off-line in an external (to the mobile device) environment. Moreover, the streaming nature of data is not taken into account nor the possibility that the model needs to be adapted over time. In addition, little attention has been given to the personalisation of the built model to suit a particular user, despite the results that seem to indicate that better accuracy is obtained with personalised models (i.e., training and test data from the same subject). To the best of our knowledge, no other ubiquitous data stream mining approach has been proposed so far to address mobile AR.

For a more extensive review of data stream mining systems that have been used successfully in other applications please refer to [5]. A demo of MARS has been recently presented in [2].

3 Problem Definition

Let X be the space of features that correspond to the available input sensor features and Y be the set of possible (discrete) class labels that correspond to the activities of interest. Consider a data stream DS, where $X_i = (\boldsymbol{x_i}\ , y_i)$ with $x_i \in X$ and $y_i \in Y$, represents the i^{th} record in DS. The modelling of AR is formulated as a function f that assigns each sensor feature input record $\boldsymbol{x_i}$ to the true activity label y_i. This function f can be approximated using supervised learning by training a model m. The goal is that the trained model m minimises the number of wrongly recognised activities (i.e., achieves high accuracy).

4 Open Challenges

Despite the good results of existing supervised learning approaches in AR, there are still open challenges that to the best of our knowledge have not been addressed. The following subsections introduce such challenges.

4.1 Training Data

The usual supervised learning approach to AR assumes that there is abundant training data and that the function f to model is static. However, in realistic situations, f is usually subject dependent and can even change over time within subject. Moreover, past work shows that if the training data is collected from the subject of interest then there is no advantage to use additional training data from other subjects [1,7,6]. Still, in the case where training data from the subject of interest is not available, having data from higher number of subjects is beneficial to the resulting recognition accuracy.

4.2 Model Generation

In most existing supervised learning approaches to mobile AR, the training data is collected, a classification model is generated offline from the collected data, and finally the obtained model is deployed. Nevertheless, there are disadvantages that can result from using this type of offline learning process:

– The obtained model is static - Once a model is generated it cannot incorporate new information.
– Computational costs - The batch algorithms typically used to generate the model are not designed to be executed in mobile devices. Such algorithms usually require several passes over the dataset and require that the entire dataset is allocated into main memory. In contrast, ubiquitous data stream mining approaches process each record only once and are memory efficient [5].
– Accuracy assessment in a realistic scenario - The static model that is deployed can have good accuracy on a testing set which is usually similar to the training set, when methods such as cross validation or a hold-out set are used. However, the performance of this model in a realistic situation depends on how the test set is representative of the usage scenario.

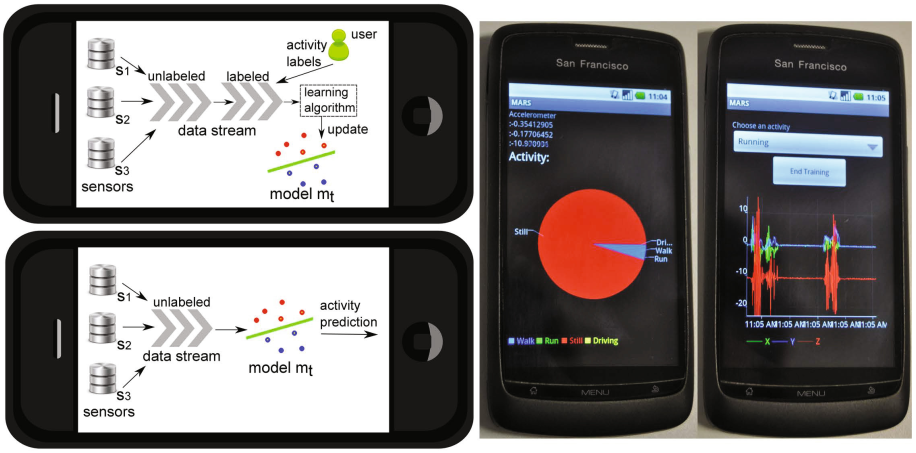


Fig. 1. MARS: framework and implementation

5 MARS: Mobile Activity Recognition System

This paper proposes MARS, a ubiquitous data stream mining approach to mobile AR. Such approach is motivated and focused on addressing the open challenges described in the previous section. Conversely to traditional supervised learning, data stream classification algorithms are able to update an anytime model m_t as new training records are available in the stream. Moreover, these algorithms are light-weight and can be executed using the computational resources usually available on nowadays mobile devices. The proposed approach enables greater personalisation and privacy while bringing the whole learning process on-board the mobile device.

The learning process is divided into two phases:

– Training - During the training phase the user performs the activities of interest, either in predefined drills or freely during normal activities and annotates interactively the data collected from the sensors using a user-friendly interface (i.e., usually simply by selecting from a list the activity that he previously executed). This type of naturalistic data collection has been successfully used before, however, the records are saved to be then processed by a offline learning algorithm, while we propose that the annotated data stream should be processed on-board by an incremental learning algorithm. Moreover, since an anytime model m_t (i.e., model at time t) is assumed, it is possible to estimate the accuracy of this model as new records are incorporated. For this purpose we propose that the prequential statistic is used [4]. Figure 1 illustrates MARS training phase, where the data (unlabelled) is coming from the available sensors, then the user annotates/labels such data which is processed by the learning algorithm that updates the anytime model m_t.
– Activity Recognition - the new records (unlabelled) to be classified are given to the anytime model m_t that returns the predicted activity. This phase is also illustrated in Figure 1.

6 Experiments

This section describes the experiments that were performed to evaluate the MARS feasibility and accuracy. The data used in the experiments has been released for the OPPORTUNITY AR challenge[1], which aims to provide a comparative benchmark dataset for AR approaches.

6.1 The AR Challenge Data

The lack of established benchmarking problems for AR is one of the motivations behind the OPPORTUNITY AR challenge [8]. The data contains daily human activities recorded in a sensor rich environment: a room simulating a studio

[1] http://www.opportunity-project.eu/challenge

flat with kitchen, deckchair, and outdoor access where subjects performed daily morning activities. Two types of recording sessions were performed: Drill sessions where the subject performs sequentially a pre-defined set of activities and "daily living activities" runs (ADL) where the subject executes a high level task (wake up, groom, prepare breakfast, clean) with more freedom about the sequence of individual atomic activities. It records 72 sensors of 10 modalities, integrated in the environment, in objects, and on the body. It consists of an annotated dataset of complex, interleaved and hierarchical naturalistic activities, with a particularly large number of atomic activities (around 30.000), collected in a rich sensor environment. Data was manually labelled during the recording and later reviewed by at least two different persons based on the video recording.

The data used for the challenge is composed of the recordings for 4 subjects. For each subject there are 5 unsegmented sessions. During the challenge, for 3 of the 4 subjects the last two session were used by the organisers to evaluate the performance of the contributed methods. Moreover, the subject number 4 is used to assess robustness to noise, as rotational and additive noise has be added to the test data (last two sessions) of this subject.

The challenges consisted of 4 challenges, but for the purpose of this work evaluation we will consider the multimodal classification tasks that are:

- Modes of locomotion (Task A) - The goal of this task is to classify the subject mode of locomotion (i.e., stand, walk, sit, lie) from body-worn sensors.
- Gestures (Task B2) - This task concerns recognition of right-arm gestures (17 classes) performed in a daily activities scenario. Gestures include, clean the table, open/close a door/fridge/dishwasher/drawer and toggle a switch.

For the experiments we followed the evaluation proposed in the challenge, that is, the last two sessions were used for testing and the Drill session plus the first 3 ADL sessions were used for training. Note that we are also using data from subject 4 that contains noise in the test set (last two sessions). Therefore, we can also asses how the approach deals with noise, which is somehow similar to the challenge Task C which used data from subject 4, instead of subjects 2 and 3, to asses the accuracy for the Gestures classification task.

6.2 Implementation

To evaluate the feasibility of the MARS, we implemented a prototype on the Android platform. The experiments where carried out on a low-end Android phone, ZTE Blade, sold in UK as Orange San Francisco, that in early 2011 was one of the budget Android phones on the market. The phone has a Qualcomm MSM7227 600 MHz processor, 512MB of RAM, 1250 mAh battery and runs Android 2.2 Froyo.

The learning algorithms used in the experiments (i.e., the incremental Naive Bayes and C4.5 decision tree) are available in the WEKA Since these algorithms are developed in Java it was easy to port them to the Android application. However, during our preliminary tests we noted it was not even possible to execute the decision tree algorithm (implemented in WEKA as J48) without getting a memory exception (java.lang.OutOfMemoryError), as this algorithm requires that all the dataset is loaded into memory. The max heap size for an Android application depends on the device but if the application is supposed to run on any device the memory allocation should be kept under 16MB. Since each AR challenge session file ranges from 10MB to 33 MB and the training set for each subject contains 4 sessions. Consequently, we had to perform the accuracy measures and running time experiments for the decision tree algorithm on a laptop computer with a 2.10 GHz Intel Core 2 Duo processor and 4GB of RAM memory.

The incremental Naive Bayes approach executed on the device without problems. The data files are read sequentially as a stream to incrementally train the classification model. Then this model is used to sequentially classify the session files that belong to the test set. During this process the only memory consumed is the one required to keep the anytime model and to read the record to be processed, that is then subsequently freed from memory.

The learning algorithm is executed in a dedicated thread that can run in background as a service and therefore the anytime classification model can be updated or asked for prediction at anytime.

6.3 Incremental vs Traditional

Here we report the experiments where we compared in terms of accuracy and running time the incremental Naive Bayes (NB) approach and the J48 Decision Tree (DT). We decided to compare the proposed approach with DT because this algorithm have been shown high accuracy in AR problems and was the first ranked algorithm on the locomotion predictive task of the OPPORTUNITY AR challenge (task A).

Table 1 summarises the results for the different subjects (rows) on the two classification tasks (i.e., locomotion and gestures) that are shown on the respective column. The first thing noticed is that for both algorithms there is higher accuracy on the locomotion task than in the gestures task. This is justified by the fact that the gestures task is more demanding, that is, has more classes (17 instead of 4) and the available sessions have less annotated data for this task than the locomotion one. Nevertheless, these results are within what has been reported in the literature for state of the art approaches in AR [7] and what has been reported recently for the OPPORTUNITY AR challenge [8].

Looking at the accuracy results between the NB vs DT, we observe that for the locomotion task the mean accuracy for the NB is 86,4% ± 3.9 while it is slightly lower for the DT with 85,9% ± 2,5. For subjects S1 and S2 the incremental NB even achieves higher accuracy than the DT. When the gestures task is considered, the mean accuracy for the NB is 56,1% ± 5,4 and the DT obtains 60,9% ± 7,1. Still, the simple incremental NB approach achieves similar accuracy, particularly for subject S3 where only 1.1% difference exists. Moreover, subject 4 achieves the lowest accuracy on both tasks, due to the presence of noise, which seems to have a minor impact on the accuracy.

Table 1. Accuracy results of NB vs DT

	Locomotion		Gestures	
	NB	DT	NB	DT
S1	91.5%	88.8%	61.7%	69.1%
S2	87.4%	86.8%	53.3%	54.4%
S3	84.4%	85.5%	59.4%	64.4%
S4	82.3%	82.7%	50.0%	55.7%

Table 2. Training times of NB vs DT

	Locomotion		Gestures	
	NB	DT	NB	DT
S1	21.91s	664.09s	6.48s	127.28s
S2	18.93s	564.03s	6.11s	151.62s
S3	23.67s	793.60s	6.56s	119.46s
S4	16.78s	546.70s	4.65s	100.82s

Table 2 summarises the training times for obtaining the models for the different subjects (rows) and tasks/learning algorithms in columns. The training times, show that on both tasks the NB is much faster than the DT. Note that both measures of the training time were performed on a laptop computer as it was not possible to run the DT on the Android device. The results show that NB only takes a few seconds to build a model while the DT algorithm takes on average about 10 minutes for the locomotion task and 2 minutes for the gestures task. The greater training time for the locomotion task results from the larger number of training records in this task.

6.4 Accuracy over Time

To further analyse the accuracy results of NB and since we are proposing an incremental approach, we decided to measure how the accuracy changes over time as more records are processed.

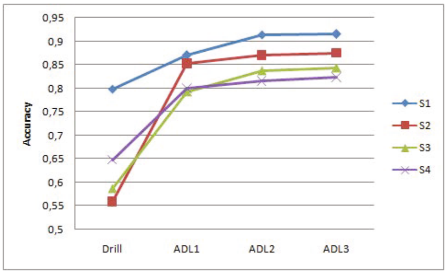


Fig. 2. Accuracy for the Locomotion task

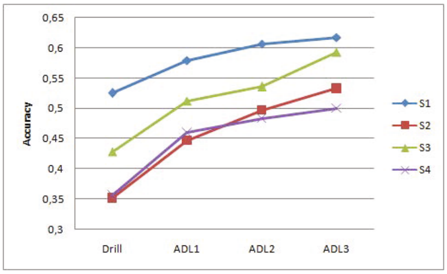


Fig. 3. Accuracy for the Gestures task

Figure 2 shows for the different subjects the accuracy for the locomotion task anytime model after training is performed with data from the different sessions (on the horizontal axis) in an incremental way. The accuracy curves show that the accuracy increases as more data is processed. The results are similar across the subjects, however, it can be observed that the Drill session for subject one is enough to achieve accuracy in the order or 80% while for the other subjects at this stage accuracies of around 60% are obtained. Moreover, the gain in accuracy from an additional session decreases with the number of sessions as can be observed in

Figure 2. This is an interesting result because it can be used to determine when a stable accuracy value has been obtained and no further training is needed.

Figure 3 shows for the different subjects the accuracy for the gestures task anytime model. Again the accuracy curves show that the accuracy increases as more data is processed. However, for this task, which is more demanding than the locomotion one, for the reasons that have been mentioned previously, the accuracy grows more slowly as more records are observed. We should note again that each session has less training data for the gestures task than the locomotion one, which can itself have an impact on the accuracy growth rate with additional sessions. For this task the learning curves are even more similar among the subjects.

6.5 Memory Consumed by the Model

Since memory is a critical resource in mobile environments, we measured the memory size consumed by the classification models.

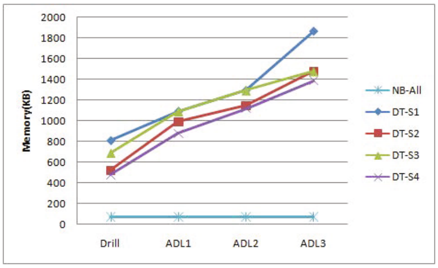


Fig. 4. Memory user in the Locomotion task

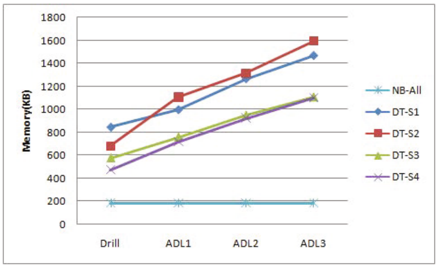


Fig. 5. Memory used in the Gestures task

Figure 4 shows the memory consumption of the incremental NB and DT with the number of training sessions for the locomotion task. Since DT is a non-incremental approach the algorithm was run again with the sessions desired. For instance the results for ADL3 of a particular subject represent a classification model that was built using a training set that contains session ADL3 and the all the previous sessions (i.e., Drill, ADL1 and ADL2). The results show that the NB algorithm resulting model achieves the lowest memory consumption (71KB) and that this value is independent of the number of training records. This is the consequence of the model representation of the NB which only requires to store estimators for the marginal and conditional distributions, which once built stays the same over the entire learning process (possible infinite number of training records). In contrast, the decision tree structure size in memory depends on the tree itself. In Figure 4 we observe that this value often increases linearly with the number of training records.

Figure 5 shows results similar to the Figure 4. For the gestures task the NB model requires again the smallest memory size (185KB). This size is larger than the locomotion model, as this task has more classes, and therefore more estimators need to be stored. For the DT we observe that the trees are of similar size than the ones used to model the locomotion task as the number of classes does not influence the model size directly.

In general these results show that both approaches achieve model sizes that are very small when compared to nowadays mobile devices memory capabilities. However, since the DT requires that the training set must be stored in memory in order to built the model, its usability for on-board modelling is compromised. Consequently, for this reason the majority of studies that use DT for mobile AR require that the model is built on an external server that is later deployed on the mobile for classification of the activities.

6.6 Battery Consumed by the Process

In this experiment the training process was programmed to be in an infinite loop and was left running in background. In addition to the battery level we measured the processing rate, that is, how many records per second the algorithm processes. This way we can analyse how the processing rate influences the battery consumption and assess the feasibility of the approach. Please note that there a battery cost associated with the data collection from sensors while here we are accessing the file system. Nevertheless, we are controlling our experiment for the impact that the learning algorithm has on the battery, since the cost associated with collecting data from the sensors will be independent of the learning method and will be an existing factor to consider in every approach.

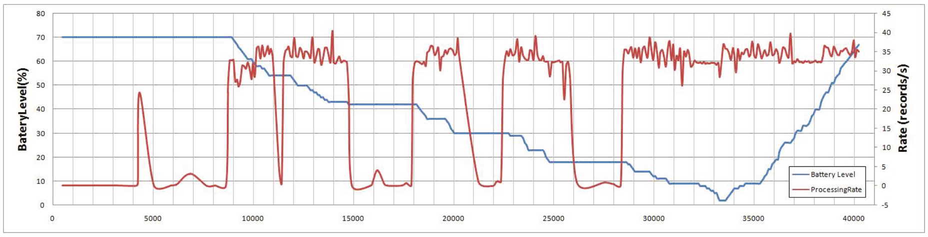


Fig. 6. Battery consumption and processing rate over time

Figure 6 shows the battery consumption and processing rate over a period of about 12hours. The battery level starts at 70% and it decreases slowly over time until the phone is plugged-in at around time 33569s. It can also be seen that the battery consumption rate is related to the processing rate.

The processing rate can be as low as 1 record for every 10 seconds to 40 records per second. This is well within what is required in mobile AR. For instance in [6] where 10-second intervals are considered for basic locomotion activities or in the more demanding case of the OPPORTUNITY AR challenge dataset [8] used in our experiments, where for the gestures task the shortest gesture found in the dataset lasts about 0.5 seconds. It is considered that such rate is enough to achieve high speed of recognition without missing activities. We should note that the dataset used in the experiments contains a large number of attributes (i.e., 114 numeric attributes + 2 discrete class attributes) than what is normal and no feature selection or tuning to increase the efficiency of our proposed approach. This way we intend to demonstrate the feasibility of the approach in a highly demanding learning task.

A possible way to save battery would be to adapt the processing rate to the situation (i.e., resources and context) [3] to address the mobile application AR requirements.

6.7 Adaptation to Different Subjects

In MARS, there is greater opportunity for personalisation of the model as the training data is subject specific. Nevertheless, in some applications, for instance in elderly monitoring, asking the subjects to annotate their activities can become an issue if they suffer from Alzheimer or other memory related condition.

In this set of experiments we tested how a model built using Naive Bayes with data from other subjects can be used to accurately classify the activities of a particular subject.

The results indicate (tables not included due to space limitations) that in general for subjects 1, 2 and 3 the models are able to still achieve good accuracy (in both tasks) in relation to the scenario where subject-specific training data is used. Moreover, we observe that for subjects 2 and 3, when the training data from subject 1 is used better accuracy than with the subject-specific data is obtained. We can note that this can be attributed to the fact that the training set for subject 1 has more training records than the other two. Nevertheless, for subject 4 the adaptation is poor, either when its data is used for training of testing. The bad performance with subject can be attributed to the noise that its test dataset includes or maybe to a very different user profile. To further investigate this issue we performed the same experiments without using training data of subject 4 and this results in higher accuracy values.

7 Conclusions and Future Work

In this paper we proposed and experimentally evaluated the Mobile Activity Recognition System (MARS), that is an ubiquitous data stream mining approach to mobile activity recognition. In MARS, a data stream classification algorithm (incremental Naive Bayes) is used to update an anytime model from a stream of ubiquitous sensor data. The main contribution of this work is to show the feasibility to execute such integrated learning approach to mobile AR on the mobile device itself. The advantages of on-board mobile data stream mining are higher personalisation of the AR models (that are built based on individual users annotated data), increased privacy as the data is not sent to an external site, and also using an adaptive anytime model instead of a static model enables adaptation of the activity profile to changes.

In future work, in line with the latest experiments conducted in this work we plan to study how to depend less on labelled/annotated data, for instance by using a semi-supervised or active learning approach to AR.

Acknowledgements. We would like to thank Dr. Daniel Roggen for the release of the OPPORTUNITY AR challenge dataset labels promptly after the challenge results were presented at the SMC2011 conference workshop.This research is partially financed by project TIN2008-05924 of Spanish Ministry of Science and Innovation. We would also like to thank to the FCT project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA-EIA/98355/2008).

References

1. Bao, L., Intille, S.S.: Activity Recognition from User-Annotated Acceleration Data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
2. Gomes, J.B., Krishnaswamy, S., Gaber, M., Sousa, P.A.C., Menasalvas, E.: Mars: a personalised mobile activity recognition system. In: Proceedings of the International Conference on Mobile Data Management, MDM 2012, Bengaluru, India, July 23-26, IEEE (2012)
3. Bartolo Gomes, J., Menasalvas, E., Sousa, P.A.C.: Situation-aware data stream mining service for ubiquitous applications. In: 2010 Eleventh International Conference on Mobile Data Management (MDM), pp. 360–365. IEEE (2010)
4. Gama, J., Sebastiao, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 329–338. ACM, New York (2009)
5. Krishnaswamy, S., Gama, J., Gaber, M.M.: Advances in data stream mining for mobile and ubiquitous environments. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 2607–2608. ACM (2011)
6. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SIGKDD Explorations Newsletter 12(2), 74–82 (2011)
7. Preece, S., Goulermas, J., Kenney, L., Howard, D., Meijer, K., Crompton, R.: Activity identification using body-mounted sensors – a review of classification techniques. Physiological Measurement 30(4), R1–R33 (2009)
8. H. Sagha, S.T. Digumarti, J.d.R. Millán, R. Chavarriaga Lozano, A. Calatroni, D. Roggen, G. Tröster: Benchmarking classification techniques using the Opportunity human activity dataset. In: IEEE International Conference on Systems, Man, and Cybernetics (2011)

Warehousing Manufacturing Data

A Holistic Process Warehouse for Advanced Manufacturing Analytics

Christoph Gröger, Johannes Schlaudraff,
Florian Niedermann, and Bernhard Mitschang

Institute of Parallel and Distributed Systems
University of Stuttgart
Universitätsstrasse 38, 70569 Stuttgart, Germany
{christoph.groeger,johannes.schlaudraff,florian.niedermann,
bernhard.mitschang}@ipvs.uni-stuttgart.de

Abstract. Strong competition in the manufacturing industry makes efficient and effective manufacturing processes a critical success factor. However, existing warehousing and analytics approaches in manufacturing are coined by substantial shortcomings, significantly preventing comprehensive process improvement. Especially, they miss a holistic data base integrating operational and process data, e. g., from Manufacturing Execution and Enterprise Resource Planning systems. To address this challenge, we introduce the Manufacturing Warehouse, a concept for a holistic manufacturing-specific process warehouse as central part of the overall Advanced Manufacturing Analytics Platform. We define a manufacturing process meta model and deduce a universal warehouse model. In addition, we develop a procedure for its instantiation and the integration of concrete source data. Finally, we describe a first proof of concept based on a prototypical implementation.

Keywords: Data Warehouse, Manufacturing, Process Optimization, Analytics, Business Intelligence, Data Integration.

1 Introduction

1.1 Motivation

The manufacturing industry is faced with strong global competition. Apart from product quality and pricing, flexibility, short lead times and a high adherence to delivery dates have become critical success factors [1]. Efficient, effective and continuously improved manufacturing processes thus play a central role in a comprehensive competitive strategy [2].

Both in research and in industry, Business Intelligence (BI) technology is recognized as an enabler for analytics-based optimization of business activities as well as decision support. It has repeatedly proven its potential in the service industry for the improvement of workflow-based business processes [3], [4].

With respect to BI approaches in manufacturing, we currently know of essentially two types wide-spread in industry: Pre-packaged dashboard applications, typically part of Manufacturing Execution systems (MES), based on standardized metrics with simple reporting functions [5] as well as custom BI applications, partly built on data warehouses, mainly focusing on spreadsheet-based Online Analytical Processing (OLAP) and reporting functions [6].

Several decisive insufficiencies of these approaches significantly prevent comprehensive process optimization in manufacturing: Most importantly, they miss a holistic process view integrating operational data and process data, e. g., from MES and Enterprise Resource Planning (ERP) systems, to find a broader range of optimization opportunities. Moreover, they do not make use of advanced analytics techniques, esp. data mining, to automatically identify hidden data patterns for process improvement. To overcome these deficiencies, in our overall work we develop the Advanced Manufacturing Analytics (AdMA) Platform as a novel approach for data-driven manufacturing process optimization. In this article, we focus on the central component of the AdMA Platform, the Manufacturing Warehouse, a manufacturing-specific holistic process warehouse.

The remainder is organized as follows: First, we introduce the AdMA Platform and structure related work for process warehousing and data integration in Section 2. Next, we define analytic requirements and potential data sources for the Manufacturing Warehouse in Section 3. Based on a manufacturing process meta model, described in Section 4, we develop a standardized warehouse model as well as a procedure for its instantiation and the integration of concrete data in Section 5. Our prototypical implementation and a first proof of concept are presented in Section 6. We conclude in Section 7 and highlight future work.

1.2 The Advanced Manufacturing Analytics Platform

The Advanced Manufacturing Analytics Platform, introduced in [7], is an integrated BI platform for manufacturing process optimization. It is based on a transfer of concepts of the Deep Business Optimization Platform [3], [8-10] to manufacturing with its conceptual architecture comprising the following components (see Fig. 1).

The *manufacturing process* is typically deployed on an MES and corresponding execution data is generated during process execution. Process data and additional operational data are integrated in the *Manufacturing Warehouse*, the focal point of this article. It is provisioned by the *Manufacturing Data Integrator,* which matches process and operational data. In general, operational data is subject-oriented and represents data of traditional data warehouses, e. g., financial data. Process data is flow-oriented and comprises execution data, i. e., events recorded during process execution, and process model data [9].

Process Analytics comprises different analysis methods, esp. data mining techniques and metrics calculation, to generate insights with the *Manufacturing Insight Repository* serving as a central component for the sharing of analysis results.

Indication-based Manufacturing Optimization uses pre-configured data mining models to explain and predict process characteristics.

Pattern-based Manufacturing Optimization goes beyond that and proposes concrete process modifications using optimization patterns. Both focus on the overall manufacturing process from the creation of the production order until the finishing of the product including all process steps and resources. They are further detailed and differentiated from existing data mining approaches in manufacturing in [11].

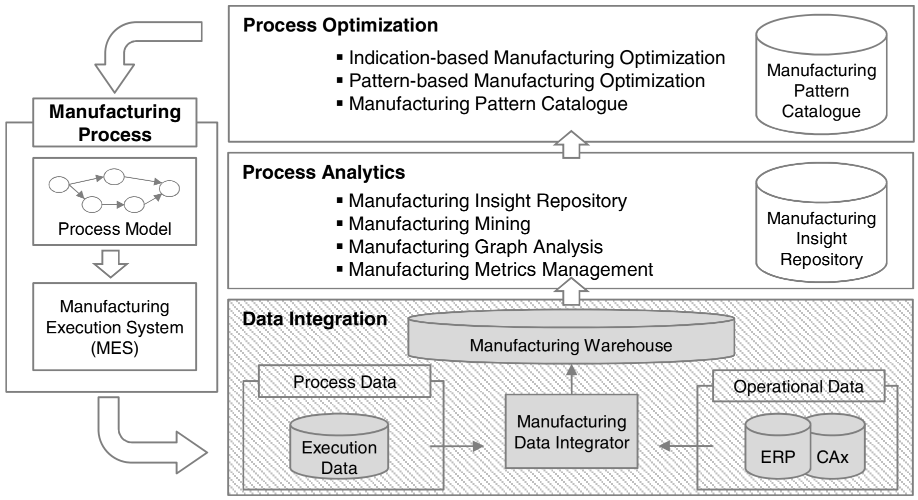

Process Optimization
▪ Indication-based Manufacturing Optimization
▪ Pattern-based Manufacturing Optimization
▪ Manufacturing Pattern Catalogue
Manufacturing Pattern Catalogue
Process Analytics
▪ Manufacturing Insight Repository
▪ Manufacturing Mining
▪ Manufacturing Graph Analysis
▪ Manufacturing Metrics Management
Manufacturing Insight Repository
Data Integration
Manufacturing Warehouse
Process Data
Execution Data
Manufacturing Data Integrator
Operational Data
ERP
CAx
Manufacturing Process
Process Model
Manufacturing Execution System (MES)

Fig. 1. Conceptual architecture of the Advanced Manufacturing Analytics Platform with the focus of this article marked in grey

The AdMA Platform can be seen as an application of process mining [12] to manufacturing whereas it focuses on the enhancement of process models, not on the classic process mining disciplines, i. e., discovery and conformance of process models. In contrast to traditional enhancement concepts, we use not only process data but also operational data in combination with novel analytical approaches, esp. indication-based and pattern-based optimization.

2 Related Work

For the discussion of related work, we distinguish between work referring to warehousing and analysing business processes as well as work related to data integration aspects.

Concepts and techniques for warehousing and analysing business processes are discussed in the area of Business Process Intelligence (BPI) [13]. BPI primarily focuses on workflow-based business processes and related process modelling and process execution concepts. Thus, traditional process warehouse concepts like [14], [15] are based on audit trail data of Workflow Management systems and corresponding meta models like [16]. Next-generation process warehouse approaches – we call them holistic process warehouses – try to enrich process data with additional operational data to realize corresponding holistic BI applications. Initial holistic process warehouse concepts are proposed in [17], [18]. The only fully developed holistic process warehouse, we know of, is the integrated data warehouse of the Deep Business Optimization Platform (dBOP) [8], [19] focusing on workflows.

Yet, workflow-oriented approaches cannot simply be applied to manufacturing for several reasons. First, manufacturing processes are significantly more complex than workflow-based business processes, involving a variety of heterogeneous resources and activities [20]. In addition, process planning and process execution systems in manufacturing use proprietary event models and process data formats. Moreover, processes optimization requires specific metrics and suitable optimization patterns. Based on these constitutive differences, we take the dBOP process warehouse as a starting point to develop a manufacturing-specific holistic process warehouse.

Considering existing manufacturing-specific process warehousing approaches, an initial traditional process warehouse for manufacturing is modelled in [21]. It defines five rudimentary dimensions and a few basic metrics to analyse processes at the level of the whole process. With respect to standardized data warehouse implementations in industry practice, the SAP Business Content as part of the SAP NetWeaver Business Intelligence Platform [22] provides a variety of manufacturing-specific metrics and multi-dimensional data models, called InfoCubes. Yet, it misses a consequent integration of process and operational data in a holistic approach.

Regarding data integration aspects, traditional warehousing concepts are based on Extraction, Transformation and Load (ETL) processes for materialized data integration [23], [24]. A holistic process warehouse requires an extended ETL approach based on the matching of operational and process data [25]. The foundations are general concepts for schema matching and integration [26], [27] that have to be adapted to the specific semantics of process data. [28], [19] present a framework and a tool for matching process and operational data based on workflow standards, esp. BPEL. Taking these concepts as a basis, we develop a procedure for manufacturing-specific matching and ETL that is able to cope with heterogeneous event models and source formats of various data acquisition systems.

3 Requirements and Data Sources

3.1 Analytic Requirements

From a business perspective, there are two central preconditions for efficient and effective manufacturing processes, namely process transparency and process responsiveness [5]. The former alludes to the availability of integrated up-to-date information about currently running processes and their status as well as details about the performance and weaknesses of completed processes, always with respect to the whole process and all participating resources. Transparency is necessary for responsiveness, referring to the ability to quickly realize potentials for improvement and react to changing environmental conditions. The analytic requirements for the Manufacturing Warehouse and the corresponding data integration concepts have to implement these preconditions and realize the vision of the AdMA Platform to provide a standardized integrated BI platform for the holistic data-driven optimization of manufacturing processes. Hence, the following analytic core requirements result:

- Holistic data base: Holistic process optimization requires the integration of all data pertaining to process performance, i. e., operational and process data related to a manufacturing process have to be consolidated and integrated.
- Standardization and Flexibility: To realize pre-defined optimization services a standardized and generalized data model is necessary. As manufacturing processes and data sources are extremely heterogeneous in industry practice, both the data model and the integration concepts should be flexible enough to be adapted to conditions of different manufacturing companies.
- Real-time capability: Both data integration and analytics have to work (near-) real-time to provide the user with up-to-date information about processes in progress.
- Historization: All integrated data have to be historized to analyse process performance over time.

3.2 Data Sources in Manufacturing

To structure potential data sources for the Manufacturing Warehouse we refer to a simplified version of the ISA hierarchy model of manufacturing [29]. We distinguish the following three levels on top of the actual manufacturing process:

The *Business Planning and Logistics* level comprises business-related activities, esp. product and process planning using Computer Aided Design (CAD) and Computer Aided Planning (CAP) systems. Moreover, production planning and scheduling typically supported by ERP systems is carried out on this level as well as Customer Relationship Management (CRM).

The level for *Manufacturing Operations Management* contains all activities to coordinate the execution of manufacturing processes and related resources. Typical IT systems are Production Data Acquisition (PDA) systems for the recording of process execution data as well as Computer Aided Quality (CAQ) systems. MES are the central IT systems for Manufacturing Operations Management integrating and extending PDA and CAQ functionalities. They connect the business level with the actual process by transforming production plans into concrete process executions and reporting results [5].

The *Automation* level comprises all activities for the direct technical monitoring and control of the actual process. At this level, Computer Aided Manufacturing (CAM) systems are used.

On this basis, the central data sources for the Manufacturing Warehouse can be defined: Process execution data is provided by MES, PDA and CAQ systems whereat process model data is supposed to be contained in MES, ERP as well as CAP systems. Operational data, esp. master data concerning product and customer information as well as production plans, is provided by ERP, CRM and CAD systems.

4 Manufacturing Meta Models

4.1 Motivation

To specify a standardized and universal data model, we follow a top-down development approach independent of syntax and semantics of concrete data sources. In order to define generalized information needs for the analysis of manufacturing processes, we develop both conceptual manufacturing meta models and a catalogue of basic manufacturing-specific metrics. In this section, we focus on the meta models, esp. the manufacturing process meta model (MPMM).

The MPMM provides a unified and technology-independent definition of essential concepts and their relationships relevant to the execution of manufacturing processes, e. g., process steps and different types of resources. It is based on a holistic view integrating process and operational aspects independent of the actual data source. Hence, the MPMM represents the basis for the selection of central analysis objects and related entities, i. e., facts and dimensions in the multi-dimensional warehouse scheme.

We complement the MPMM with a conceptual manufacturing event meta model (MEMM) in terms of a state machine with the main states and state transitions of a manufacturing process step, e. g., start, pause and completion of a step. Thus, the MEMM defines requirements for necessary process execution data, i. e., events that have to be provided by corresponding process data sources, e. g., MES. Moreover, it supports the definition of event-based facts in the Manufacturing Warehouse. Due to space restrictions, we do not go into detail on the MEMM in this article.

Both the MPMM and the MEMM abstract proprietary meta models used in data sources, esp. PDA systems and MES, to establish a common and consistent understanding of manufacturing processes and related events for data integration and analytics.

4.2 Manufacturing Process Meta Model

We conducted literature analyses to define a comprehensive meta model for manufacturing processes esp. adapted to the needs of serial and mass manufacturers. It takes a static perspective and models concepts relevant for the execution of a single process instance, i. e., it adopts a run-time not a built-time point of view and doesn't differentiate between process model and process instance. We took generic manufacturing meta models, esp. [21], [29], [30], as a starting point to concretize and extend them with respect to factors influencing the four basic target and analysis dimensions of manufacturing, i. e., time, cost, quality and flexibility [31]. Moreover, for an initial evaluation and refinement of the model, we did industry interviews with manufacturing consultants.

Fig. 2 shows a simplified excerpt of the MPMM as a UML class diagram. A *manufacturing process* consists of *production steps* and is linked with a *production order* defining, e. g., the batch size to be produced. A production order is associated with a *customer*, who can be internal or external, as well as with the *product* that is going to be produced as an output of the process. There are different types of production steps, esp. the actual *manufacturing steps* as well as *transportation steps*. Manufacturing steps refer to the manufacturing and assembly of parts, whereat transportation steps comprise the transportation of parts between different manufacturing steps, e. g., by pallet transporters. In each manufacturing step *scrap quantity* and *yield* of parts can be measured.

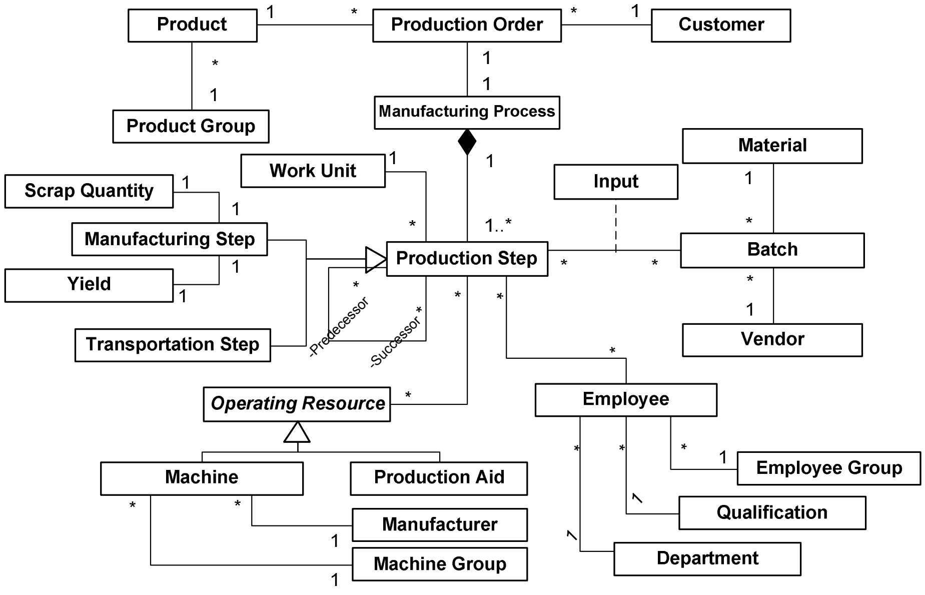


Fig. 2. Excerpt of the Manufacturing Process Meta Model (MPMM)

Regarding spatial aspects, a *work unit* and a corresponding hierarchy of areas and sites can be assigned to a production step. Moreover, a production step can have several *successors* and *predecessors* in a process execution. Resources used and processed in production steps are *employees*, i. e., production workers, *operating resources*, i. e., *machines* and *production aids*, like tools, as well as *material*. All resources can be described by various additional information like *vendors* of material or *manufacturers* of machines. For the sake of simplicity, we omit many details of the MPMM full version, esp. the modelling of operating supply items, environmental emissions or failures of production steps.

5 Manufacturing Warehouse

5.1 Conceptual Warehouse Model

To define a standardized conceptual warehouse model, i. e., a multi-dimensional scheme of the Manufacturing Warehouse, we first describe the generic structure of facts and dimensions of a holistic process warehouse. Next, we develop the actual warehouse model based on the above MPMM.

The generic structure represents a framework for the instantiation of the standardized model in individual cases based on available source data. This is necessary as concrete process and operational data sources vary significantly in existing manufacturing environments. For example, in an energy-intensive manufacturing process power consumption is relevant and recorded. In contrast, in another case CO2 emissions are logged. Hence, different warehouse models result which mainly differ by the dimensions describing the process.

In general, a limited number of process-oriented information is obligatory to define an activity-centric model for a holistic process warehouse. According to [32] we assume that each event during process execution occurs in a certain point of time, is associated with the instance of a single process step, thus is related to a single process instance, and provides a description about itself. Moreover, it provides various information about its context, i. e., links to objects relevant for the corresponding event. E. g., the start event provides information about machines used in the step. Hence, facts at lowest level of granularity are events with the obligatory dimensions "Time", "Process" and "Event". These dimensions are called flow dimensions as they describe the process flow over time. Context information provided by events, e. g., identifiers for material or machines, is the basis for additional dimensions, so called context dimensions. Flow and context dimensions result from process data and are enriched with supplementary operational data forming additional hierarchy levels, so called operational sub dimensions. It has to be remarked that events as central facts have no quantitative characteristics like metrics in traditional data warehouses. Hence, to ease analytics, so called derived, i. e., aggregated, facts are defined at the level of process steps and whole processes comprising basic process-oriented metrics, e. g., cycle time, which may already be computed during ETL. Although it extends the data volume of the warehouse, we decide to model event facts in addition to derived facts as they define the entire scope of process information available, thus enabling flexibility regarding previously unknown information needs. Finally, a fact constellation scheme with shared dimensions results as a generic structure for a holistic process warehouse.

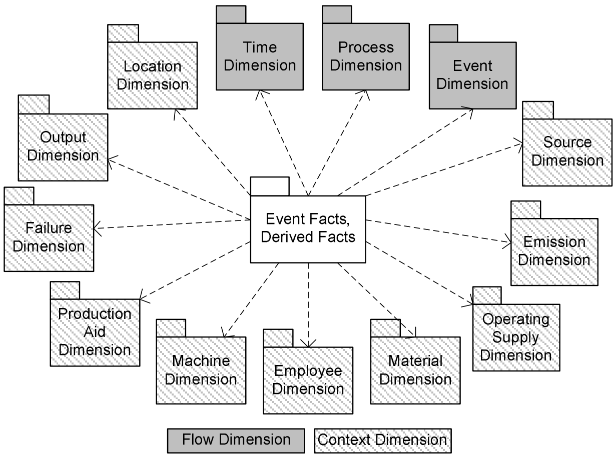


Fig. 3. Conceptual Model of the Manufacturing Warehouse

On the basis of this generic structure we define the standardized model of the Manufacturing Warehouse (see Fig. 3) using the MPMM as well as a set of basic manufacturing metrics related to process steps and whole processes operationalising the four manufacturing target dimensions cost, quality, time and flexibility. We defined ten standardized context dimensions in addition to the obligatory flow dimensions *“Time”*, *“Process”* and *“Event”*.

Fig. 3 shows a simplified version of the model in a multi-dimensional UML [33] package diagram. The *source dimension* refers to the technical source of an event, e. g., a certain machine. *Emissions* comprise environmental pollution like the generation of CO2 or waste. *Material* refers to input that becomes part of the product to be manufactured, whereat *operating supply* items like oil or electricity are consumed during manufacturing. The *output dimension* refers to yield and scrap quantity and the *failure dimension* comprises failures occurred during process execution.

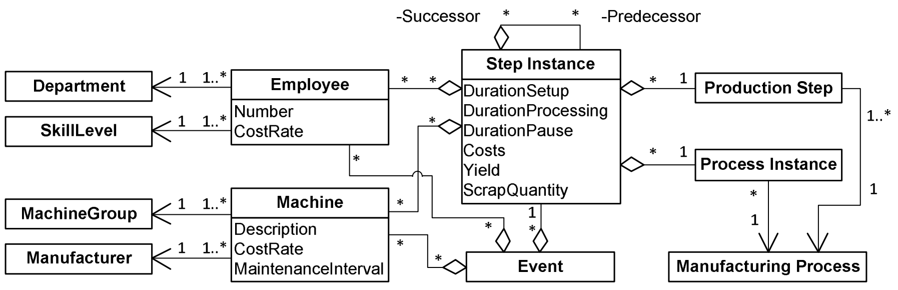


Fig. 4. Excerpt of the detailed Manufacturing Warehouse model

For illustrative purposes, an excerpt of a detailed model of the machine, employee and process dimension is shown in Fig. 4. According to the MPMM, a process comprises steps, whereat steps and processes are instantiated for process execution. Events provide context information about employees and machines and refer to a specified step instance. Derived facts are defined at process step level, e. g., duration for setup or costs of a step, with machines and employees taking part in a step.

5.2 Warehouse Instantiation and Data Integration

As mentioned in Section 2, we need an extended ETL approach for the integration of concrete source data in the Manufacturing Warehouse. Moreover, our standardized warehouse model has to be instantiated in each individual case as mentioned in Section 5.1. Hence, in the following, we provide a coarse-grained overview about the major steps of our integrated procedure for both warehouse instantiation and the integration of source data (see Fig. 5).

Instantiation focuses on the tailoring of the standardized warehouse model to available data sources in an individual case. Integration aims at determining matches between source data and warehouse model to define necessary ETL processes. Our concept relies on the ontology-based annotation [34] of both available process source data, i. e., concrete event logs esp. from PDA systems and MES, and standardized dimensions of the warehouse model. Therefore, we refer to an adapted version of the manufacturing-specific domain ontology in [35]. To enrich process data with operational data, we first match process data with the standardized warehouse model and then match operational data with the resulting selected warehouse dimensions.

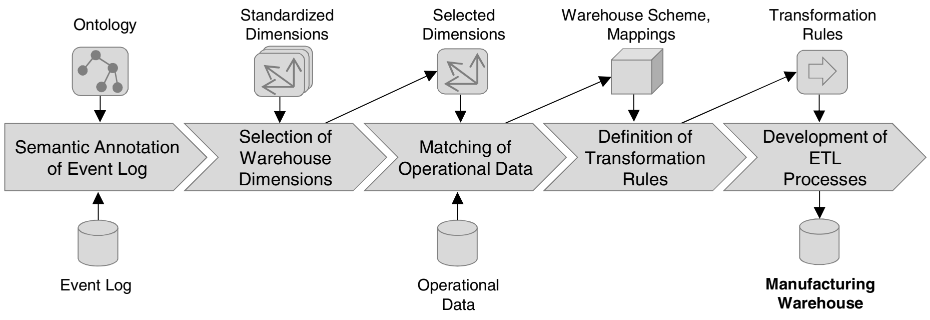

Ontology
Standardized Dimensions
Selected Dimensions
Warehouse Scheme, Mappings
Transformation Rules
Semantic Annotation of Event Log
Selection of Warehouse Dimensions
Matching of Operational Data
Definition of Transformation Rules
Development of ETL Processes
Event Log
Operational Data
Manufacturing Warehouse

Fig. 5. Procedure for warehouse instantiation and data integration

The essential steps of our procedure are sketched in Fig. 5. First, all attributes of a given event log are annotated using the ontology to infer corresponding standardized context and process dimensions. For example, an event log containing the attributes “TimeStamp”, “EventName”, “MachineNumber” and “EmployeeNumber” is annotated. Thus, the process dimensions “Event” and “Time” as well as the two context dimensions “Machine” and “Employee”, resp. the corresponding dimensional attributes, result. Next, the selected dimensions are enriched with available operational data, e. g. from ERP or CRM systems, defining operational sub dimensions. Hence, matching mechanisms are employed to map given operational attributes to dimensional attributes of the standardized warehouse model. In this context, various traditional schema matching techniques [26] or ontology-based methods may be used. E. g., master data of machines and employees, like names and cost rates, are matched. Thus, the complete instantiated model of the Manufacturing Warehouse is based on concrete source data.

Finally, transformation rules are defined to realize all identified mappings. For example, transformations to convert proprietary event names or adjust different currencies are created. These transformation rules are the basis for the development of the corresponding ETL processes populating the warehouse and calculating standardized facts, i. e., metrics.

6 Prototypical Implementation and First Proof of Concept

Our current prototypical implementation comprises a first relational version of the Manufacturing Warehouse, basic data transformation and data mining functionalities as well as a dashboard-oriented GUI and is described in [7]. In addition, we developed universal process-centric data mining use cases for Indication-based Manufacturing Optimization (IbMO) presented in [11]. We are currently realizing the above instantiation and integration procedure in the Manufacturing Data Integrator, too. In the following, we demonstrate that the Manufacturing Warehouse in combination with IbMO enables the generation of novel insights for process improvement beyond traditional process warehousing approaches.

In a first proof of concept we implemented the so called metric-oriented root cause analysis as an IbMO explication use case designed for production managers [11]. It aims at explaining categorized metrics of process instances, e. g., lead time, by providing comprehensible explication models, namely decision trees. E. g., reasons for excessive lead times can be identified. Moreover, we developed a sample scenario for a manufacturing process, the production of steel springs for the automotive industry, and generated corresponding data to load the Manufacturing Warehouse. On this basis, we conducted metric-oriented root cause analyses on lead times. The latter are categorized as "OK" or "too high" in our proof of concept. Two exemplary decision rules, that are based on the decision tree depicted in Fig. 6, are:

- If the *first machine* in *production step 1* was maintained more than *15 days ago* and *vendor V7* delivered the *material* processed in *step 3*, then *lead time* is typically too high.
- If the former isn't the case but the *skill level of the first employee* in *step 2* is lower than *level 4* and *machine M2* is used, then *lead time* is typically too high.

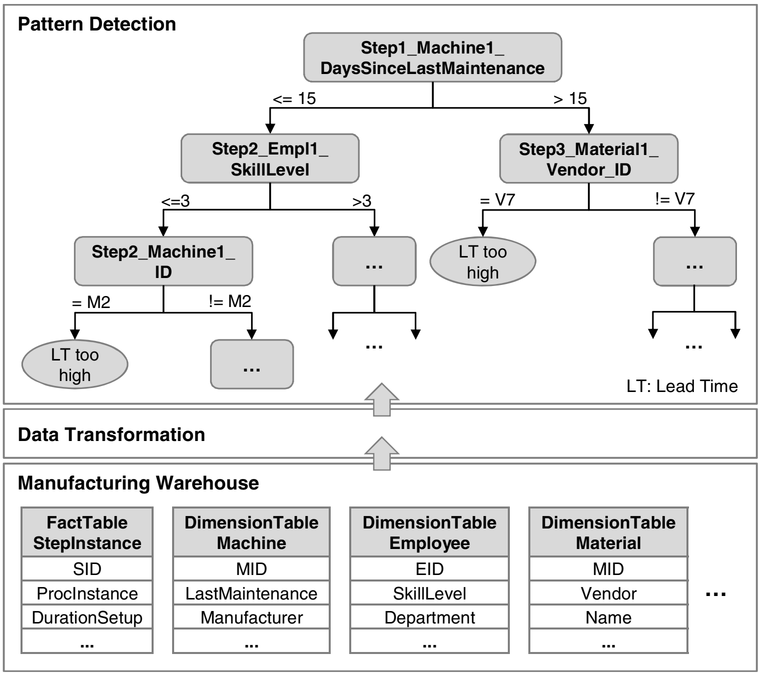


Fig. 6. Functional components of the prototype

These decision rules represent valid indications for concrete process improvements, e. g., to enhance training for employees engaged in step 2 or improve maintenance schedules for machines used in step 1. They demonstrate the fundamental feasibility and usefulness of the Manufacturing Warehouse in combination with suitable analytics. Based on the integration of operational and process data, the Manufacturing Warehouse enables the cross-correlation of all relevant aspects pertaining to process performance, e. g., machine- product-, material-, and employee-oriented aspects, in order to generate novel insights for process optimization. In contrast, typical traditional process warehouses are not aware of operational aspects like additional information on vendors of input material or employee training. In general, the universal holistic data model of the Manufacturing Warehouse can be used as a basis for various holistic analytics, ranging from holistic OLAP and reporting concepts to data mining-driven approaches like IbMO and pattern-based optimization.

Fig. 6 shows the exemplary decision tree from which the above listed decision rules were deduced from as well as the necessary functional components of our prototype for metric-oriented root cause analyses. On top of the *Manufacturing Warehouse*, *data transformation* is concerned with data denormalization and data filtering which prepare data for *pattern detection*, i. e., decision tree induction. The relational structure of the warehouse is deduced from the above conceptual model. Further technical details about the prototype are given in [7], [11].

7 Conclusion and Future Work

In this article we presented the Manufacturing Warehouse, a concept for a holistic manufacturing-specific process warehouse as central part of the overall Advanced Manufacturing Analytics Platform. It integrates operational and process data in a standardized multidimensional warehouse and is based on a generalized manufacturing process meta model. In addition, we introduced a procedure for warehouse instantiation and the integration of concrete source data.

To demonstrate the usefulness and feasibility of the Manufacturing Warehouse, we described a first proof of concept comprising a process-centric data mining use case for Indication-based Manufacturing Optimization on top of the warehouse.

In our future work, we plan to investigate application scenarios in discussion with industry partners comprising typical MES and ERP systems to further validate and extend the Manufacturing Warehouse. Moreover, we are going to refine it with respect to the implementation of pattern-based optimization in manufacturing.

Acknowledgement. The authors would like to thank the German Research Foundation (DFG) for financial support of this project as part of the Graduate School of Excellence advanced Manufacturing Engineering (GSaME) at the University of Stuttgart.

References

1. Jacob, F., Strube, G.: Why Go Global? The Multinational Imperative. In: Global Production. A Handbook for Strategy and Implementation, pp. 2–33. Springer, Berlin (2008)
2. Slack, N., Chambers, S., Johnston, R.: Operations Management, 6th edn. Financial Times Prentice Hall, Harlow (2010)
3. Niedermann, F., Radeschütz, S., Mitschang, B.: Business Process Optimization Using Formalized Optimization Patterns. In: Abramowicz, W. (ed.) BIS 2011. LNBIP, vol. 87, pp. 123–135. Springer, Heidelberg (2011)
4. Muehlen, M.z., Shapiro, R.: Business Process Analytics. In: Handbook on Business Process Management 2. Strategic Alignment, Governance, People and Culture, pp. 137–158. Springer, Berlin (2010)
5. Kletti, J. (ed.): Manufacturing Execution Systems - MES. Springer, Berlin (2007)
6. Connolly, T., Begg, C.E., Holowczak, R.: Business database systems. Addison-Wesley, New York (2008)
7. Gröger, C., Niedermann, F., Schwarz, H., Mitschang, B.: Supporting Manufacturing Design by Analytics. Continuous Collaborative Process Improvement enabled by the Advanced Manufacturing Analytics Platform. In: Proceedings of CSCWD 2012 (to appear, 2012)
8. Niedermann, F., Radeschütz, S., Mitschang, B.: Deep Business Optimization: A Platform for Automated Process Optimization. In: INFORMATIK 2010 - Business Process and Service Science - Proceedings of ISSS and BPSC, Leipzig, Germany, September 27-October 1, pp. 168–180. Gesellschaft für Informatik, Bonn (2010)
9. Niedermann, F., Schwarz, H.: Deep Business Optimization: Making Business Process Optimization Theory Work in Practice. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 88–102. Springer, Heidelberg (2011)
10. Niedermann, F., Schwarz, H., Mitschang, B.: Managing Insights - A Repository for Process Analytics, Optimization and Decision Support. In: Proceedings of the International Conference on Knowledge Management and Information Sharing (KMIS) 2011. SciTePress, Paris (2011)
11. Gröger, C., Niedermann, F., Mitschang, B.: Data Mining-driven Manufacturing Process Optimization. In: Proceedings of ICMEEM 2012 (to appear, 2012)
12. van der Aalst, W.M.P.: Process mining. In: Discovery, Conformance and Enhancement of Business Processes, Springer, Heidelberg (2011)
13. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M.S.M.: Business Process Intelligence. Computers in Industry 53, 321–343 (2004)
14. Bonifati, A., Casati, F., Dayal, U., Shan, M.-C.: Warehousing Workflow Data: Challenges and Opportunities. In: Very Large Databases. Twenty-Seventh International Conference on Very Large Data Bases, Roma, Italy, September 11-14, pp. 649–652. Morgan Kaufmann, San Francisco (2001)
15. Muehlen, M.z.: Process-driven Management Information Systems - Combining Data Warehouses and Workflow Technology. In: Proceedings of the Fourth International Conference on Electronic Commerce Research (ICECR-4), Dallas, pp. 550–566 (2001)
16. Leymann, F., Roller, D.: Production Workflow. Concepts and techniques. Prentice Hall, New Jersey (2000)

17. Casati, F., Castellanos, M., Umeshwar, D., Salazar, N.: A Generic solution for Warehousing Business Process Data. In: Proceedings of the 33rd International Conference on Very Large Data Bases University of Vienna, University of Vienna, Austria, September 23-28, pp. 1128–1137. ACM, New York (2007)
18. Muehlen, M.z.: Workflow-based Process Controlling. Foundation, Design and Application of Workflow-driven Process Information Systems. Logos, Berlin (2004)
19. Radeschütz, S., Niedermann, F., Bischoff, W.: BIAEditor - Matching Process and Operational Data for a Business Impact Analysis. In: Proceedings of 13th International Conference on Extending Database Technology, EDBT 2010, Advances in Database Technology, Lausanne, Switzerland, March 22-26, pp. 705–708. ACM, New York (2010)
20. Committee to Study Information Technology and Manufacturing, Computer Science and Telecommunications Board, Manufacturing Studies Board, National Research Council of the US: Information Technology for Manufacturing. A Research Agenda. National Academy Press, Washington (1995)
21. Silverston, L.: The data model resource book. vol. 2. A Library of Universal Data Models by Industry Types. Wiley, New York (2001)
22. McDonald, K., Wilmsmeier, A., Dixon, D.C., Inmon, W.H.: Mastering the SAP business information warehouse. Leveraging the business intelligence capabilties of SAP NetWeaver, 2nd edn. Wiley, Indianapolis (2006)
23. Inmon, W.H.: Building the Data Warehouse, 4th edn. Wiley, Indianapolis (2005)
24. Bernstein, P., Haas, L.M.: Information Integration in the Enterprise. Communications of the ACM 51, 72–79 (2008)
25. Radeschütz, S., Mitschang, B.: An Annotation Approach for the Matching of Process Variables and Operational Business Data Models. In: Proceedings of the ISCA 21st International Conference on Computer Applications in Industry and Engineering, CAINE 2008, Honolulu, Hawaii, USA, November 12-14, pp. 144–149. ISCA, Honolulu (2008)
26. Leser, U., Naumann, F.: Informationsintegration. Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. dpunkt, Heidelberg (2007)
27. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB Journal 10, 334–350 (2001)
28. Radeschütz, S., Mitschang, B., Leymann, F.: Matching of Process Data and Operational Data for a Deep Business Analysis. In: Enterprise Interoperability III, pp. 171–182. Springer, London (2008)
29. International Society of Automation (ISA): Enterprise-Control System Integration. Part 1: Models and Terminology. ISA 95-1 (2000)
30. Dangelmaier, W.: Fertigungsplanung. Planung von Aufbau und Ablauf der Fertigung, 2nd edn. Springer, Berlin (2001)
31. Thonemann, U.: Operations Management. Konzepte, Methoden und Anwendungen, 2nd edn. Pearson, München (2010)
32. van Dongen, B.F., van der Aalst, W.M.P.: A Meta Model for Process Mining Data. In: van Dongen, B.F., van der Aalst, W.M.P. (eds.) EMOI-INTEROP (2005)
33. Luján-Mora, S., Trujillo, J., Song, I.-Y.: Multidimensional Modeling with UML Package Diagrams. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 199–213. Springer, Heidelberg (2002)
34. Linkova, Z.: Ontology-Based Schema Integration. In: SOFSEM 2007: Theory and Practice of Computer Science, pp. 71–80. Institut of Computer Sciences AS CR, Prague (2007)
35. Verein Deutscher Ingenieure (VDI): Manufacturing Execution Systems (MES). Part 3. VDI 5600-3 (2007)

Queen-Bee: Query Interaction-Aware for Buffer Allocation and Scheduling Problem

Amira Kerkad, Ladjel Bellatreche, and Dominique Geniet

LIAS/ENSMA Poitiers University, Futuroscope, France
{amira.kerkad,bellatreche,dgeniet}@ensma.fr

Abstract. In the relational data warehouses, each OLAP query shall involve the fact table. This situation increases the interaction between queries that can have a significant impact on the warehouse performance. This interaction has been largely exploited in solving isolated problems like (i) the multiple-query optimization, (ii) the materialized view selection, (iii) the buffer management, (iv) the query scheduling, etc. Recently, some research efforts studied the impact of the query interaction on optimization problems combining interdependent sub-problems such as buffer management problem (BMP) and the query scheduling problem (QSP). Note that combining two complex problems usually increases the complexity of the integrated problem. In this paper, we study the effect of considering the query interaction on an integrated problem including BMP and QSP (namely, BMQSP). We first present a formalization of the BMQSP and show its hardness study. Due to high complexity of the BMQSP, we propose an algorithm called queen-bee inspired from the natural life of bees. Finally, theoretical and effective (on Oracle 11G) experiments are done using the star schema benchmark data set.

Keywords: Multi-query optimization, Query Scheduling, Buffer Management, Query Interaction.

1 Introduction

Relational data warehouses ($\mathcal{RDW}$) represent the ideal environment in which complex OLAP queries interact with each other. These queries often have a lot of common sub-expressions, either within a single query, or across multiple such queries run as a batch. This is because binary operations (such as join) involve the fact table of the $\mathcal{RDW}$ schema. This phenomenon is related to the *problem of multi-query optimization* (MQO). MQO aims to exploit (reuse) results of *common sub-expressions*. This is a major cause of performance problems in database systems [1]. Several research studies were focused on the *modeling* and the *exploitation* of the query interaction. From modeling point of view, several graph structures were proposed: *multiquery graph* [4, 21] and *multiple view processing plan* [22]. It has been exploited by several research studies either in the centralized and the distributed environments by proposing solutions for materializing and caching the intermediate results of the sub-expressions [18, 15, 14, 3].

The selection of materialized views is one of the major problems that exploits this phenomenon [22]. When materialized views are selected; their storage may concern two main devices: the *hard disk* and the *main memory*. Materialized intermediate results are usually stored on the disk, especially, when a large number of huge views is selected. Due to the growing size of the main memory, the intermediate results become candidate for bufferisation (consequently stored in the memory). This solution avoids the high cost of reading from or writing to disk. In this case, the MQO impacts directly the buffer management problem (BMP). Another important problem that may be impacted by MQO is the query scheduling (QSP). The QSP consists in finding an ordering of the queries that reduces the overall cost of processing queries. Recently, in [14], the authors propose a method for selecting materialized views incorporating the QSP.

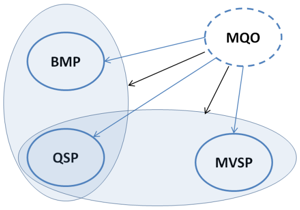


Fig. 1. Interaction between different optimization techniques

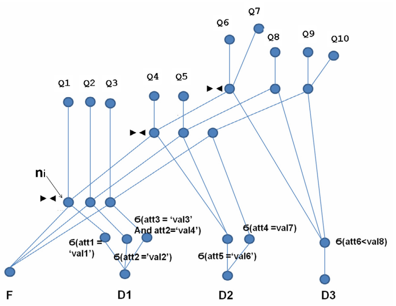


Fig. 2. A MVPP for ten star join queries

By exploring the most important studies related to QSP, we distinguish two main classes: (i) *partial query scheduling*: it consists in finding the best order to evaluate sub-expressions identified by the means of MQO (this problem is known as NP-complete) [17, 18, 20] and (ii) *entire query scheduling*: it considers the entire queries for the scheduling process [20, 14]. Usually, most studies done on the buffer management in traditional databases, in general and in the $\mathcal{RDW}$, in particular, assume that queries are already ordered. As consequence, to execute the first query of a given workload, the DBMS may perform the following tasks: (1) it identifies the relevant disk pages; (2) it loads them in the main memory cache and (3) it executes the query. The next query of the workload may get benefit from the actual content of the buffer if it shares some intermediate results with the first query. If not, DBMS repeats the same process as for the first one, and so on. Based on this scenario, we claim that if the query scheduler has a snapshot of the buffer content, it may reorder the queries to allow them getting benefit from the buffer. This shows the *strong interaction* between the BMP and the QSP [20, 9]. The complexity of the integrated problem including BMP and QSP may be very high [20]. It is proportional to the number of queries of the workload and the number of intermediate sub-expressions. To reduce this complexity, we propose in this paper to use the *divide and conquer* method. It consists in dividing the queries of the workload into subsets treated separately. This partitioning is performed based on the affinity between queries. To illustrate our proposal, let us consider the following motivating example.

1.1 Motivating Example

We assume a workload with 10 OLAP queries, where each query may be represented by an algebraic tree. Due to the strong interaction between queries, their 10 trees may be merged to generate a graph structure called, *Multiple View Processing Plan* (MVPP) [22] (Figure 3). The leaf nodes of MVPP represent the tables of the $\mathcal{RDW}$. The root nodes represent the final query results and the intermediate nodes represent the common sub-expressions shared by the queries. Among intermediate nodes, we distinguished: (i) *unary nodes* representing the selection (σ) and the projection operations and (ii) *binary nodes* representing join ($\bowtie$), union, intersection, etc. The intermediate results of the MVPP are candidates for bufferization. To show the interaction between BMP and QSP, we propose to reorganize the initial structure of the MVPP by generating clusters of queries, each one contains queries having at least one common node. Each cluster is called *hive*. Figure 3 shows the results of clustering of our MVPP, where three hives are obtained. In each hive, we elect a query to be the *Queen-Bee*. Once elected all its common nodes are cached. Thus, queries in the same *hive* will be *ordered* and get benefit from the buffer content. A long this paper, we detail the hints announced in this example: (1) the identification of intermediate nodes, (2) the generation of hives, (3) buffer allocation, (4) query scheduling, and (5) validation of the proposal.

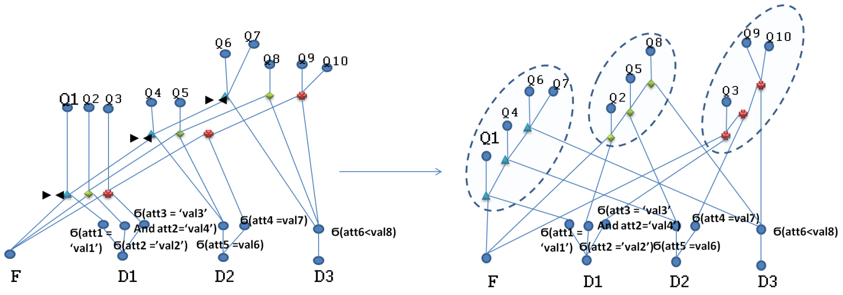


Fig. 3. An example for MVPP with clustered queries (hives)

This paper is structured as follows: Section 2 presents related works. Section 3 formalizes our combined problem and presents all ingredients of our approach. Section 4 details our queen-bee algorithm. Section 5 gives experiments validating our proposal in a simulated environment and on Oracle 11G DBMS. Section 6 concludes our paper by summarizing the main results and presenting some open issues.

2 Related Work

Several studies have shown the importance of the query interaction and its practicability in commercial DBMS [2, 14–16, 21, 22]. This interaction is mainly related to an important problem of the physical design of $\mathcal{RDW}$ known by the MQO. Note that each query can have several alternative evaluation plans, each with a different set of tasks. Therefore, the goal of MQO is to choose the right set of plans for queries which minimizes the total execution time by performing common tasks only once. MQO is known as NP-hard problem [16]. The MQO impacts other hard optimization problems such as materialized view selection problem (MVSP) [22], BMP [7, 6, 8], QSP [5, 12] and the problem combining BMP and QSP [19, 9, 20] as shown in Figure 1.

The BMP got a lot of attention from the database community, where it has studied in different types of databases: (a) traditional databases [7, 6, 8], (b) semantic databases [23], (c) data warehouses [15], (d) flash databases [13]. In the first generation of studies related to BMP, solutions were proposed to allocate pages and to replace them when the buffer is full using policies like *LRU*, *FIFO*, etc. without considering the impact of the MQO. In the second generation, some research efforts were concentrated on incorporating MQO in the buffer allocation [7, 15], where algorithms for selecting common intermediate results to be cached in a limited cache space were proposed.

Similarly, the QSP was studied in isolated way in several environments: *centralized* [20], *distributed* and *parallel* databases/data warehouses [12]. It has been proved as a strongly NP-complete problem [9, 5, 12]. After, it has been mixed with other optimization problems like the MVSP [14]. [19, 9] considered the problem of caching and QSP in the context of $\mathcal{RDW}$ and proposed some heuristics. [20] presented several issues related to the combination of BMP and QPS by considering MQO problem. A complete hardness study was proposed and a tentative of algorithms without a real validation is proposed. By summarizing the few existing studies on combined BMP and QSP (BMQSP) considering MQO, some observations are identified:

– they explore the large search space of the combined problem which may be very large.
– they use simple cost models that ignore parameters related to the cache content, the order of queries, the size of intermediate results of joins, the characteristic of star join queries of $\mathcal{RDW}$, etc.
– they do not validate their proposals using a real DBMS with large memories.

3 Formalization and Cost Models

In this section, we give some key concepts to facilitate the formalization of our problem. We also give a hardness study and describe our cost model. In this work, we consider some assumptions: **(1)** prior knowledge of the workload (offline scheduling), **(2)** a centralized $\mathcal{RDW}$ environment and **(3)** the initial cache content is considered as empty.

3.1 Concepts and Formalization

Given a workload with n queries generated by user applications presented in a queue. Based on their MVPP, some definitions are given (Figure 3).

Definition 1. *The fan-out of a query is defined as the number of overlapping nodes with other queries.*

Definition 2. *The overlapping accumulated cost of a query (noted* OAC*) is the total execution cost of all its overlapping nodes with other queries. It represents the participation of this query in the overall cost reduction.*

To facilitate the understanding of the formalization of the combined problem BMQSP, we start presenting a separate formalization of both BMP and QSP.

BMP is formalized as follows: **(1)** Inputs : (i) $\mathcal{RDW}$, (ii) a workload with a set of queries $\mathcal{Q} = \{Q_1, Q_2, ..., Q_n\}$ represented by a MVPP, (iii) a set of intermediate nodes of MVPP candidates for caching $\mathcal{N} = \{no_1, no_2, ..., no_l\}$, **(2)** Constraint: a buffer size $\mathcal{B}$ and **(3)** Output : a buffer management strategy $\mathcal{BM}$ that allocates nodes in the buffer to optimize the cost of processing $\mathcal{Q}$.

QSP is formalized as follows: **(1)** Inputs : (i) $\mathcal{RDW}$, (ii) a workload with a set of queries $\mathcal{Q} = \{Q_1, Q_2, ..., Q_n\}$ and (iii) a $\mathcal{B}M$. **(2)** Output : scheduled queries $\mathcal{QS} = \{SQ_1, SQ_2, ..., SQ_n\}$.

BMQSP is described based on the above formalizations as follows:

- Inputs : (i) A $\mathcal{RDW}$ and (ii) a set of queries $\mathcal{Q} = \{Q_1, Q_2, ..., Q_n\}$ represented by a MVPP, (iii) a set of intermediate nodes of MVPP candidates for caching $\mathcal{N} = \{no_1, no_2, ..., no_l\}$;
- Constraint: a buffer size $\mathcal{B}$;
- Output : (i) scheduled queries $\mathcal{SQ} = \{SQ_1, SQ_2, ..., SQ_n\}$ and (ii) a $\mathcal{B}M$, minimizing the overall processing cost of Q.

3.2 Hardness Study

To compute the complexity of our BMQSP, we first relax the buffer management. Exhaustively, a workload with n queries requires $n!$ possible schedules. If we consider the cache, the number of possible subsets of nodes candidates[1] for bufferization is given by: $\sum_{i=1}^{l}(i!)$. Thus, simultaneous resolution of query scheduling and buffer management needs an exploration of a search space of size: $n! \times \sum_{i=1}^{l} i!$, where n and l represent respectively, the number of queries and intermediate results. This is because different solutions may potentially interact with each other. This complexity motivates us to develop efficient and reasonable solutions.

[1] We assume that a node is cached at most once during the workload runtime.

3.3 Cost Model

To quantify the quality of the solutions, we define a cost model to estimate the number of inputs/outputs (I/O) required for executing the set of OLAP queries. We describe the cost model in two cases: without caching and with caching.

Without Caching. Most studies using cost models for selecting optimization structures ignore buffer management aspects. Therefore, the cost of a given query Q_i may be estimated as the sum of the costs of all joins, aggregations and projections. More concretely, the cost of executing a query Q_i involving several joins between the fact table F and the dimension tables $\mathcal{D}^{Q_i} = \{D_1^{Q_i}, ..., D_{n_i}^{Q_i}\}$ is composed of three types of operations : (1) first join (FJ), (2) intermediate result (IR) and (3) final operation (AG) (aggregation, group by). More details about joins order and estimating result sizes are available in [10].

We assume that each query Q_i is represented by an ordered set of all the operations of its plan : $op_1^{Qi}, op_2^{Qi}, \dots op_{l_i}^{Qi} / op_k^{Qi} \in \{FJ, IR, AG\}$ from the first join till the final operation, with l_i the number of operations in Q_i. We define a function $size(op_k^{Qi})$ estimating the result size of op_k^{Qi}. Recursively, we estimate the cost of a query starting from the final operation $op_{l_i}^{Qi}$ as follows :

$$Cost(op_k^{Q_i}) = \begin{cases} size(op_k^{Q_i}) & \textit{if } k = 1 \\ size(op_k^{Q_i}) + Cost(op_{k-1}^{Q_i}) & \textit{if } k \in [2, l_i]\} \end{cases} \tag{1}$$

With Caching. To take into account cache management in our cost model, a buffer content checking is required. The cost of an operation op_k^{Qi} is ignored if : (i) its result is cached, or (ii) one of its successors ($op_{k+\alpha}^{Qi}$ with $\alpha > 1$) is cached. For this reason, we define a function $b(op_k^{Qi})$ to check if the result of op_k^{Qi} is present in the buffer (=1) or not (=0). Therefore, the execution cost will be :

$$Cost(op_k^{Q_i}) = \begin{cases} \left[size(op_k^{Q_i})\right] \times \prod_{j=k}^{l_i} b(op_j^{Q_i}) & \textit{if } k = 1 \\ \left[size(op_k^{Q_i}) + Cost(op_{k-1}^{Q_i})\right] \times \prod_{j=k}^{l_i} b(op_j^{Q_i}) & \textit{if } k \in [2, l_i] \end{cases} \tag{2}$$

Total Cost. The total execution cost of the workload is estimated as the sum of all queries cost: $Total\_cost = \sum_{i=1}^{n} Cost(op_{l_i}^{Q_i})$.

4 Queen-Bee Algorithm

To reduce the complexity of the combined problem, we propose an approach inspired from the natural life of bees. It is illustrated in Figure 4. We choose to present our approach incrementally since it concerns two main problems: BMP and QSP. The basic idea behind our queen-bee algorithm is to partition the queries of the MVPP and then for each *hive*, elect a query (*queen-bee*) to be executed first and its nodes will be cached. We associate the dynamic buffer management strategy DBM[11] to our queen-bee algorithm. DBM mainly traverses the query plan and, for each intermediate node (operation), it checks the buffer content: if the result of the current operation is already cached, then no need to load it from the hard disk, else it is cached while there is enough buffer space. Once a node of a given *hive* is treated, its rank[2] value is decremented. When the rank of a node is equal 0, it will be removed from the buffer since it is useless for coming queries.

Our query scheduler works on the clusters of queries (*hives*) and it shall order the queries inside each *hive* according the buffer content. To do so, three modules define our queen-bee algorithm: (1) generating of a query graph with connected components (QGCC), (2) sorting the components (α-sort) which is optional depending on whether queries have priority or not and (3) sorting queries inside each component (β-sort). Contrary to the scheduling strategy proposed in [11] (called dynamic query scheduler) that takes into account only the cache content, our scheduler considers other parameters regarding the *query execution cost*, the *query fan-out* and the overlapping accumulated cost of a query (OAC) (see Definitions 1 and 2). These factors are used to sort nodes of each *hive*.

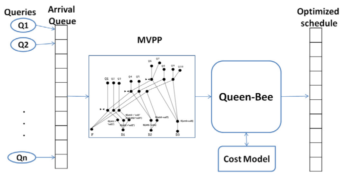


Fig. 4. Our methodology

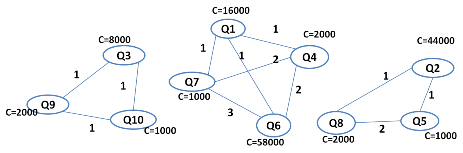


Fig. 5. An example of QGCC

Generating the Query Graphs with Connected Components. Starting from MVPP, a QGCC representing *hives* is obtained. Vertices represent the queries of each *hive*. Each vertex is tagged with a value C representing the I/O cost of its corresponding query. An edge exists between two vertices if they have common node(s). It is labeled by the number of the shared nodes. Figure 5 shows the corresponding QGCC for the MVPP in Figure 2.

α-Sort. In our study, all queries have the same priority. Therefore, the required data for two different hives are disjoint ({(Q1,Q7,Q4,Q6);(Q3,Q9,Q10);(Q2,Q8,Q5)} and {(Q3,Q9,Q10);(Q1,Q7,Q4,Q6);(Q2,Q8,Q5)} have the same cost).

β-Sort. The queries inside each component need to be scheduled. For this reason, β-sort takes each component and schedules its queries by performing two steps: (1) identification of the *Queen-Bee* based on the chosen criterion (cost, fan-out, *OAC*)[3] (2) exploration of the rest of nodes (using the same criterion used

[2] We define the rank value of a node no_i as a counter representing the number of queries accessing no_i.

[3] The choice of criterion is done by the database administrator.

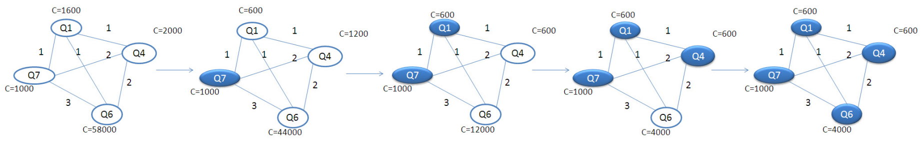


Fig. 6. Traversing the component by the minimal cost

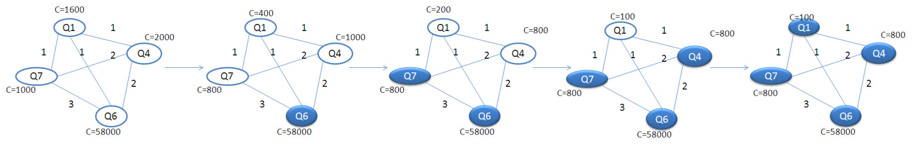


Fig. 7. Traversing the component by the maximal fan-out

for selecting the queen-bee), and (3) once the connected component is entirely traversed, return the obtained sub-schedule. Figure 6 gives an example of β-sort using the cost criterion in ascending order. When we traverse the component starting from the query having the minimal execution cost, the cost of vertices which are not traversed yet is updated depending on cache content. The next vertex to be visited is the one with minimal cost. The algorithm repeats the operation while the component is not entirely traversed. This sort "*sacrifices*" the query with minimal cost by executing it first. This is because it will not get any relevant data in cache. But, this same query will give relevant data for the other queries which are more expensive, because they share at least one overlapping node.

The fan-out criterion may also be interesting for choosing the Queen-Bee. The idea is to start by the query giving more relevant data to others. The figure 7 gives an example of β-sort of fan-out in descending order. We can easily observe that theoretically, sorting by minimal cost gives different performance than sorting by maximal fan-out [4]. Sorting criteria are studied in the experimental studies.

5 Experiments

To validate our proposal, we conduct intensive experimental studies in both *theoretical* and *effective* ways. The theoretical experiments are based on our mathematical cost model and our simulation tool. The results by our simulations are then implemented on Oracle11G.

5.1 Dataset and Workload

Our experiments are done on the Star Schema Benchmark (SSB of 100GB) having a fact table *Lineorder* (6,000,000 $\times SF$ tuples) and 4 dimension tables

[4] In Figure 6, total cost=1000+600+600+4000=6200 I/O, and in Figure 7, total cost=58000+800+800+100=59700 I/O.

with a scale factor $SF = 100$. A server with 32 GB of RAM is used. We consider thirty queries, detailed in [10], covering different types of OLAP queries.

5.2 Simulation Tool

To make easier our testing, we developed a *simulation tool* using a Java development environment. It contains three main modules : **(1)** $\mathcal{RDW}$ connectivity and meta-data extraction needed for the cost model, **(2)** Setting $\mathcal{RDW}$ parameters and handle the workload and **(3)** optimization module handles different parameters (e.g. Buffer size), the choice of algorithms and the output detailing the obtained solutions for administrators. If they are not satisfied by these results, the optimization module gives them the possibility to tune some parameters. Otherwise, administrators deploy them on Oracle11G using appropriate scripts.

5.3 Obtained Results

We start by studying the impact of criteria used to select the queen-bees of all components: *cost*, *fan-out* and *OAC*. We run experiments by considering ascendant sorting (minimal value) and descend sorting (maximal value) for each criterion. Figure 8 summarizes the obtained results. The three factors give better performance when they are used in an ascendant sorting. Note that a query with a minimal *fan-out* or OAC is usually a query with few nodes (less operations). This explains the fact that these two factors give similar results.

In [11], we proposed some algorithms to get a near optimal query scheduling and its buffer management using respectively dynamic query scheduler (DQS) and dynamic buffer management (DBM). In Figure 10, our previous algorithms and the queen-bee are compared with results obtained using LRU policy instead of DBM, and no scheduling instead of DQS. Different buffer pool sizes are used to show its impact on the total performance. From these results, we observe the following: (1) DBM is more adapted than LRU in our context; (2) DQS evolves the efficiency of the buffer management policy; and (3) Queen-Bee gives the same performance as the heuristics that use DQS and DBM. We also notice that beyond a threshold of buffer pool space, query scheduling has no effect on the final performance because all candidate nodes can fit in the cache. That's why the DBM without scheduling gives the same performance as DBM-DQS and Queen-Bee at a buffer space of $32GB$.

One of the motivations of our proposal is to prune the search space. In Figure 9, we can see that the queen-bee algorithm is much faster than the DBM-DQS even though they give the same query performance.

5.4 Validation on Oracle11g

For validation, we deploy our simulation results on an Oracle 11g DBMS with the same data set (SSB of 100GB and 30 queries). Queries are executed for each solution schema obtained by our algorithms. To perform this validation, queries are rewritten to take into account caching solution proposed by different

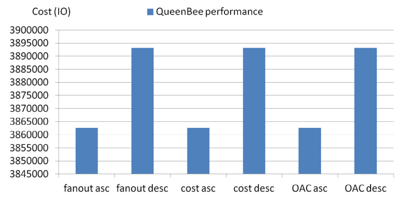


Fig. 8. Different factors used in β-sort

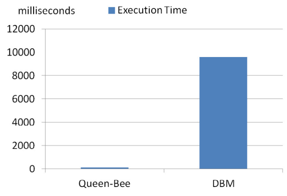


Fig. 9. Comparing Run time

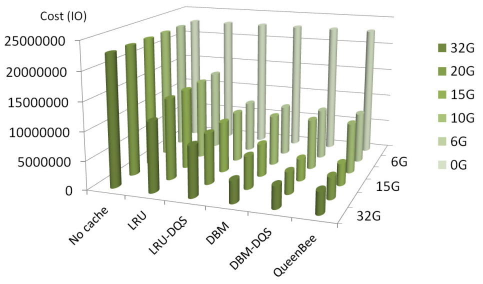


Fig. 10. Comparing different algorithm's performance

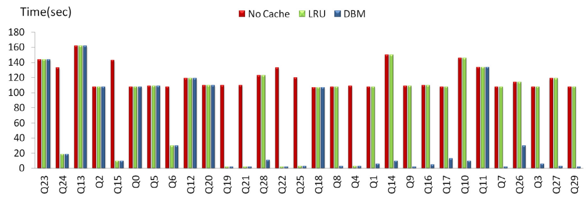
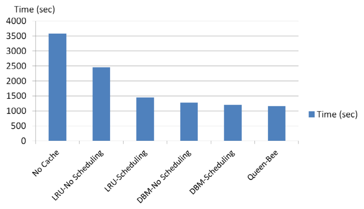


Fig. 11. LRU vs DBM : experiments with static schedule



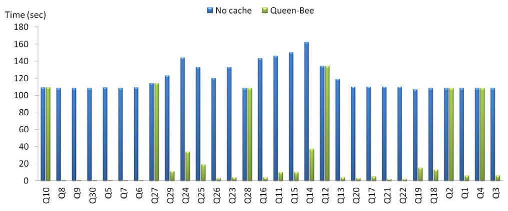


Fig. 12. Deploying algorithms' results **Fig. 13.** Deploying Queen-Bee results

algorithms. The DBMS parameters are tuned to prepare buffer pool[11]. To compare LRU with the DBM, we take the same workload in a static order on Oracle 11g with 6GB of buffer pool. The queries are executed using: (1) no buffer management, (2) LRU, and (3) our DBM. In Figure 11, we can observe that LRU policy gives a good performance for some queries but not enough to cover a larger number of queries as the DBM.

The validation of the simulation experience in Figure 10 is done on our DBMS. Real performance is given in Figure 12 which shows the similarity with the theoretical results. This proves the quality of our cost model. Figure 13 shows the impact of Queen-Bee optimization on the workload performance. We can see that the number of queries that have no benefit from cache content is 6 of 30 queries. This corresponds exactly to the number of query *hives* of our workload. We can see that only the first query (the queen-bee) has no reduction in cost, but all the remaining queries have important performance gain because of sharing at least one common result.

6 Conclusion

The query interaction is one of the most important characteristics of data warehouse applications. It has been exploited to solve several complex optimization problems during the physical design phase of a data warehouse. Recently, it has been exploited to solve problems combining other sub-problems such as buffer management and query scheduling. This integrated problem is studied in this paper. To tackle this problem, a methodology is given. It starts from a set of queries obtained from user applications; and placed in a queue. Then, algebraic trees of these queries are merged to form a graph, called multiple views processing plan. The intermediate nodes of this plan are candidates for bufferization. To avoid the explosion of evaluating each intermediate node and study its effect on query scheduling, we propose an approach inspired from the natural life of bees. It partitions queries into clusters (hives), and for each hive, a query is elected (called queen bee). Once it is elected, all its nodes are potentially cached in the buffer. Based on sorting mechanism, each hive is scheduled. This sorting is based on three main criteria: the query cost, query fan-out and the overlapping accumulated cost. Experimental studies were conducted by the use of a simulator and its results are validated on Oracle11G. The obtained results are encouraging and show the effectiveness of our queen-bee approach.

Actually, we are extending this work to handle queries defined on semantic data warehouses. An interesting issue that should be considered concerns the impact of the buffer management and the query scheduling on the horizontal partitioning problem.

References

1. Ahmad, M., Aboulnaga, A., Babu, S., Munagala, K.: Modeling and exploiting query interactions in database systems. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), pp. 183–192 (2008)

2. Ahmad, M., Aboulnaga, A., Babu, S., Munagala, K.: Interaction-aware scheduling of report-generation workloads. VLDB Journal, 589–615 (2011)
3. Arion, A., Benzaken, V., Manolescu, I., Papakonstantinou, Y.: Structured materialized views for xml queries. In: VLDB, pp. 87–98 (2007)
4. Chakravarthy, U.S., Minker, J.: Multiple query processing in deductive databases using query graphs. In: VLDB, pp. 384–391 (1986)
5. Chipara, O., Lu, C., Roman, G.-C.: Real-time query scheduling for wireless sensor networks. In: RTSS, pp. 389–399 (2007)
6. Chou, H.-T., DeWitt, D.J.: An evaluation of buffer management strategies for relational database systems. In: VLDB, pp. 127–141 (1985)
7. Cornell, D.W., Yu, P.S.: Integration of buffer management and query optimization in relational database environment. In: VLDB, pp. 247–255 (1989)
8. Effelsberg, W., Härder, T.: Principles of database buffer management. ACM Trans. Database Syst. 9(4), 560–595 (1984)
9. Gupta, A., Sudarshan, S., Viswanathan, S.: Query scheduling in multi query optimization. In: IDEAS, pp. 11–19 (2001)
10. Kerkad, A., Bellatreche, L., Geniet, D.: Heuristics for solving the integrated buffer management and query scheduling problem. Technical report, LIAS-ENSMA (2011)
11. Kerkad, A., Bellatreche, L., Geniet, D.: Simultaneous resolution of buffer allocation and query scheduling problems. In: Proceedings of the 6th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS) (2012)
12. Märtens, H., Rahm, E., Stöhr, T.: Dynamic Query Scheduling in Parallel Data Warehouses. In: Monien, B., Feldmann, R.L. (eds.) Euro-Par 2002. LNCS, vol. 2400, pp. 321–331. Springer, Heidelberg (2002)
13. Ou, Y., Härder, T., Jin, P.: CFDC: A Flash-Aware Buffer Management Algorithm for Database Systems. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 435–449. Springer, Heidelberg (2010)
14. Phan, T., Li, W.-S.: Dynamic materialization of query views for data warehouse workloads. In: Proceedings of the International Conference on Data Engineering (ICDE), pp. 436–445 (2008)
15. Roy, P., Ramamritham, K., Seshadri, S., Shenoy, P., Sudarshan, S.: Don't trash your intermediate results, cache 'em. CoRR
16. Sellis, T.K.: Multiple-query optimization. ACM Trans. Database Syst. 13(1), 23–52 (1988)
17. Sethi, R.: Complete register allocation. In: Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, pp. 182–195 (1973)
18. Swami, A.N., Gupta, A.: Optimization of large join queries. In: SIGMOD Conference, pp. 8–17 (1988)
19. Tan, K.-L., Lu, H.: Workload scheduling for multiple query processing. Information Processing Letters 55(5), 251–257 (1995)
20. Thomas, D., Diwan, A.A., Sudarshan, S.: Scheduling and caching in multiquery optimization. In: COMAD, pp. 150–153 (2006)
21. Le, W., Kementsietsidis, A., Duan, S., Li, F.: Scalable multi-query optimization for sparql. In: Proceedings of the International Conference on Data Engineering, ICDE (2012)
22. Yang, J., Karlapalem, K., Li, Q.: Algorithms for materialized view design in data warehousing environment. In: VLDB, pp. 136–145 (1997)
23. Yang, M., Wu, G.: Caching intermediate result of sparql queries. In: WWW (Companion Volume), pp. 159–160 (2011)

Efficient Distributed Parallel Top-Down Computation of ROLAP Data Cube Using MapReduce

Suan Lee[1], Jinho Kim[1], Yang-Sae Moon[1], and Wookey Lee[2]

[1] Department of Computer Science, Kangwon National University,
192-1, Hyoja2-Dong, Chuncheon, Kangwon, Korea
{salee,jhkim,ysmoon}@kangwon.ac.kr

[2] Department of Industrial Engineering, Inha University,
100 Inha-ro, Nam-gu, Incheon, Korea
wookeylee@gmail.com

Abstract. The computation of multidimensional OLAP(On-Line Analytical Processing) data cube takes much time, because a data cube with D dimensions consists of 2^D cuboids. To build ROLAP(Relational OLAP) data cubes efficiently, existing algorithms (e.g., GBLP, PipeSort, PipeHash, BUC, etc) use several strategies sharing sort cost and input data scan, reducing data computation, and utilizing parallel processing techniques. On the other hand, MapReduce is recently emerging for the framework processing a huge volume of data like web-scale data in a distributed/parallel manner by using a large number of computers (e.g., several hundred or thousands). In the MapReduce framework, the degree of parallel processing is more important to reduce total execution time than elaborate strategies. In this paper, we propose a distributed parallel processing algorithm, called MRPipeLevel, which takes advantage of the MapReduce framework. It is based on the existing PipeSort algorithm which is one of the most efficient ones for top-down cube computation. The proposed MRPipeLevel algorithm parallelizes cube computation and reduces the number of data scan by pipelining at the same time. We implemented and evaluated the proposed algorithm under the MapReduce framework. Through the experiments, we also identify factors for performance enhancement in MapReduce to process very huge data.

Keywords: Data Cube, ROLAP, MapReduce, Hadoop, Distributed Parallel Computing.

1 Introduction

Due to the advance of information technology and WWW(World-Wide Web) recently, many applications require to manage a large amount data and to analyze them in online over multi-dimensions. In order to handle these requirements efficiently, there have been a lot of researches on multidimensional data cubes [1]. The data cube is an essential part of OLAP(On-Line Analytical Processing), which maintains aggregate results pre-computed over source data sets. It takes a lot of time to compute a data cube, because each data cube keeps the values aggregated by every possible combination of dimensional attributes. If a source table has T tuples, the cost of $T \times 2^D$ is required to compute a data cube with D dimensions. In order to reduce such high cost problem of the multidimensional cube computation, many algorithms have been proposed [3,4]. These algorithms are classified into several categories such as Relational OLAP(ROLAP), Multidimensional OLAP(MOLAP), and Graph-based Methods [2].

This paper focuses on ROLAP cube computation because it can be easily incorporated into existing DBMSs. GBLP [1], PipeSort [3], PipeHash [3], and BUC [4] are examples of ROLAP cube computation. Some others proposed parallel processing algorithms [5,6] (e.g., RP, BPP, ASL [6]). While these algorithms use parallel processing computers/clusters consisting of several ten CPUs, MapReduce emerging recently can use a large number of computers (e.g., several hundred or thousands more). It becomes a popular framework to handle efficiently a huge volume of data like web-scale data in distributed parallel manner. Several algorithms [10–13] (e.g., MRCube [13]) based on the MapReduce framework have been also developed. However these algorithms use bottom-up approach to compute closed cubes and/or data cubes on holistic measurements. The bottom-up approach has to load a set of data into main memory. In case of computing full cubes rather than closed and iceberg cubes, furthermore, it is less efficient than the top-down approach.

Recently, it is required to analyze and manage extremely large data such as web data and social media. In order to handle these massive data, this paper proposes MRPipeLevel, which is a distributed parallel data cube computation algorithm under the MapReduce framework. The MRPipeLevel is based on the existing PipeSort algorithm known as one of the most efficient ones for top-down ROLAP cube computation. The MRPipeLevel algorithm parallelizes the calculation of cuboids which are in the same level of a cube lattice and should be sorted, and it also reduces the number of data scan by pipelining the computation of several cuboids with the same sorting order at the same time.

In this paper, we implement and evaluate the proposed algorithm through various experiments. We carry out various experiments with large scale high-dimensional data and comparative experiments with MRNaïve, MRGBLP and MRPipeSort (i.e., the MapReduce version of Naïve, GBLP and PipeSort algorithms respectively) which are typical top-down ROLAP data cube algorithms. Through the experiments, we found that the proposed algorithm is more efficient than the others and effective to handle large-scale multidimensional data through the MapReduce framework. We also identify the important factors for performance enhancement in MapReduce to process very huge data.

2 Background

2.1 Data Cube

A data cube consists of measurements and dimensions which are the data to analyze and the analysis criteria, respectively. The cube keeps aggregate values for the GROUP BYs of every possible combination of dimensions. The result for each GROUP BY is called a cuboid, and all of cuboids forms a lattice structure according to their inclusion relationship. Fig. 1 shows a cube lattice structure built for sales data by year, store and item dimensions.

Aggregate functions, which calculate aggregate values stored in the cells of a data cube, can be classified into three types as follows:

– Distributive: $COUNT(), MIN(), MAX(), SUM()$
– Algebraic: $AVG()$, $standard\ deviation()$, $MaxN(), MinN()$, $center\ of\ mass()$
– Holistic: $Median(), Mode(), Rank()$

Among these three types, distributive and algebraic functions could compute lower cuboids by using upper cuboids in a cube lattice structure. In Fig. 1, the $\langle Year \rangle$ cuboid can be computed from the $\langle Year, Store \rangle$ cuboid. For example, if they use measurements as $SUM(Sales)$, the $\langle Year, Store \rangle$ cuboid has $\langle 2012, S1, *, 100 \rangle$, $\langle 2012, S2, *, 81 \rangle$, $\langle 2011, S1, *, 18 \rangle$, $\langle 2011, S2, *, 57 \rangle$, and $\langle 2011, S3, *, 32 \rangle$ cells. These cells can be used to find out $\langle 2012, *, *, 181 \rangle$ and $\langle 2011, *, *, 107 \rangle$ cells for the $\langle Year \rangle$ cuboid. With this inclusion relationship between cuboids, a cuboid can be computed from several cuboids in its upper level. By taking advantage of this inclusion relationship, the cube computation time can be reduced. Several top-down algorithms have been developed by using this concept.

Year	Store	Product	Sales
2012	S1	TV	22
2012	S1	Mobile	78
2012	S2	PC	81
2011	S1	TV	18
2011	S2	PC	57
2011	S3	Mobile	32

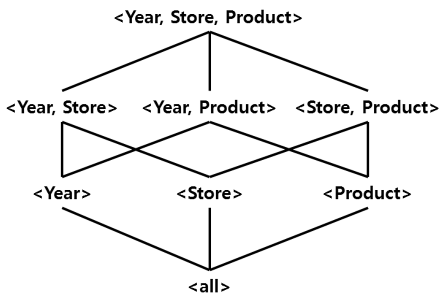


Fig. 1. Examples of source data and cube lattice

2.2 MapReduce

MapReduce is a distributed parallel processing framework for massive data, which was proposed by Google in 2004 and has been applied to various services of Google. The MapReduce is being used in various applications and it becomes a *defacto* standard in large-scale parallel processing fields. This paper utilizes the MapReduce framework and the HDFS(Hadoop Distributed File System) [8] developed by an open source software development project, *Hadoop* [7]. As shown in Fig. 2, the data flow of the MapReduce is as follows: (1) input data is split to deliver to map functions; (2) each map function stores the split input data into its own in-memory buffer, and it partitions, sorts, and spills the input data into disks; (3) the copy phase merges the partitions in the result of each

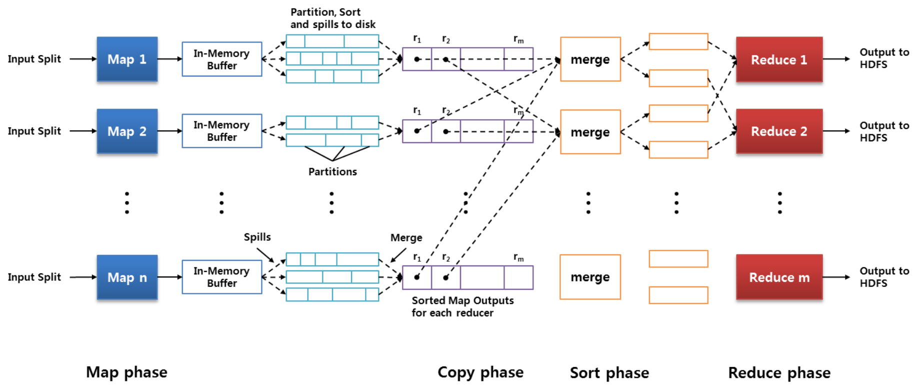


Fig. 2. The data flow on MapReduce

map function; (4) the sort phase delivers the merged results to corresponding reduction functions; and (5) each reduce function processes the delivered data and output its final result to HDFS.

3 The Proposed Algorithm MRPipeLevel

The MRPipeLevel is an algorithm based on the PipeSort which generates the minimum cost sort plan tree from a cube lattice. It computes a set of cuboids sharing the same sort order together with one scan of source table (or another cuboid) by pipelining the computation of these cuboids. The proposed MR-PipeLevel incorporates a distributed parallel processing strategy for the PipeSort in the MapReduce framework which maximizes the degree of parallelism and minimizes the number of MapReduce phases and the number of data scans.

3.1 Sort Tree with Pipeline

The MRPipeLevel builds sort trees and pipelines from a cube lattice structure to compute data cubes efficiently. The sort trees represent the cuboids which don't share the sort order of their parent cuboids thus have to sort these parents to compute them. For example, Fig. 3 includes two sort trees whose root nodes are represented as dotted circles (i.e., ABC and AC cuboids with dotted circles). The tree in the middle of the figure shows that the cuboid BC is computed by sorting the cuboid ABC in the order of AB and the cuboid AC by sorting the ABC in the order of AC. In order to reduce the number of MapReduce phases and to maximize the degree of parallelism, the MRPipeLevel processes each sort tree level by level which all the cuboids in the same level are sorted together from their parents by one MapReduce phase. Thus three MapReduce phases are used to process the sort trees in the example of Fig 3. After computing the cuboids in sort trees, the MRPipeLevel executes the other cuboids with pipelines.

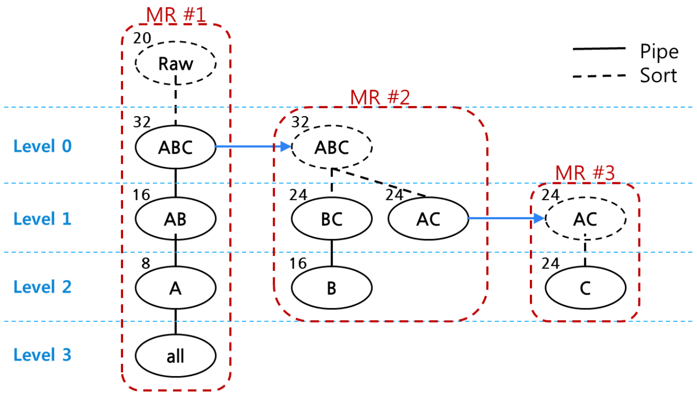
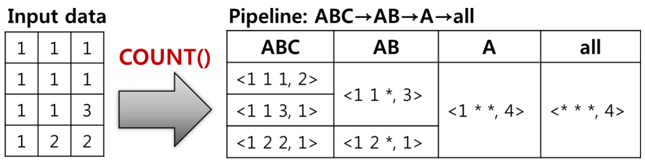


Fig. 3. The example of Cube Execution Pipe Tree

3.2 Pipelines

Pipeline Aggregation. The MRPipeLevel's pipeline technique is a method used in the PipeSort algorithm [3], which computes several child cuboids without sorting if they correspond to their parent cuboid's prefix. As shown in Fig. 4, the AB, A, and $\langle all \rangle$ cuboids can be calculated in the course of calculating the ABC cuboid by the pipeline technique, without sorting and scanning input data repeatedly. When carrying out pipeline aggregation on the first tuple, for example, it computes ABC: $\langle 1\ 1\ 1,\ 1 \rangle$, AB: $\langle 1\ 1\ *,\ 1 \rangle$, A: $\langle 1\ *\ *,\ 1 \rangle$, all: $\langle *\ *\ *,\ 1 \rangle$, and for the second tuple, it does ABC: $\langle 1\ 1\ 1,\ 2 \rangle$, AB: $\langle 1\ 1\ *,\ 2 \rangle$, A: $\langle 1\ *\ *,\ 2 \rangle$, all: $\langle *\ *\ *,\ 2 \rangle$. If aggregating the third tuple, $\langle 1\ 1\ 1,\ 2 \rangle$ is emitted as a resulting cell of the ABC cuboid and it produces ABC: $\langle 1\ 1\ 3,\ 1 \rangle$, AB: $\langle 1\ 1\ *,\ 3 \rangle$, A: $\langle 1\ *\ *,\ 3 \rangle$, all: $\langle *\ *\ *,\ 3 \rangle$. When accepting the fourth tuple, $\langle 1\ 1\ 3,\ 1 \rangle$ and $\langle 1\ 1\ *,\ 3 \rangle$ are emitted as the resulting cells of ABC and AB cuboids respectively. If we use such a pipeline aggregation, all cells of the ABC, AB, A, and $\langle all \rangle$ cuboids can be computed together by scanning input data once. The MRPipeLevel minimizes the computation time by exploiting this pipeline aggregation by distributed processing within a MapReduce phase.



Input data

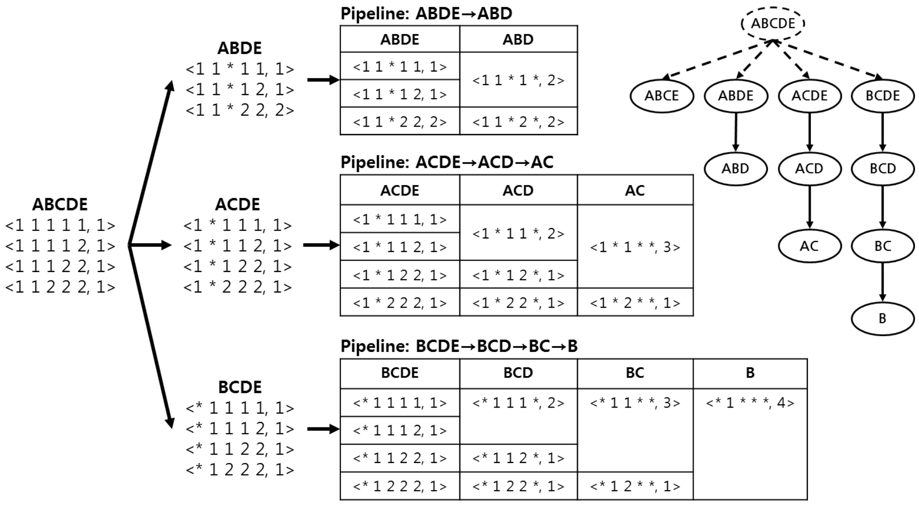
1	1	1
1	1	1
1	1	3
1	2	2

Pipeline: ABC→AB→A→all

ABC	AB	A	all
<1 1 1, 2>	<1 1 *, 3>	<1 * *, 4>	<* * *, 4>
<1 1 3, 1>			
<1 2 2, 1>	<1 2 *, 1>		

Fig. 4. The example of Pipeline aggregation

Multi-pipeline Aggregation. In the MRPipeLevel, multiple pipelines can be simultaneously executed within one MapReduce phase. Fig. 5 shows an example sort plan tree for a 5-dimensional data cube. It contains three aggregation pipelines starting from ABCE, ABDE, ACDE, BCDE cuboids, while these four cuboids can be computed from the top cuboid ABCDE. The MRPipeLevel executes all of these three pipelines in parallel by one MapReduce phase. That is, ABDE, ABD, ACDE, ACD, AC, BCDE, BCD, BC and B cuboids are computed by one MapReduce phase to maximize the degree of parallelism.



Pipeline: ABDE→ABD

ABDE	ABD
<1 1 * 1 1, 1>	<1 1 * 1 *, 2>
<1 1 * 1 2, 1>	
<1 1 * 2 2, 2>	<1 1 * 2 *, 2>

Pipeline: ACDE→ACD→AC

ACDE	ACD	AC
<1 * 1 1 1, 1>	<1 * 1 1 *, 2>	<1 * 1 * *, 3>
<1 * 1 1 2, 1>		
<1 * 1 2 2, 1>	<1 * 1 2 *, 1>	
<1 * 2 2 2, 1>	<1 * 2 2 *, 1>	<1 * 2 * *, 1>

Pipeline: BCDE→BCD→BC→B

BCDE	BCD	BC	B
<* 1 1 1 1, 1>	<* 1 1 1 *, 2>	<* 1 1 * *, 3>	<* 1 * * *, 4>
<* 1 1 1 2, 1>			
<* 1 1 2 2, 1>	<* 1 1 2 *, 1>		
<* 1 2 2 2, 1>	<* 1 2 2 *, 1>	<* 1 2 * *, 1>	

Fig. 5. The example of Multi-Pipeline aggregation

3.3 MapReduce Data Flow of MRPipeLevel

Fig. 6 is an example for a data flow in the process carried out by the MRPipeLevel. As shown in the figure, three MapReduce processes are carried out for three-dimensional input data. At the first MapReduce phase, the map function emits a cell corresponding to the ABC cuboid for the original data. The emitted cells of the ABC cuboid are aggregated for the same cell. At the second MapReduce phase, the map function uses the ABC cuboid as an input to emit cells of the AC and BC cuboids. The reduce function computes the AC and BC cuboids to emit them. At the third MapReduce phase, the map function uses the AC cuboid as an input to emit the C cuboid's cell, and the reduce function computes the C cuboid to emit. At the fourth MapReduce phase, it computes the AB-A-all and B cuboids, which comprise of pipelines. However, because the ABC and BC cuboids used as input data are the sorted data, the reduce function is not operated and the result could be computed immediately.

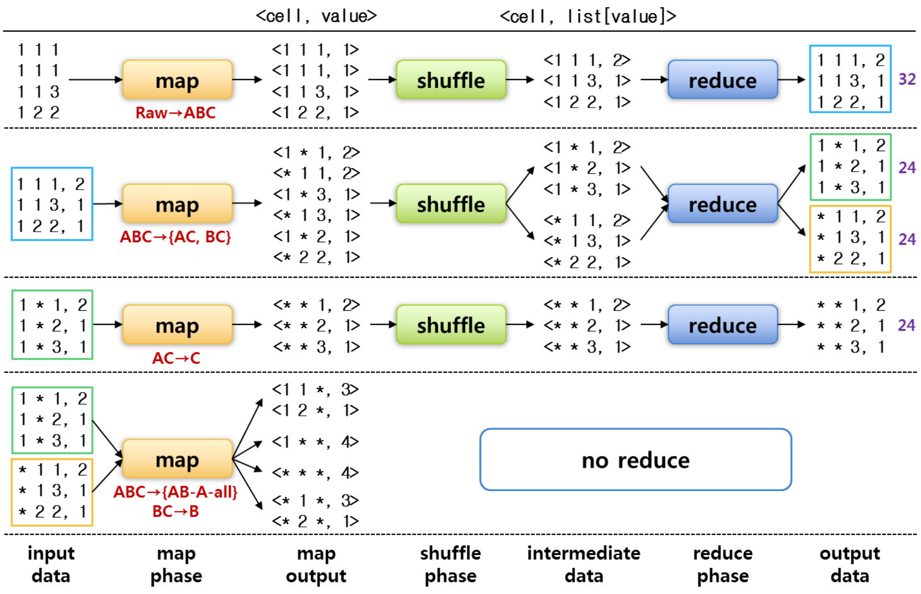


Fig. 6. The example of MRPipeLevel data flow

3.4 MRPipeLevel Algorithm

Algorithm 1 is the MRPipeLevel algorithm which consists of the Map(), Reduce(), MultiPipeMap(), and MRPipeLevel() procedures. However, the MRPipeLevel algorithm includes the features creating sort tree and pipelines from a cube lattice and processing MultiPipeMap() procedure for multi-pipeline aggregation. First, the MRPipeLevel() procedure traverses a cube lattice level by level, finds the cuboids sharing the same prefix and connects them by a pipeline, and constructs SortTrees for the other cuboids with the minimum cost matching. If the whole cube lattice is traversed, all the pipelines computable without sorting and the SortTree requiring sorting are constructed. First, the Map() and Reduce() are carried out for the SortTree, then the MultiPipeMap() is conducted for the pipeline.

MRPipeLevel. MRPipeLevel() procedure traverses a cube lattice level by level and constructs pipelines of cuboids with the same prefix and sort trees of the other cuboids. For each sort tree, Map() and Reduce() functions are executed. Then MultiPipeMap() procedure is invoked for the pipelines.

MultiPipeMap. The MultiPipeMap() procedure processes aggregation for the pipeline introduced in the section 3.2, and it can process the multi-pipeline aggregation together. Looking at the algorithm, there could be one or more pipelines in the pipeline P, and each pipeline's cuboid has a space to store a cell. Comparing each cuboid's cell with the cell coming into as input, measurement is computed to store for the identical cell, the existing cell is emitted for the non-identical cell, and the entered cell is stored.

Algorithm 1. MRPipeLevel

```
 1: procedure MRPIPELEVEL
 2:     for all level $k$ in cube lattice do
 3:         Pipelines $P \leftarrow P \cup$ FindPrefixCuboid$(k+1 \rightarrow k)$
 4:         SortTree $S \leftarrow S \cup$ MinimumCostMatching$(k+1 \rightarrow k)$
 5:     end for
 6:     for all SortSubTree $s_k$ in $S$ do
 7:         $R \leftarrow R \cup$ Map$(s_k) \cup$ Reduce$()$
 8:     end for
 9:     $R \leftarrow R \cup$ MultiPipeMap$(P)$
10:     return $R$
11: end procedure

12: procedure MAP$(S)$                                          ▷ $S$ is a Sort Tree
13:     for all parent cuboid $P$ in $S$ do
14:         for all child cuboid $C$ in $S$ do
15:             $C(cell, measure) \leftarrow P(cell, measure)$
16:             emit $(cell, measure)$
17:         end for
18:     end for
19: end procedure

20: procedure REDUCE
21:     measure $\leftarrow$ aggregation$(m_1, m_2, \cdots, m_n)$        ▷ measure $\ni m_1, m_2, \cdots, m_n$
22:     emit $(cell, measure)$
23: end procedure

24: procedure MULTIPIPEMAP$(P)$                                 ▷ $P$ is a Pipelines
25:     $M \leftarrow$ function$(m_1, m_2, \cdots, m_n)$             ▷ measure $\ni m_1, m_2, \cdots, m_n$
26:     for all cuboid $C$ in $P$ do
27:         if $cell$ = cell $c$ in $C$ then
28:             measure $\leftarrow$ aggregation(measure $\cup$ $M$)
29:         else
30:             emit $(cell, measure)$
31:         end if
32:     end for
33: end procedure
```

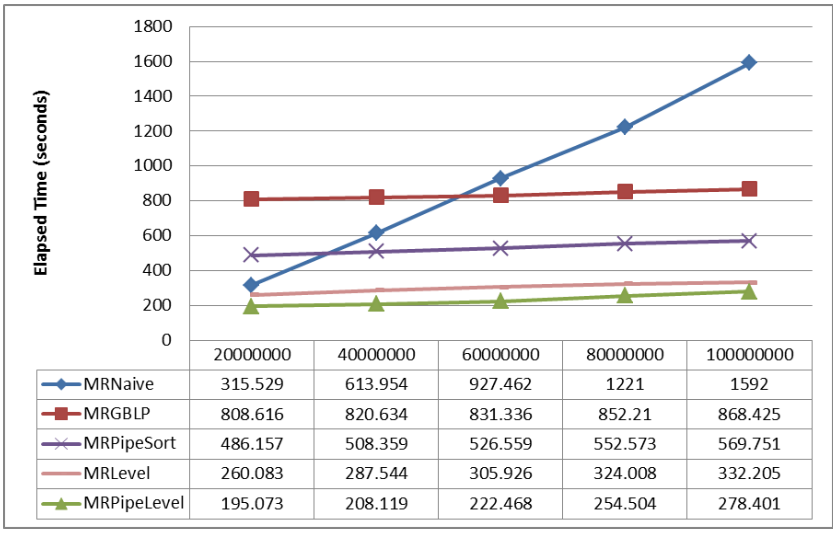
4 Experiments

4.1 Experimental Setup

In the experiments, we used 1 NameNode, 20 DataNodes, and total 21 PCs in a cluster. The NameNode is equipped with Intel Pentium 4 3.0GHz CPU, 1GB RAM, and a 400GB HDD. The DataNodes are equipped with Intel Pentium 4 3.0GHz CPU, 512MB RAM, and a 150GB HDD. The operating system is Ubuntu Linux, the Java version is JDK 1.6, and the MapReduce framework is Hadoop 0.20.2. The network speed is 1G bps. In the experiments, we compare our MRPipeLevel algorithm to four algorithms: MRNaïve algorithm (which is a naïve MapReduce algorithm), MRLevel algorithm (which parallelizes the computation of cuboids in each level), and MRGBLP [12] and MRPipeSort (which are the MapReduce version of GBLP [1] and PipeSort [3,5], respectively).

4.2 Varying the Number of Tuples

Fig. 7 shows the elapsed time for cube computation by varying the number of tuples, where we increase the number of tuples from 20 million to 100 million. As shown in the figure, MRNaïve algorithm execution time increases significantly as the data size increases. For the other algorithms, the difference of cube computation time is not significant. However, MRPipeLevel algorithm shows the smallest execution time and MRLevel algorithm shows a similar rate with MRPipeLevel algorithm.

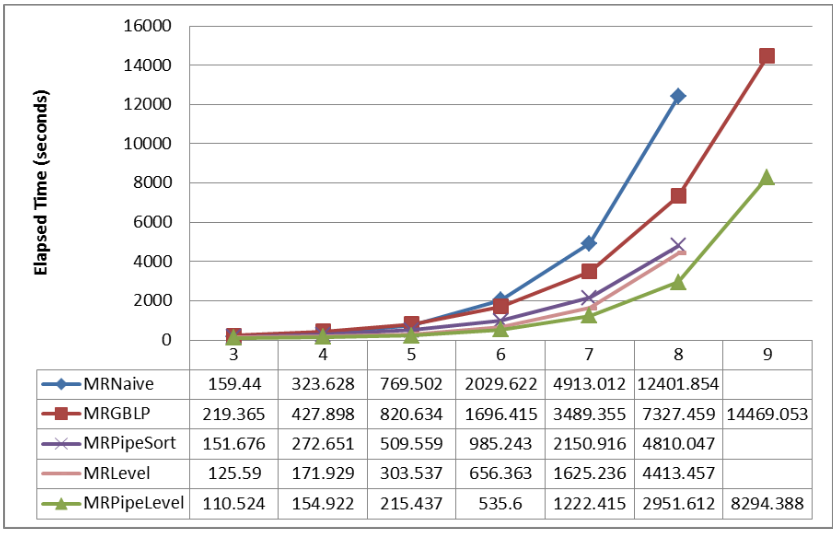


	20000000	40000000	60000000	80000000	100000000
◆ MRNaive	315.529	613.954	927.462	1221	1592
■ MRGBLP	808.616	820.634	831.336	852.21	868.425
× MRPipeSort	486.157	508.359	526.559	552.573	569.751
— MRLevel	260.083	287.544	305.926	324.008	332.205
▲ MRPipeLevel	195.073	208.119	222.468	254.504	278.401

Fig. 7. Elapsed time by varying the number of tuples

4.3 Varying the Number of Dimensions

Fig. 8 shows the elapsed time obtained by varying the number of dimensions. In this experiment, we set the number of tuples to 50 billion and we increase the number of dimensions from three to nine by one. As shown in Fig. 8, MRPipeLevel algorithm is fastest in all dimensions and MRNaïve algorithm is slowest. In the case of 9 dimensions, MRPipeSort and MRLevel algorithm does not work because they emit too much data. MRGBLP algorithm is faster than MRNaïve and MRPipeSort faster than MRLevel. MRGBLP and MRPipeLevel are processed normally up to 9 dimensions or higher dimensions.

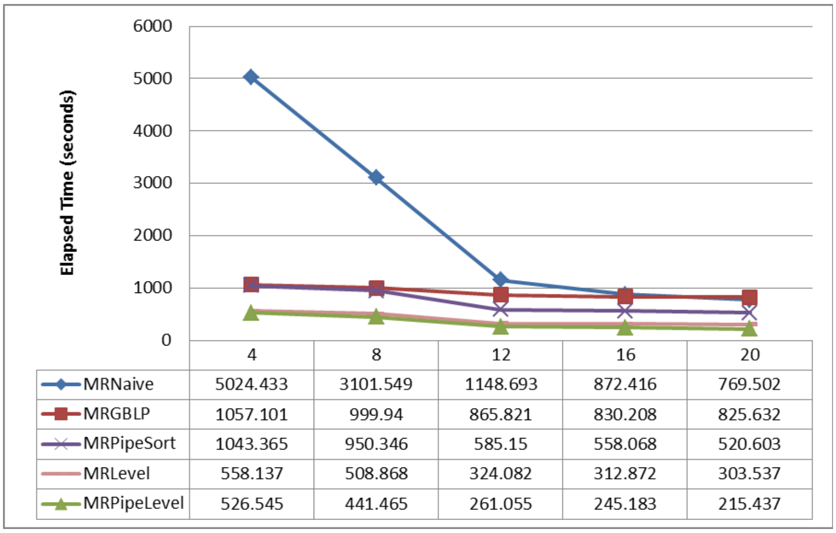


	3	4	5	6	7	8	9
◆ MRNaive	159.44	323.628	769.502	2029.622	4913.012	12401.854	
■ MRGBLP	219.365	427.898	820.634	1696.415	3489.355	7327.459	14469.053
× MRPipeSort	151.676	272.651	509.559	985.243	2150.916	4810.047	
— MRLevel	125.59	171.929	303.537	656.363	1625.236	4413.457	
▲ MRPipeLevel	110.524	154.922	215.437	535.6	1222.415	2951.612	8294.388

Fig. 8. Elapsed time by varying the number of tuples

4.4 Varying the Number of Nodes

Fig. 9 is a comparison among algorithms as increasing the number of nodes. Fig. 9 measured the execution time of each algorithm for 50 billion tuples with 5 dimensions as increasing the number of nodes from 4 to 20. From the result,



	4	8	12	16	20
◆ MRNaive	5024.433	3101.549	1148.693	872.416	769.502
■ MRGBLP	1057.101	999.94	865.821	830.208	825.632
× MRPipeSort	1043.365	950.346	585.15	558.068	520.603
— MRLevel	558.137	508.868	324.082	312.872	303.537
▲ MRPipeLevel	526.545	441.465	261.055	245.183	215.437

Fig. 9. Elapsed time by varying the number of nodes

5 dimensions as increasing the number of nodes from 4 to 20. From the result, we can observe that the execution time of every algorithms decreases as the number of nodes increases. MRPipeLevel algorithm is fastest in all nodes and MRNaïve algorithm reduces the execution time the most significantly. Up to 12 nodes, the execution of every algorithms is reduced in a meaningful degree. But for more than 16 nodes, the computation time is enhanced in smaller degree, because it is getting saturated.

5 Conclusion

There have been many data cube computation algorithms but they are not suitable for a high degree of distributed parallel processing. In this paper, we proposed MRPipeLevel algorithm to effectively compute the data cube using the MapReudce framework utilizing a large number of PCs. The MRPipeLevel algorithm extracts the execution plan of sort tree and pipeline on a cube lattice structure. Cuboids in sort tree minimize scan cost for each level at a time by MapReduce using distributed parallel computation. Cuboids in pipelines are computed at once using sorted cuboids without emitting the data on each node. Thus, MRPipeLevel algorithm reduces the computation time of full cubes using a strategy of parallel processing as much data as possible and reducing the number of data scan.

In this paper, we implement and evaluate the MRPipeLevel algorithm through various experiments. We carry out various experiments with both low-dimensional and high-dimensional data, and we also perform comparative experiments with other MapReduce data cube algorithms which are typical top-down ROLAP data cube computation algorithms. Through the experiments, we show that the proposed MRPipeLevel outperforms the other top-down algorithms in every cases.

Acknowledgements. This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2011-0011824).

References

1. Gray, J., et al.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In: Proc. Int'l Conf. on Data Engineering, New Orleans, LA, pp. 152–199 (February 1996)
2. Konstantinos, M., Stratis, K., Yannis, I., Nikolaos, K.: ROLAP implementations of the data cube. Journal ACM Computing Surveys (CSUR) 39(4), Article No. 12 (2007)
3. Agarwal, S., et al.: On the Computation of Multidimensional Aggregates. In: Proc. the 22nd Int'l Conf. on Very Large Data Bases, Bombay, India, pp. 506–521 (September 1996)
4. Kevin, B., Raghu, R.: Bottom-up Computation of Sparse and Iceberg Cubes. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, Phiiladelphia, PA, pp. 359–370 (June 1999)

5. Dehne, F., Eavis, T., Rau-Chaplin, A.: The cgmCUBE Project: Optimizing Parallel Data Cube Generation for ROLAP. Distributed and Parallel Databases 19(1), 29–62 (2006)
6. Raymond, T.N., Alan, W., Yu, Y.: Iceberg-cube Computation with PC Clusters. In: Proc. Int'l Conf. on Management of Data, ACM SIGMOD, Santa Barbara, CA, pp. 25–36 (June 2001)
7. Hadoop, `http://hadoop.apache.org/`
8. HDFS, `http://hadoop.apache.org/hdfs/`
9. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Communication of the ACM 51(1), 107–113 (2008)
10. Jinguo, Y., Jianging, X., Pingjian, Z., Hu, C.: A Parallel Algorithm for Closed Cube Computation. In: Proc. 7th Int'l Conf. on Computer annd Information Science, Portland, OR, pp. 95–99 (May 2008)
11. Yuxiang, W., Aibo, S., Junzhou, L.: A MapReduceMerge-based Data Cube Construction Method. In: Proc. 9th Int'l Conf. on Grid and Cooperative Computing, Nanjing, China, pp. 1–6 (November 2010)
12. Suan, L., Yang-Sae, M., Jinho, K.: Distributed Parallel Top-Down Computation of Data Cube using MapReduce. In: Proc. 3rd Int'l Conf. on Emerging Databases, Incheon, Korea, pp. 303–306 (August 2011)
13. Arnab, N., Cong, Y., Philip, B., Raghu, R.: Distributed Cube Materialization on Holistic Measures. In: Proc. 27th Int'l Conf. on Data Engineering, Hannover, Germany, pp. 183–194 (April 2011)

Landmark-Join: Hash-Join Based String Similarity Joins with Edit Distance Constraints

Kazuyo Narita, Shinji Nakadai, and Takuya Araki

Cloud System Research Laboratories, NEC Corporation
Kawasaki, Kanagawa 211-8666, Japan
k-narita@ct.jp.nec.com, s-nakadai@az.jp.nec.com, t-araki@dc.jp.nec.com

Abstract. Parallel data processing complicates the completion of string similarity joins because parallel data processing requires the use of a well designed data partitioning scheme. Moreover, efficient verification of string pairs is needed to speed up the entire string similarity join process. We propose a novel framework that addresses these requirements through the use of edit distance constraints. The Landmark-Join framework has two functions that reduce two kinds of search spaces. The first, q-bucket partitioning, reduces the number of verifications of dissimilar string pairs and lowers skewness among buckets. The second, local upper bound calculation, prunes the search space of edit distance to speed up each verification. Experimental results show that Landmark-Join has good parallel scalability and that the two proposed functions speed up the entire string similarity join process.

1 Introduction

String similarity joins enumerate similar string pairs in a given data set on the basis of the distances between strings. They have attracted much attention recently because of the wide range of their potential application, including similar DNA sequence discovery, duplicate record detection, and record linkage. An overall join process is time consuming because it involves much verification (i.e., calculation of the distance between strings and judgment of whether they are similar or not). Various approaches based on different distance measures have been proposed for completing the string similarity join process quickly and efficiently. In particular, we focus on edit distance, which is sensitive to differences between characters.

Since most existing methods for completing the string similarity join process are based on the assumption of centralized processing [1–4, 6, 9–11, 13–15], their processing speed drops drastically when dealing with a large amount of data. The key to overcoming this problem is to use parallel data processing, that is, to partition input data into small buckets so that each bucket can be processed independently. However, parallel string similarity joins suffer from skewness among buckets and string duplication: the number of strings among buckets is often skewed because of data skewness and a string pair may be verified in more than one bucket unlike equi-joins. We thus need a well designed partitioning scheme and an efficient verification technique for use in each bucket. Vernica et al. proposed using MapReduce to compute parallel string similarity joins following data partitioning [12]. However, the token frequency statistics must be calculated in an additional preprocessing step. In addition, they did not focus on making the verification efficient.

We have developed a novel framework that addresses the need for a well designed partitioning scheme and an efficient verification technique. Our **Landmark-Join** framework is based on a *hash-join* approach and has two functions of Landmark-Join that reduce two kinds of search spaces. The first, ***q*-bucket partitioning**, partitions input data into small buckets (without any preprocessing) and thereby reduces the number of candidate similar string pairs. The second, **local upper bound calculation**, prunes candidate shortest paths to speed up each verification. Furthermore, we prove by theoretical analysis that the skew among buckets can be reduced by using parameter q, which is the length of a bucket label. Experimental results show that Landmark-Join has good scalability and that the two proposed functions speed up the entire string similarity join process.

The rest of this paper is organized as follows: The preliminaries are covered in the next section. Section 3 presents the overall framework of Landmark-Join and explains the two search space reduction functions. Section 4 describes the partition and join algorithms. The experimental results are presented and discussed in Section 5. Section 6 discusses related work, and Section 7 summarizes the key points and mentions future work.

2 Preliminaries

We start by defining the problem and describing the techniques used.

Let Σ be a finite domain (a set) of all unique characters and an input data set X be a set of strings on the domain Σ. That is, a string $x\ inX$ is a finite sequence composed of Σ elements. The notation $|\cdot|$ is used to denote set cardinality. Therefore, $|X|$ is the number of strings in X, and $|x|$ represents the length of string x. For $0 \leq k \leq |x|$, $x[k]$ denotes the k-th character in x, where $x[0]$ corresponds to empty character (''). In this paper, $x[0]$ is not counted in any set cardinality (e.g., $|\Sigma|$ and $|x|$). Take x=“abbccd” for example. We know $|x|$= 6$x[0]$=‘’$x[1]$=‘a’and |“”|=0. We use x_i^q to represent a substring in x with a length of q starting from the i-th character. If x=“abbccd”, then x_0^3=“abb”x_4^3=“ccd”and x_2^3=“bbc”. The special substring x_0^k is called *prefix*. Similarly, $x_k^{|x|-k+1}$ is called *suffix*. Because of space constraints, we will often use x_k^∞ in place of $x_k^{|x|-k+1}$, and let x_k^0=“”. The tuple $(\cdot)$ is an instance of a set of attributes where each attribute has a domain. A string pair is represented as a tuple (x, y), where strings x, y are attribute values.

Definition 1. *Given two string sets, X and Y, and distance threshold τ, a string similarity join finds all string pairs and their distance d, $\{((x, y), d)\}$, for which $d \leq \tau$.*

A join is a *self-join* when $X{=}Y$. For the ease of exposition, we will focus on the self-join case in this paper.

Edit Distance. The edit distance between two strings is defined as the minimum number of atomic edit operations needed to transform one string into the other using insertion, deletion, or substitution of a single character. Let $ed(x, y)$ represent the edit distance between strings x and y. We can derive a value for the edit distance between two strings on the basis of dynamic programming [7].

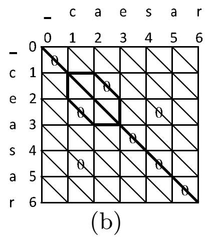
$$ed(x_0^i, y_0^j){=}\begin{cases} \max(|x_0^i|, |y_0^j|) & (i{=}0 \vee j{=}0) \\ \min\left(ed(x_0^{i-1}, y_0^j){+}1, ed(x_0^{i-1}, y_0^{-1}){+}\theta, ed(x_0^i, y_0^{j-1}){+}1 \right) & (\text{otherwise}) \end{cases}$$

Here, θ is 0 if $x[i]{=}y[j]$ and 1 otherwise. Note that there is an upper bound of edit distance known generally: $ed(x, y){\le}\max(|x|, |y|)$.

As an example, consider the strings x=“ceasar” and y=“caesar”. The edit distance between them is calculated using the matrix shown in Figure 1(a). The value in the i-th row and the j-th column (i.e., element (i, j)) corresponds to the edit distance between prefixes $ed(x_0^i, y_0^j)$. The $(6, 6)$-element corresponds to edit distance $ed(x, y)$.

	_	c	a	e	s	a	r
_	0	1	2	3	4	5	6
c	1	0	1	2	3	4	5
e	2	1	1	1	2	3	4
a	3	2	1	2	2	2	3
s	4	3	2	2	2	3	3
a	5	4	3	3	3	2	3
r	6	5	4	4	4	3	2

(a)



(b)

Fig. 1. Example of edit distance calculation

Shortest Path Search on an Edit Graph. Calculating the edit distance of a string pair (x, y) is equivalent to searching for the shortest path by using a weighted grid graph, $G_{x,y}(V, E, w)$. We call such a graph an *edit graph* and use it to simplify our discussion. Let V be a vertex set. A vertex $v_{i,j}{\in}V$ corresponds to a character pair $(x[i], y[j])$ and has weighted edges to the neighbor vertexes. Edges in set E have a weight of 1, but only if $x[i]{=}y[j]$; the corresponding edge from $v_{i-1,j-1}$ to $v_{i,j}$ has a weight 0. The edit distance of the string pair (x, y) corresponds to the shortest path length from the vertex $v_{0,0}$ to $v_{|x|,|y|}$ on the edit graph $G_{x,y}$.

An example edit graph for x=“ceasar” and y=“caesar” is shown in Figure 1(b). The vertex $v_{0,0}$ is the start point, and the vertex $v_{6,6}$ is the end point. Edges attached “0” have a weight 0; the other edges have a weight 1. The three paths in bold are the shortest, and their lengths represent the edit distance, 2. We call an edge with a weight 0 a *0-edge*.

Prefix Filtering. We apply a general filtering technique, *Prefix Filtering*, which is widely used to filter out dissimilar candidate string pairs [4, 13, 15]. It works by excluding a string pair (x, y) from consideration if x and y do not share any characters within their prefixes, which have a length of $\tau+1$, because $ed(\mathrm{x,y}) > \tau$ for a given threshold.

3 Landmark-Join

We first describe the overall framework of Landmark-Join. It is composed of three phases: partition, join, and aggregation (see Figure 2). Let τ be the given distance threshold and q a user-specified parameter. First, in the partition phase, each input string is assigned to at most $\binom{\tau+q}{q}$ different buckets. The details of this partition technique are described in 3.1, *q-bucket partitioning*. Next, in the join phase, the string pairs in each bucket are verified, and a local set of similar pairs is obtained. The verifications for each bucket are speeded up by calculating the *local upper bound* rather than the edit distance. The local upper bound is formally presented in 3.2. Finally, in the aggregation phase, the local results are aggregated, and a complete set of similar pairs is output. Landmark-Join does not generate false positives or false negatives.

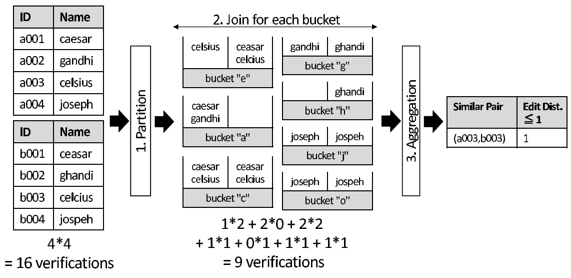


Fig. 2. Landmark-Join framework (τ=1, q=1)

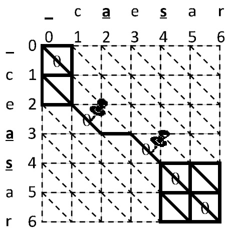


Fig. 3. Search space of candidate paths traversing two pinned landmarks in edit graph

Before providing formal definitions, we give intuitive explanations of q-bucket partitioning and local upper bound calculation. Let x and y be similar strings ($ed(x, y) \leq \tau$), and $G_{x,y}$ be the corresponding edit graph. An important intuition underlying our idea is that the shortest path on $G_{x,y}$ is likely to traverse many 0-edges (as seen in Figure 1(b)). With a user-specified parameter q, we partition the input data into small buckets so that there are only string pairs with paths through at least q 0-edges in each bucket. Note that we assume here that all input strings have a enough long length (more than equal to $\tau + q$), to simplify the explanation. Buckets have a unique string with a length of q as a *bucket label*. Thus, input strings sharing q characters $c_1, \cdots, c_q$ are distributed to a same bucket with the label $c_1 \cdots c_q$. This strategy results in a massive number of dissimilar string pairs being left out of consideration.

We similarly explain the function used to speed up verification. Figure 3 shows the search space for the edit graph of a pair "caesar" and "ceasar" assigned to a bucket labeled "as" ($q = 2$). That is, "caesar" and "ceasar" may be similar because they share the label components 'a' and 's'. Thus, in that bucket, only those paths that pass pinned 0-edges $(v_{2,1}, v_{3,2})$ and $(v_{3,3}, v_{4,4})$ are checked as shortest path candidates. The other paths (dashed-lined edges) are ignored. The local shortest path with *led* length from the vertex $v_{0,0}$ to the vertex $v_{6,6}$ can be derived by using led=ed("ce", "c")+ed("", "e")+ed("ar", "ar"). This calculation is less expensive than that of edit distance $ed(x, y)$ because the search space is obviously much smaller. In this paper, we call such a pinned 0-edge a landmark, i.e., $(v_{2,1}, v_{3,2})$, and $(v_{3,3}, v_{4,4})$ are landmarks.

Of course, a local shortest path is not always the shortest one on an edit graph $G_{x,y}$. That is, it may be that $ed(x, y) \leq led$. However, at least one of the local shortest paths is the globally shortest path because all global shortest paths of similar string pairs necessarily pass many 0-edges, i.e., at least q 0-edges.

3.1 *q*-Bucket Partitioning

We now formally define q-bucket partitioning, i.e., the partition phase in the Landmark-Join framework. We first explain how to assign strings with a length more than equal to τ+q to buckets. Then we explain the case of short strings.

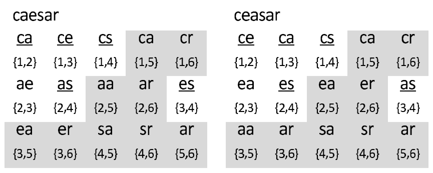
Definition 2. *Let s be a bucket label, which is a sequence of q characters. The label s is detected from a string x if there exists a position list with a size of q such that $\mathcal{P}_{x;s}$=$\{i_1, \ldots, i_q | s[1]$=$x[i_1] \wedge \cdots \wedge s[q]$=$x[i_q]\}$.*

Given an input data set, the framework first traverses each string x, detects all bucket labels with q length from x, and assigns x to all buckets corresponding to those labels. Because a string x has $\binom{|x|}{q}$ bucket labels in total, a significant number of bucket labels will be detected for a string if $|x|$ is large. To reduce detection cost, we limit the number of bucket labels detected for a string by using the principle of Prefix Filtering and thus derive a following pruning technique.

Lemma 1. Bucket Label Pruning *If $ed(x, y) \leq \tau$ holds for strings x and y, their prefixes with a length of $\tau + q$ share at least one bucket label.*

Therefore, we detect $\binom{\tau+q}{q}$ bucket labels not for a full string but for its prefix, $x_0^{\tau+q}$, and assign x to at most $\binom{\tau+q}{q}$ buckets corresponding to the detected labels. In many cases, τ+q is small enough relative to $|x|$ so that Bucket Label Pruning effectively reduces the cost of bucket label detection.

As an example of bucket label detection, we again consider strings "caesar" and "ceasar," this time assuming τ=2 and q=2. Figure 4 lists all $\binom{6}{2}$=15 bucket labels for full strings and the corresponding position lists. Identical labels may be detected from a string more than once, for example, "ca" is detected twice from "caesar" with position lists {1,2} and {1,5}. The labels in gray areas are ignored by Bucket Label Pruning because they are not included in the prefix with τ+q length. Ignoring the gray areas, we can see that "caesar" and "ceasar" will be verified in five buckets since they share five unique labels (underlined).



caesar				
ca {1,2}	ce {1,3}	cs {1,4}	ca {1,5}	cr {1,6}
ae {2,3}	as {2,4}	aa {2,5}	ar {2,6}	es {3,4}
ea {3,5}	er {3,6}	sa {4,5}	sr {4,6}	ar {5,6}

ceasar				
ce {1,2}	ca {1,3}	cs {1,4}	ca {1,5}	cr {1,6}
ea {2,3}	es {2,4}	ea {2,5}	er {2,6}	as {3,4}
aa {3,5}	ar {3,6}	sa {4,5}	sr {4,6}	ar {5,6}

Fig. 4. Bucket labels for "caesar" and "ceasar" (τ=2, q=2)

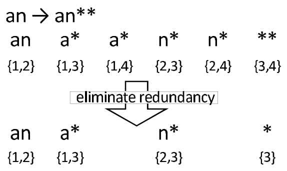


Fig. 5. Bucket labels for "An" (τ=2, q=2)

Bucket Assignment for Short Strings. Here we consider how to assign strings with a length less than τ+q to buckets. This can be achieved in a manner similar to that above.

We start by introducing a special character, '*', that does not belong to the domain Σ. For a string x, if $|x|<\tau+q$, a temporal string x' with $\tau+q$ length is generated by appending $\tau+q-|x|$ instances of this special character to the tail of x, and bucket labels with q length are generated from x'. Any successive '*' (e.g., "***") in a label are considered to be one '*', and elements i in a position list are eliminated such that $i>|x|+1$ because they are redundant information. Finally, x is assigned to appropriate buckets in accordance with the labels detected. This approach does not violate our basic idea that the shortest path for a similar string pair pass at least q 0-edges, so all similar string pairs always meet in at least one bucket.

We present an example in Figure 5, where τ=2 and q=2. Six labels are detected for a temporal string "an**" by appending two special characters to the tail of the original string "an". After eliminating redundancy, we get four bucket labels from "an". Theoretically, at most $\sum_{k=max(0,|x|-\tau)}^{min(q,|x|)} \binom{|x|}{k}$ bucket labels are detected from a string x with a length less than τ+q.

Analysis of Bucket Skewness. String data sets are often skewed since the strings follow a Zipf-like distribution. Thus, bucket sizes are often skewed. While use of a large q results in the duplication of many strings, it also results in a huge number of buckets and thus reduces the skewness among buckets. How does it affect joining cost? Let S_l be a set of all bucket labels which have l characters except the special character '*'. The occurrence probability of a label $s \in S_l$ is approximated by the product of a label length weight u_l and $\frac{1}{q+1}\prod_{k=1}^{l} p(s[k])$, where $p(\cdot)$ is a frequency distribution of characters in input data. The number of strings in a bucket with the label s is approximated by $m_s \approx \frac{u_l}{q+1}\prod_{k=1}^{l} p(s[k])\binom{\tau+q}{q}|X|$. Let $\mathcal{N}_q$ be $\binom{\tau+q}{q}|X|$. The theoretical cost of joining entire buckets is approximated as follows.

$$\sum_{l=0}^{q}\sum_{s\in S_l} m_s^2 \approx \sum_{l=0}^{q}\sum_{s\in S_l} \mathcal{N}_q^2 \frac{u_l^2}{(q+1)^2}\prod_{k=1}^{l} p(s[k])^2 = \frac{\mathcal{N}_q^2}{(q+1)^2}\sum_{l=0}^{q} u_l^2 \prod_{k=1}^{l}\sum_{c\in\Sigma} p(c)^2$$
$$= \frac{\mathcal{N}_q^2}{(q+1)^2}\sum_{l=0}^{q} u_l^2 \alpha^l \qquad (1)$$

We denote by α a constant $\sum_{c\in\Sigma} p(c)^2$. For u_l, we use a Binomial distribution for string lengths in input data, $u_l = bin(l)$, if $l<q$; otherwise, $u_l = \sum_{k=q}^{\infty} bin(k)$. Note that $0 \le l \le q$.

Figure 6 plots Formula 1 for three synthetic data sets with different skewness where we apply a Zipf distribution to $p(\cdot)$. It shows that the total joining cost increases with the data skewness. Moreover, an appropriate value of the parameter q reduces the joining cost because a vast number of buckets lowers skewness among buckets. However, too large q duplicates so many strings and thus takes much cost for joining. Optimizing q is left for future work.

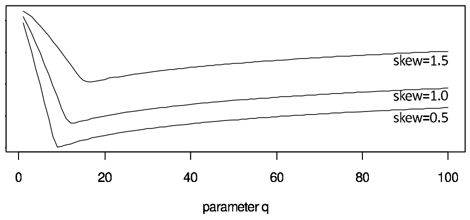


Fig. 6. Theoretical joining cost for three synthetic data sets with different skews

3.2 Local Upper Bound Calculation

We now formally define the local upper bound calculation, which corresponds to the join phase in the Landmark-Join framework and speeds up each verification. To simplify the explanation, we focus on strings with a length more than equal to $\tau + q$. However, we can discuss the case of short strings in a similar fashion.

As mentioned above, we try to find, for each bucket, the local shortest path that passes at least q 0-edges (landmarks) on the edit graph.

Definition 3. *If a label s is detected for a string x in accordance with a position list $\mathcal{P}_{x;s}$, we derive an interval list $\mathcal{I}_{x;s}$ such as $\mathcal{I}_{x;s}[k]=\mathcal{P}_{x;s}[k]-\mathcal{P}_{x;s}[k-1]-1$, where $1 \le k \le q-1$ and $\mathcal{I}_{x;s}[0]=\mathcal{P}_{x;s}[0]-1$. We use notations $\mathcal{I}_x$ and $\mathcal{P}_y$ instead of $\mathcal{I}_{x;s}$ and $\mathcal{P}_{x;s}$ if it is obvious from the context.*

We define the local shortest path length (i.e., local edit distance) of a string pair (x, y) in a bucket with the label s as

$$led(x,y;s)=ed(x_0^{\mathcal{I}_x[0]}, y_0^{\mathcal{I}_y[0]})+\sum_{k=1}^{q-1} ed(x_{\mathcal{P}_x[k-1]}^{\mathcal{I}_x[k]}, y_{\mathcal{P}_y[k-1]}^{\mathcal{I}_y[k]})+ed(x_{\mathcal{P}_x[q-1]}^{\infty}, y_{\mathcal{P}_y[q-1]}^{\infty}). \quad (2)$$

Lemma 2. *Let S be the set of detected bucket labels shared by a similar string pair (x, y); the edit distance is specified as $ed(x, y)=\min_{s\in S} led(x, y; s)$.*

To reduce search spaces more effectively, we introduce a novel upper bound for the local edit distance.

Definition 4. *For a string pair (x, y) in a bucket with the label s, the local upper bound, $lub(x, y; s)$, for the local edit distance, $led(x, y; s)$, is*

$$lub(x,y;s)=\sum_{k=0}^{q-1} \max(\mathcal{I}_x[k], \mathcal{I}_y[k]) + ed(x_{\mathcal{P}_x[q-1]}^{\infty}, y_{\mathcal{P}_y[q-1]}^{\infty}). \quad (3)$$

Lemma 3. *From Lemma 2, $ed(x,y) \leq led(x,y;s) \leq lub(x,y;s)$ for each $s \in S$.*

Consider a label set S' such that $ed(x,y) = led(x,y;s')$ for all $s' \in S'$. There exits at least one label $s'' \in S'$ such that $ed(x^{\mathcal{I}_x[k]}_{\mathcal{P}_x[k-1]}, y^{\mathcal{I}_y[k]}_{\mathcal{P}_y[k-1]}) = \max(\mathcal{I}_x[k], \mathcal{I}_y[k])$ for any k. That is, $lub(x,y;s'') = led(x,y;s'') = ed(x,y)$. Thus, we can get the edit distance from the local upper bound for each bucket.

Theorem 1. *The edit distance can be derived as $ed(x,y) = \min_{s \in S} lub(x,y;s)$.*

Thus, we derive an additional pruning method.

Lemma 4. LUB Pruning *Given the threshold τ and the parameter q, a string pair (x,y) can be pruned under the bucket labeled s if $\sum_{k=0}^{q-1} \max(\mathcal{I}_x[k], \mathcal{I}_y[k]) > \tau$ or $ed(x^{\infty}_{\mathcal{P}_x[q-1]}, y^{\infty}_{\mathcal{P}_x[q-1]}) > \tau - \sum_{k=0}^{q-1} \max(\mathcal{I}_x[k], \mathcal{I}_y[k])$.*

Again we take as an example strings "ceasar" and "caesar". Figure 7 presents the search spaces for the local upper bound for the five buckets to which strings are assigned. More paths can be ignored than in the search spaces for the local edit distance. Moreover, the searching can often be terminated earlier due to the use of LUB Pruning.

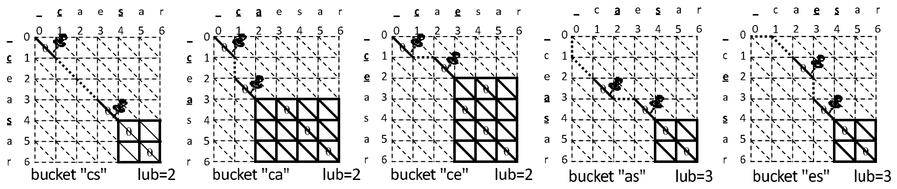


Fig. 7. Search spaces for local upper bound of "caesar" and "ceasar" (q=2)

4 Algorithms

Three algorithms are used in Landmark-Join corresponding to the partitioning phase, join phase, and aggregation phase (Figure 2). These algorithms are given string data, a distance threshold, and a parameter q and output the set of similar string pairs. One of the many existing algorithms can be used for the join phase. Here, as a simple example, a nested-loop algorithm is used.

Algorithm 1 shows the pseudo code for the **partition** function. For each input string x, it detects all position lists by using Bucket Label Pruning. Then, for each position list $\mathcal{P}$, it creates the corresponding label and interval list $\mathcal{I}$ and appends a tuple of the string x, $\mathcal{I}$, and the tail position of $\mathcal{P}$ to the corresponding bucket.

The **simjoin** function is called for each bucket process and finds all local similar string pairs in the bucket. Here, we use the simplest algorithm, nested-loop, in our framework (Algorithm 2). It receives each pair of $(x, \mathcal{I}_x, p_x)$ and $(y, \mathcal{I}_y, p_y)$ from a given bucket and verifies them with LUB Pruning. If the verification returns true, it appends a pair of the string pair and the local upper bound to the set of the local results.

Algorithm 1. partition

```
Input: string data $X$threshold $\tau$parameter $q$
Output: set of buckets $BucketsSet$
 1. $BucketsSet \leftarrow \emptyset$
 2. for all $x \in X$ do
 3.     $PosListsSet \leftarrow \emptyset$
 4.     $PosListsSet \leftarrow detect\_all\_pos\_lists\_with\_label\_pruning(x)$
 5.     for all $\mathcal{P} \in PosListsSet$ do
 6.         $label \leftarrow make\_label(x, \mathcal{P})$;  $\mathcal{I} \leftarrow make\_interval\_list(\mathcal{P})$
 7.         $Bucket \leftarrow BucketsSet.getBucket(label)$
 8.         append$(Bucket, (x, \mathcal{I}, \mathcal{P}[|\mathcal{P}| - 1]))$
 9.     end for
10. end for
```

Algorithm 2. simjoin (self-join)

```
Input: bucket $Bucket$, threshold $\tau$, parameter $q$
Output: a local set of similar string pairs $LocalResult$
 1. $LocalResult \leftarrow \emptyset$
 2. for all $(x, \mathcal{I}_x, p_x) \in Bucket$ do
 3.     for all $(y, \mathcal{I}_y, p_y) \in Bucket$ do
 4.         $interval \leftarrow \sum_{k=0}^{q-1} \max(\mathcal{I}_x[k], \mathcal{I}_y[k])$
 5.         if Verify$(x_{p_x}^{\infty}, y_{p_y}^{\infty}, \tau - interval)$ is true then
 6.             append$\big(LocalResult, \big((x, y), interval + ed(x_{p_x}^{\infty}, y_{p_y}^{\infty})\big)\big)$
 7.         end if
 8.     end for
 9. end for
```

Finally, the **aggregate** function receives all local result sets and outputs a complete set of similar string pairs. Any multiple elements including an individual string pair are deleted except for the one with the minimum local upper bound.

5 Performance Evaluation

First, we show the effectiveness of two search space reductions. Second, we present the parallel scalability. For the former evaluation, we executed a serial runtime in a single machine, which had Ubuntu 10.10 OS, 3.60 GHz CPU, and 16 GB RAM. For the latter evaluation, we executed a parallel runtime in a four-node cluster. Each node consisted of Intel Xeon 2.00-GHz 4 Core, 12-GB memory, 178-MB/s HDD bandwidth, and 64-bit Linux (v2.6.26-2) and connected it to a 1-Gbps network (thus the maximum number of nodes is regarded as 16). The parallel runtime was implemented based on a MapReduce framework, where the first Map tasks executed the partitioning function, the first Reduce tasks corresponded to the join function, the second Map tasks did nothing (empty tasks) and the second Reduce tasks aggregated results. We implemented serial and parallel runtimes in C++ and used a g++ 4.4.5 compiler with the -O3 option.

Data Sets. We used two synthetic data sets for the ease of analysis on data skewness impacts. In both data sets, the character frequency followed a Zipf distribution and the string length followed a Binomial distribution. The Zipf distribution had two parameters, a domain size $|\Sigma|$ and a skewness s, and the Binomial distribution had two parameters, the number of trials n and the success probability of each trial p. One data set, SkewH, was generated using s=1.5, $|\Sigma|$=94, n=100, and p=0.2. The other one, SkewL, was generated using s=0.5, $|\Sigma|$=94, n=100, and p=0.2. Each had 1 million strings.

Search Space Reduction. Figure 8 shows the q-bucket partitioning can reduce the total processing cost of Landmark-Join. L/H_time corresponds to the total processing time to complete the partition, join and aggregation phases, L/H_buckets means the number of generated buckets and L/H_strings equals to the number of partitioned (duplicated) strings for SkewL/SkewH. Both L_time and H_time are similar with the theoretical joining cost in Figure 6, where partition times were proportional to the number of duplicated strings and aggregation times depended on the number of similar string pairs. When the parameter q is large, the join cost of each bucket decreases because each bucket has small number of strings, while the numbers of buckets and duplicated strings increase dramatically.

Figure 9 presents the local upper bound calculation can reduce the join processing time. L/H_All means the join processing time of the complete method, while L/H_NO_LUB means that of the method without using LUB Pruning. According to Figure 8, we provided q = 3 to process SkewL and q = 8 to process SkewH. As the distance threshold increases, redundant string pairs also increases and thus LUB Pruning prunes them more effectively.

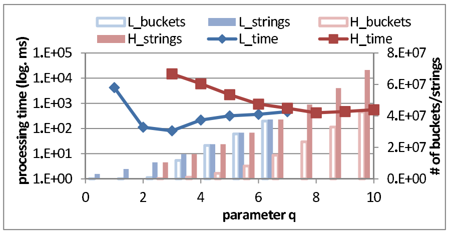


Fig. 8. Effect of partitioning (τ = 2)

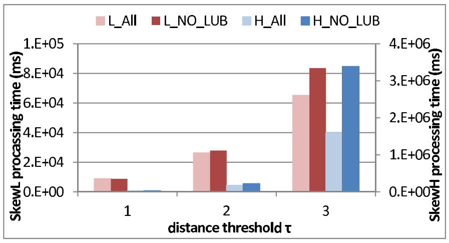


Fig. 9. Effect of LUB Pruning

Parallel Scalability. We measured the parallel processing times with changing the number of nodes (i.e., the each number of tasks for four Map/Reduce) and derived the ratios of the total processing time on N-nodes to the total processing time on 1-node. Figure 10 and 11 show the ratios of SkewL and SkewH. We provided q = 3 to SkewL and q = 8 to SkewH, and changed the distance threshold τ from 1 to 3. Generally, the growth of τ enlarges a processing time of a string similarity join. However, the ratios of the processing time do not show significant change from τ=1 to τ=3. We thus know that the parallel scalability does not depend on the change of τ. Moreover, our framework seems robust against data skewness since both Figure 10 and 11 show similar processing time ratios despite that corresponding data sets have different skewnesses.

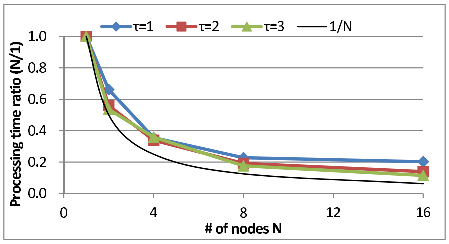


Fig. 10. Processing time ratio of N-nodes to 1-nodes for SkewL ($q = 3$)

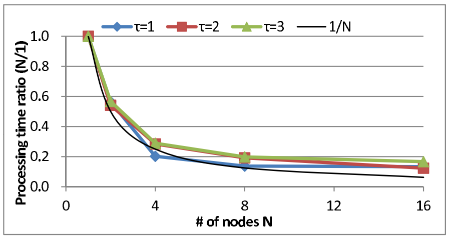


Fig. 11. Processing time ratio of N-nodes to 1-nodes for SkewH ($q = 8$)

6 Related Work

Many studies on similarity string joins used a filter-and-refine approach [1–4, 6, 9–11, 14, 15]. Filter-and-refine approaches utilized string tokens such as n-grams and variants as signatures and filtered candidate similar string pairs to speed up the entire join process. Another approach is to use a trie. J. Wang et al. proposed inserting all input strings into a trie and then finding similar pairs by using trie traversal [13]. These approaches process joins faster when their intermediate data (inverted indexes, tries, etc.) fit into workspace memory. However, if there is a large amount of input data or the threshold is large, their processing speed drops drastically because a huge amount of intermediate data causes frequent cache misses or swap-outs, and they cannot handle such cases.

Parallel processing for equi-joins was intensively studied in the 1980s, as exemplified by the parallel GRACE hash-join [8]. With respect to θ-joins, partitioning methods for joining records with a numerical join key were researched in the 1990s [5]. However, parallel processing for similarity string joins has not been well studied. One of the few such studies proposed applying set similarity joins to MapReduce [12]. However, the partitioning needs statistics on token frequency to partition the input data as an additional preprocessing step. In addition, they did not focus on making the verification efficient.

7 Conclusion

We have presented a novel framework for processing string similarity joins through the use of edit distance constraints. The Landmark-Join framework is based on a hash-join strategy. It uses a q-bucket partitioning function to reduce the number of verifications of dissimilar string pairs and a local upper bound calculation function to prune the search space of the edit distance and thereby speed up each verification. Experimental results show that Landmark-Join is robust against data skewness and has good parallel scalability, and that the two proposed functions speed up the entire join process.

Future work includes an optimization of the parameter q and an investigation of a recursive hash-join approach in which buckets are partitioned into sub-buckets if they have too many strings.

Acknowledgments. This work was carried out by the Green IT Project of the Ministry of Economy Trade and Industry (METI) R&D of Industrial Science and Technology Frontier Program supported by New Energy and Industrial Technology Development Organization (NEDO).

References

1. Arasu, A., Ganti, V., Kaushik, R.: Efficient exact set-similarity joins. In: VLDB, pp. 918–929 (2006)
2. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: WWW, pp. 131–140 (2007)
3. Bocek, T., Hunt, E., Stiller, B.: Fast similarity search in large dictionaries. Technical Report ifi-2007.02, Department of Informatics, University of Zurich (April 2007)
4. Chaudhuri, S., Ganti, V., Kaushik, R.: A primitive operator for similarity joins in data cleaning. In: ICDE, p. 5 (2006)
5. DeWitt, D.J., Naughton, J.F., Schneider, D.A.: An evaluation of non-equijoin algorithms. In: VLDB, pp. 443–452 (1991)
6. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Srivastava, D.: Approximate string joins in a database (almost) for free. In: VLDB, pp. 491–500 (2001)
7. Kim, S.-R., Park, K.: A Dynamic Edit Distance Table. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 60–68. Springer, Heidelberg (2000)
8. Kitsuregawa, M., Ichiro Tsudaka, S., Nakano, M.: Parallel grace hash join on shared-everything multiprocessor: Implementation and performance evaluation on symmetry s81. In: ICDE, pp. 256–264 (1992)
9. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: ICDE, pp. 257–266 (2008)
10. Li, C., Wang, B., Yang, X.: Vgram: Improving performance of approximate queries on string collections using variable-length grams. In: VLDB, pp. 303–314 (2007)
11. Li, G., Deng, D., Wang, J., Feng, J.: Pass-join: A partition-based method for similarity joins. In: PVLDB, pp. 253–264 (2011)
12. Vernica, R., Carey, M.J., Li, C.: Efficient parallel set-similarity joins using mapreduce. In: SIGMOD Conference, pp. 495–506 (2010)
13. Wang, J., Li, G., Feng, J.: Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. In: PVLDB, vol. 3(1), pp. 1219–1230 (2010)
14. Wang, W., Xiao, C., Lin, X., Zhang, C.: Efficient approximate entity extraction with edit distance constraints. In: SIGMOD Conference, pp. 759–770 (2009)
15. Xiao, C., Wang, W., Lin, X.: Ed-join: an efficient algorithm for similarity joins with edit distance constraints. In: PVLDB, vol. 1(1), pp. 933–944 (2008)

Incremental Itemset Mining Based on Matrix Apriori Algorithm

Damla Oguz and Belgin Ergenc

Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey
{damlaoguz,belginergenc}@iyte.edu.tr

Abstract. Databases are updated continuously with increments and re-running the frequent itemset mining algorithms with every update is inefficient. Studies addressing incremental update problem generally propose incremental itemset mining methods based on Apriori and FP-Growth algorithms. Besides inheriting the disadvantages of base algorithms, incremental itemset mining has challenges such as handling i) increments without re-running the algorithm, ii) support changes, iii) new items and iv) addition/deletions in increments. In this paper, we focus on the solution of incremental update problem by proposing the Incremental Matrix Apriori Algorithm. It scans only new transactions, allows the change of minimum support and handles new items in the increments. The base algorithm Matrix Apriori works without candidate generation, scans database only twice and brings additional advantages. Performance studies show that Incremental Matrix Apriori provides speed-up between 41% and 92% while increment size is varied between 5% and 100%.

Keywords: Itemset Mining, Matrix Apriori, Incremental Itemset Mining.

1 Introduction

Association rule mining, which was introduced by [1], has become a popular research area due to its applicability in various fields such as market analysis, forecasting and fraud detection. Given a market basket dataset, association rule mining discovers all association rules such as "A customer who buys item X, also buys item Y at the same time". These rules are displayed in the form of $X \rightarrow Y$ where X and Y are sets of items that belong to a transactional database. Support of association rule $X \rightarrow Y$ is the percentage of transactions in the database that contain $X \cup Y$. Association rule mining aims to discover interesting relationships and patterns among items in a database. It has two steps; finding all frequent itemsets and generating association rules from the itemsets discovered. Itemset denotes a set of items and frequent itemset refers to an itemset whose support value is more than the threshold described as the minimum support.

Since the second step of the association rule mining is straightforward, the general performance of an algorithm for mining association rules is determined by the first step [2]. Therefore, association rule mining algorithms commonly concentrate on finding frequent itemsets. Apriori and FP-Growth are known to be the two important algorithms each having different approaches in finding frequent itemsets [3-4]. The Apriori Algorithm uses Apriori Property in order to improve the efficiency of the level-wise generation of frequent itemsets. On the other hand, candidate generation and multiple database scans are the drawbacks of the algorithm. FP-Growth comes with an approach that creates signatures of transactions on a tree structure to eliminate the need of database scans and outperforms compared to Apriori. A recent algorithm called Matrix Apriori which combines the advantages of Apriori and FP-Growth was proposed [5]. The algorithm eliminates the need of multiple database scans by creating signatures of itemsets in the form of a matrix. It provides a better overall performance than FP-Growth [6]. Although all these algorithms handle the problem of association rule mining, they ignore the dynamicity of the databases. When new transactions arrive, the entire process needs to be done from the beginning. The solution to this problem is incremental itemset mining, in which the idea is to keep frequent itemsets up-to-date with the arrival of increments to the database.

First group of incremental itemset mining algorithms are Apriori based [7-9]. The Fast UPdate Algorithm provides a solution to incremental association rule mining problem [7]. The frequencies of itemsets are determined by comparing itemsets in the original database and new transactions. The main goal of the algorithm is to reduce number of the candidate sets. In [8], a new incremental association rule mining algorithm was proposed. It uses a trie-like tree structure that stores the frequent 1-itemsets and 2-itemsets with their supports in the database. When new transactions arrive, frequent 1-itemsets and 2-itemsets are found and the trie is updated accordingly. Another algorithm in this group also avoids scanning the entire database when new transactions arrive, however, it does not handle minimum support change [9].

Second group of incremental itemset mining algorithms are FP-Growth based; they construct FP-tree incrementally. One of them was introduced by [10], which constructs a fast updated FP-tree (FUFP-tree) structure. It is similar to FP-tree, while the links between the nodes are bi-directional and it handles insertion and deletion of items. A new method, which constructs the FP-tree by assuming the minimum support as 1, was proposed by [11]. Therefore, when new transactions are added to the database, the tree is updated with scanning the increments twice.

Adaptation of the first and the second group of incremental itemset mining algorithms on stream data is also an intensive research area [12-13]. Analysis of the challenges brought by stream data and proposing a model for incremental itemset mining over stream data are beyond the scope of this paper.

Generally speaking, incremental frequent itemset mining has four challenges: i) adapting a base frequent itemset finding algorithm as to handle increments, ii) allowing new item appearances in increments, iii) being flexible to support changes during entire process and iv) handling deletions as well as additions in increments.

In this study, an incremental itemset mining algorithm, which is called Incremental Matrix Apriori, is proposed to provide solutions to these challenges. Since the base algorithm Matrix Apriori works without candidate generation and avoids multiple database scans, the proposed incremental algorithm inherits performance advantages. Performance evaluations show that the Incremental Matrix Apriori Algorithm is significantly faster than re-running the Matrix Apriori Algorithm when new increments arrive.

This paper is organized as follows; Section 2 reviews incremental itemset mining algorithms. Section 3 presents the proposed algorithm for incremental itemset mining. In Section 4, test results and performance evaluations are discussed. Finally, this paper is concluded in Section 5 by raising several issues for future work.

2 Related Work

Association rule mining aims to discover frequent itemsets in a dataset. The Apriori Algorithm, which was proposed by [3], is one of the best-known association rule mining algorithms [14]. It uses prior knowledge of frequent itemset properties and runs an iterative approach called level-wise search. That is, *k*-itemsets are used to explore (*k*+1)-itemsets (they are called candidate itemsets before testing them against the database) by eliminating the candidates that do not satisfy the minimum support. The FP-Growth Algorithm handles the weaknesses of Apriori which are multiple scans of the database and candidate generation. It finds frequent itemsets without candidate generation by using a tree structure, called FP-tree, where each node stores an item with its number of occurrence in the database and a link to the next node [15].

The Matrix Apriori Algorithm offers a simple and efficient solution to the association rule mining. Database scan step is similar to FP-Growth whereas generating association rules is similar to Apriori. As a result, Matrix Apriori combines the two algorithms by using their positive properties [5]. FP-Growth and Matrix Apriori algorithms are compared with using different characteristics of data in [6].

All of the algorithms above ignore the dynamicity of the databases. However, transactional databases are dynamic in general. When new transactions arrive, these algorithms should be re-run in order to find up-to-date frequent itemsets. The solution to this problem is incremental frequent itemset mining. The Fast UPdate (FUP) Algorithm [7] is the first algorithm proposed for incremental mining of frequent itemsets. It handles the databases with transaction insertion only, relies on Apriori and uses the pruning techniques used in the Direct Hashing and Pruning Algorithm [16]. The main working principle of this algorithm can be summarized in two steps; scanning only new transactions to generate 1-itemsets, then comparing these itemsets with the previous ones and finally discovering all frequent itemsets of the same size iteratively. FUP2, an extended version of FUP, copes with both insertion and deletion of transactions, was proposed by [17].

A new algorithm, FOLDARM, which is suitable for incremental association rule mining, was presented by [8]. FOLDARM constructs a new data structure called Support-Ordered Trie Itemset, SOTrieIT (a trie-like tree structure). This structure only stores the frequent 1-itemsets and 2-itemsets with their supports in a descending order of support counts (the most frequent itemsets are found on the leftmost branches of the SOTrieIT) and is used to discover frequent 1-itemsets and 2-itemsets without scanning the database. When new transactions arrive, all frequent 1-itemsets and 2-itemsets are extracted from each transaction. The extracted information is used to update the SOTrieIT without considering the support threshold. In order to mine frequent itemsets depth-first search is used starting from the leftmost first-level node. Unless a node that does not satisfy the support threshold, the traversal continues. Subsequently, the Apriori Algorithm is used to obtain other frequent itemsets.

The study by [9], proposed an incremental itemset mining algorithm based on Apriori which finds frequent itemsets and infrequent itemsets that are likely to be frequent after the arrival of new transactions. This algorithm uses the maximum support count of 1-itemsets in the database before the arrival of increments for finding potential frequent itemsets, called promising itemsets. In other words, in order to find a threshold value for finding promising itemsets, the maximum support count of 1-itemsets is used. It scans only new transactions, however it assumes that minimum support value does not change.

It is clear that when new transactions are added to the database, some frequent items may become frequent or infrequent. In the FP-Growth Algorithm, creating the FP-tree from the beginning according to the changes is time consuming. A fast upda7ted FP-tree (FUFP-tree) structure and an incremental FUPF-tree maintenance algorithm was proposed by [10]. However, when a sufficiently large number of transactions are inserted to the database, the whole FUFP-tree should be re-constructed [18].

Another method for constructing FP-tree incrementally was presented by [11]. The minimum support threshold is accepted as 1 and FP-tree is updated by scanning the new transactions twice. Five synthetic and one real datasets are used in the experiments with different number of items and transactions. In both cases, this approach performs better compared to building the tree from the beginning.

The comparison of incremental itemset mining algorithms is displayed in Table 1. All these algorithms can handle the maintenance problem in case of insertion and new items can be presented in the increments. FOLDARM, Incremental FP-tree and Incremental Matrix Apriori can handle minimum support change while FUP, FUP2, FUFP-tree and Promising Frequent Itemset cannot. Also FUP, FUP2 and Promising Frequent Itemset need candidate generation. There is a point that should be taken into consideration when comparing these algorithms; FOLDARM only addresses finding 1-frequent itemsets and 2-frequent itemsets.

Table 1. Comparison of Incremental Itemset Mining Algorithms

	Deletion	Support Change	New Item Occurrence	No Candidate Generation
FUP [7]			+	
FUP2 [17]	+		+	
FOLDARM [8]	+	+	+	+
Promising Frequent Itemset [9]			+	
FUFP-tree [10]	+		+	+
Incremental FP-tree [11]	+	+	+	+
Incremental Matrix Apriori		+	+	+

3 Incremental Itemset Mining Algorithm

The Incremental Matrix Apriori Algorithm is proposed to overcome the problem of mining frequent itemsets in dynamically updated databases. As seen in Table 1, besides finding frequent itemsets by scanning only new transactions, this algorithm handles new items in the updates and allows support threshold changes. This chapter is divided into two subsections. The first one explains the base algorithm Matrix Apriori and the second one presents the Incremental Matrix Apriori Algorithm.

3.1 Matrix Apriori Algorithm

In Matrix Apriori, the frequent items are stored in a matrix called MFI (Matrix of Frequent Items) and the supports of the itemsets are stored in a vector called STE. Initially the database is scanned in order to determine the frequent items and the frequent items list. Subsequently, in the second scan, MFI and STE are built as follows. Beginning from the second row of the matrix, the matrix is stored according to the transactions. If the transaction contains the item, the value "1", if not, the value "0" is stored in the MFI and the STE value is set to "1". If the transaction is already included in the MFI, then it is not stored in a new row, but its STE is incremented by 1.

After the construction of the matrix, it is modified in order to speed up the frequent itemset search. For each column, beginning from the first line, each cell value is set to the number of the line where the cell value is equal to "1" in the unmodified matrix. If there are not any values of "1" in the remaining lines of the unmodified matrix, the cell value is set to "1". After the matrix construction, frequent itemsets are found as follows. Beginning from the item that has the least support value; the item is compared with the items found on its left in order to find frequent itemsets. Following that, their support values are counted. The support value of an itemset is found by sequentially adding the related rows of STE from top to bottom.

3.2 Incremental Matrix Apriori Algorithm

In order to provide incremental mining of frequent itemsets, the matrix is constructed by the minimum support value of 1. That means all items are kept in the MFI without considering their frequencies. Doing so, flexibility for support change is enabled as well. Due to the structure of the matrix, items are kept in a descending support order in the MFI. So finding frequent itemsets with any support threshold is easy. Since all items are kept in the MFI, frequent itemsets can be calculated from the item that satisfies the minimum support.

The process before the arrival of increments can be seen in Figure 1. The database is scanned and the support counts of items are calculated. The list of items in the specified order is named as 1-itemset ordered lists of items. All items in the transactions are included to the 1-itemset ordered list of items without considering if the item's support count is more than the minimum support or not. This process is shown in Figure 1.a. Afterwards, the MFI is constructed and then modified as in Figure 1.b. The way of construction of the MFI is the same as Matrix Apriori.

TID	Items
001	A C D
002	B C E
003	B C E
004	A B E

1-Itemsets	Support Count
{B}	3
{C}	3
{E}	3
{A}	2
{D}	1

Minimum support = 50%

1- itemset ordered list of items (*IS*) = { B, C, E, A, D }

a.

MFI				
B	C	E	A	D
0	1	0	1	1
1	1	1	0	0
1	0	1	1	0

STE
1
2
1

MFI				
B	C	E	A	D
3	2	3	2	2
0	3	0	4	1
4	1	4	0	0
1	0	1	1	0

STE
1
2
1

b. After the modification of MFI

Fig. 1. Incremental Matrix Apriori – Before Increments

When new transactions arrive, they are scanned and 1-itemset ordered list of items are updated as in Figure 2.a and as indicated in lines 1-2 of pseudo code in Figure 3. The new items in increments are included to the MFI by adding new columns as in lines 3-5 in Figure 3. The MFI is updated as follows. First, the new transaction is checked whether it exists in the MFI or does not. If it exists, its STE is incremented by 1; if it does not exist, it is added to the MFI. Adding to the MFI is done by setting the cell value to the line number of transaction where value "1" is stored in the MFI. When the item does not exist in the remaining transactions of the incremental database, the cell value is set to "1". This entire process is shown in Figure 2.b and in lines 6-12 in Figure 3. Finally, according to the change of the 1-itemset ordered list of items, the order of items in the MFI is changed as in Figure 2.c and in line of 13 in Figure 3.

Since Matrix Aprori does not have an update feature, it runs from the beginning on the updated database whereas Incremental Apriori only runs on the updates when new transactions arrive. Needless to say, Incremental Matrix Apriori finds exactly the same frequent itemsets as the Matrix Apriori does. Besides avoiding database scan and avoiding the construction of different matrices for mining frequent items with different support thresholds, management of matrix is easy. Moreover, the base algorithm scans database only twice and does not generate candidate sets.

TID	Items
005	C D
006	C F

1-Itemsets	Support Count
{C}	5
{B}	3
{E}	3
{A}	2
{D}	2
{F}	1

Minimum support = 50%

1- itemset ordered list of items (IS_{new}) = { C, B, E, A, D, F }

a.

MFI					
B	C	E	A	D	F
3	2	3	2	2	6
0	3	0	4	5	0
4	5	4	0	0	0
1	0	1	1	0	0
0	6	0	0	1	0
0	1	0	0	0	1

STE
1
1
2
1
1

b.

MFI					
C	B	E	A	D	F
2	3	3	2	2	6
3	0	0	4	5	0
5	4	4	0	0	0
0	1	1	1	0	0
6	0	0	0	1	0
1	0	0	0	0	1

STE
1
1
2
1
1

c.

Fig. 2. Incremental Matrix Apriori – After Increments

```
INPUT: MFI, STE, Additions A, 1-itemset ordered list IS
OUTPUT: MFI, STE, 1-itemset ordered list IS_new
BEGIN
1  Scan A
2  Create IS_new from IS using A
3  FOR each new item in IS_new
4    Add a column to MFI
5  END
6  FOR each transaction in A
7    IF the transaction exists in MFI
8       Update the STE by incrementing by 1
9    ELSE IF the transaction does not exist in MFI
10      Add a new line to MFI
11      Set the STE of the new line to 1
12 END
13 Update and reorder MFI using IS_new
14 Return IS_new, MFI, STE
END
```

Fig. 3. Update Process in Incremental Matrix Apriori Algorithm

4 Performance Evaluation

In this section, the Incremental Matrix Apriori Algorithm is compared with the Matrix Apriori Algorithm when new transactions are added to the database. Both algorithms are implemented in Java and test runs are performed on a computer with 2.93 GHz dual core processor and 2 GB memory. Two synthetic datasets with different characteristics are used; they are generated by utilizing ARTool (1.1.2) dataset generator [6]. Difficulties in finding real datasets compelled the tests to be done with synthetic datasets.

In the following subsections, performance analysis of the algorithms for two case studies while varying the size of incremental dataset is given. The purpose is to observe how the percentage of the increment sizes affects the performance of the algorithms for the generated datasets. The algorithms are compared for 20 increasing addition sizes in the range of 5% and 100%. In these tests, the minimum support is 10% and the initial database has 15000 transactions for both case studies. During performance evaluations, it is ensured that the system state is similar in all test runs and they give close results when repeated.

4.1 Case1: Database of Long Patterns with Low Diversity of Items

The first dataset has the following characteristics i) long patterns with low diversity of items, ii) number of items is 10000, iii) average size of transactions is 20 and iv) average size of patterns is 10. The performances of Matrix Apriori and Incremental Matrix Apriori with different increment sizes are demonstrated in Figure 4. In every increment size, Incremental Matrix Apriori performs better than re-running Matrix Apriori.

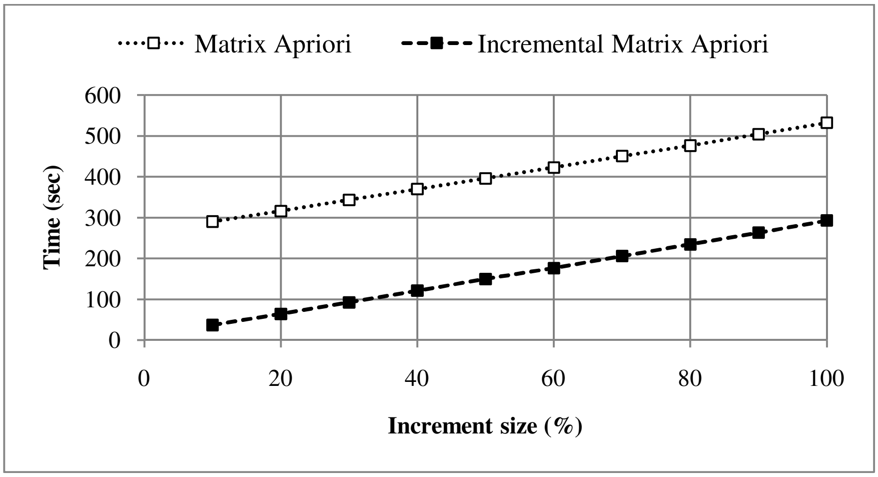


Fig. 4. Run-time with Different Increment Sizes for Case 1

In Figure 5, the speed-up with different increment sizes is shown. The speed-up increases from 45% to 92% as the increment size decreases. Although the speed-up decreases as the increment size increases, Incremental Matrix Apriori is 45% faster than re-running Matrix Apriori.

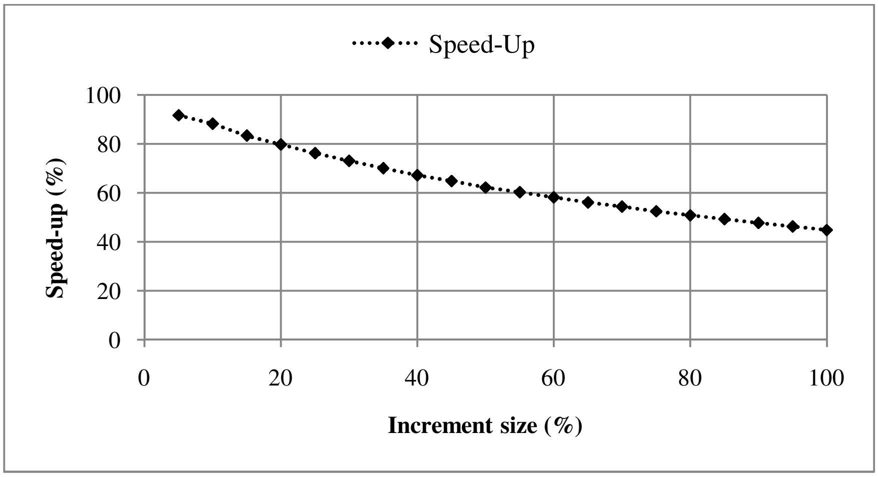


Fig. 5. Speed-up with Different Increment Sizes for Case 1

4.2 Case2: Database of Short Patterns with High Diversity of Items

The second dataset is composed of the following characteristics i) short patterns and high diversity of items, ii) number of items is 30000, iii) average size of transactions is 20 and v) average size of patterns is 5. The performances of Matrix Apriori and Incremental Matrix Apriori with different increment sizes are demonstrated in Figure 6.

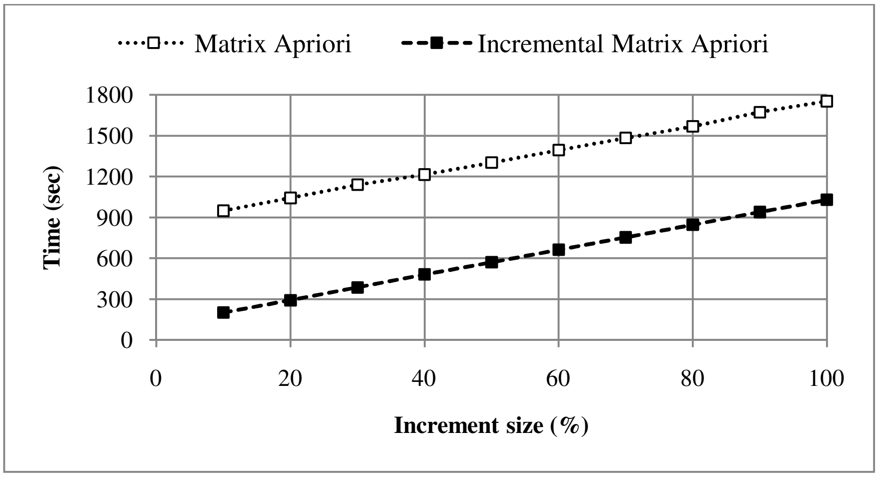


Fig. 6. Run-time with Different Increment Sizes for Case 2

As expected, in every increment size, Incremental Matrix Apriori outperforms the Matrix Apriori Algorithm as in Case 1.

The speed-up in different increment sizes is illustrated in Figure 7. The decrease in increment size increases the speed-up percentage from 41 to 81. Although there is a decrease in speed-up when the increment size becomes larger, Incremental Matrix Apriori is almost 41% faster than re-running Matrix Apriori.

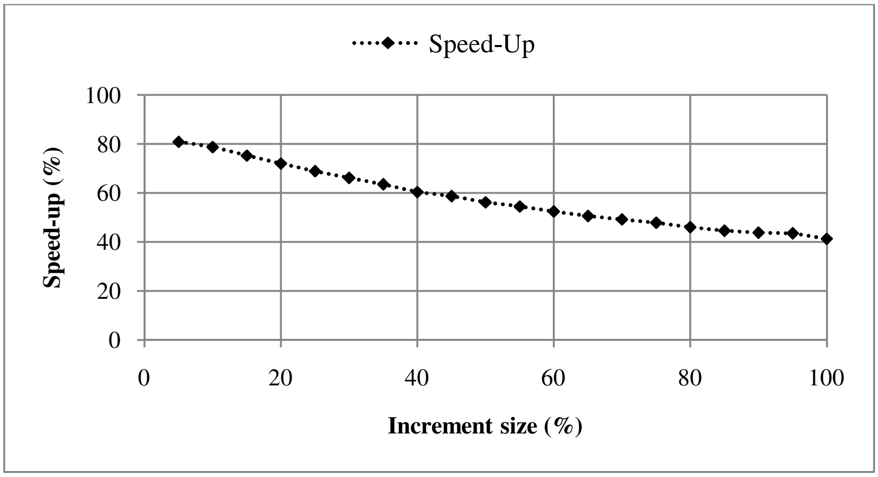


Fig. 7. Speed-up with Different Increment Sizes for Case 2

Both the test results in Case 1 and Case 2 reveal that Incremental Matrix Apriori performs better than running Matrix Apriori with each update in the range of 5% and 100%.

4.3 Discussion on Results

The test results demonstrate that, when the new transactions are added to database, Incremental Matrix Apriori finds frequent itemsets more efficiently than re-running Matrix Apriori. Two databases with different characteristics are used and the minimum support is given as 10% in both cases.

Performances of Incremental Matrix Apriori by using the first dataset and second dataset are shown in Figure 8. Finding frequent itemsets for Case 2 takes longer than Case 1. There are 10000 items in Case 1 while there are 30000 items in Case 2. This result has been expected because Incremental Matrix Apriori keeps all items in the MFI. The run-time increases as does the number of items. Moreover, the relationship between increment size and time are linear both two cases.

Figure 9 shows the speed-up comparison of Incremental Matrix Apriori for Case 1 and Case 2. Although the behavior in two cases is similar, the speed-up for Case 1 is higher than that of Case 2. This may be due to the number of items. Since Incremental Matrix Apriori keeps all the items in the matrix even it exists in only in one transaction, when the number of items increases, the total time increases as well. On the other hand, Matrix Apriori keeps the items that satisfy the minimum support. So the speed-up is lower in Case 2.

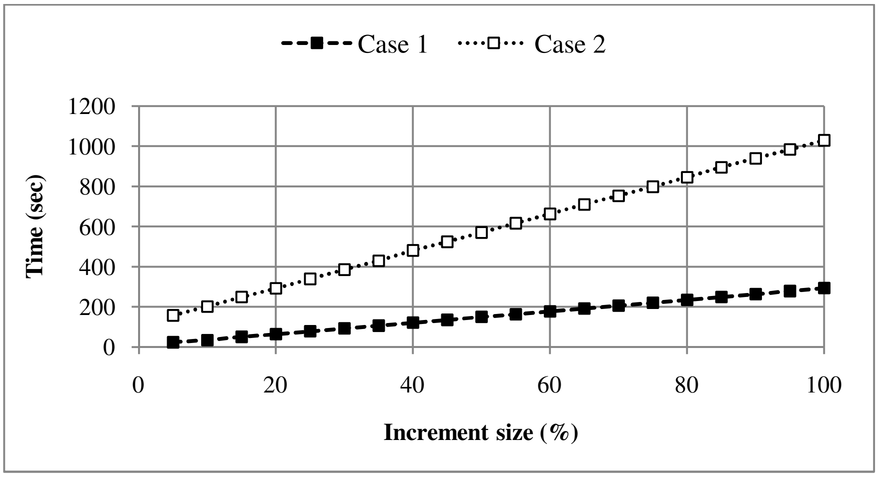


Fig. 8. Run-time for Case 1 and Case 2

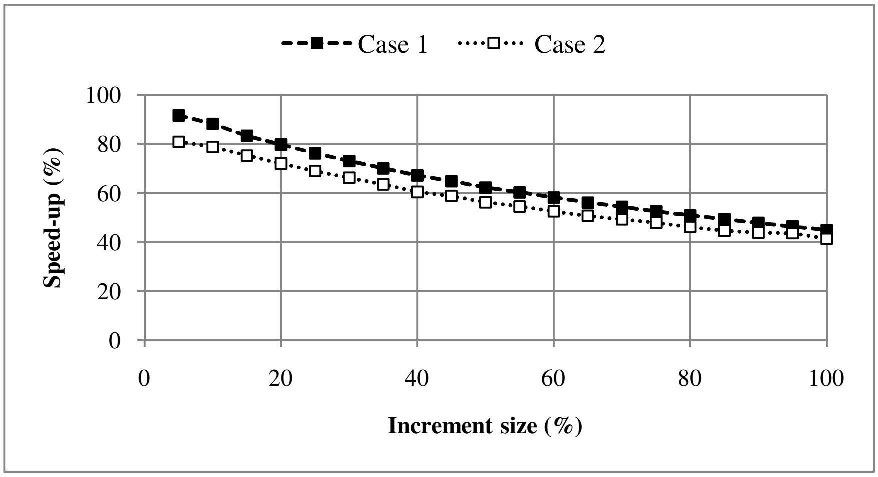


Fig. 9. Speed-up for Case 1 and Case 2

5 Conclusion and Future Work

In this paper, we propose a new algorithm for the problem of incremental itemset mining which is based on the Matrix Apriori Algorithm. A matrix structure is used in order to handle new transactions. Accepting the minimum support as 1, all items are kept in the matrix in a descending order of support counts. Therefore, without scanning the entire database, the new transactions and new items can be added to the matrix easily. Moreover, this approach allows the user to change minimum support. Since all frequent and infrequent items exist in the matrix, the calculation of frequent itemsets can start from the updated minimum support count.

Experimental results show that Incremental Matrix Apriori performs better than re-running Matrix Apriori for handling new transactions. The algorithms are compared for 20 increment sizes in the range of 5% and 100% by using two datasets with different characteristics. When the number of items or the increment size decreases, the speed-up by Incremental Matrix Apriori increases. However, both tests show that Incremental Matrix Apriori provides speed-up between 41% and 92% while increment size is varied between 5% and 100%.

Our work is ongoing and we aim to extend our algorithm to handle the deletion of transactions as well. After this step, our plan is to compare our incremental approach with other incremental itemset mining approaches in order to understand its strengths and weaknesses. In addition, we are planning to test our algorithm on real datasets.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. ACM SIGMOD Record 22(2), 207–216 (1993)
2. Han, J., Kamber, M.: Data mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier, San Francisco (2006)
3. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (1994)
4. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidate Generation. ACM SIGMOD Rec. 29(2), 1–12 (2000)
5. Pavon, J., Viana, S., Gomez, S.: Matrix Apriori: Speeding Up the Search for Frequent Patterns. In: 24th IASTED International Conference on Database and Applications, pp. 75–82. ACTA Press, Anaheim (2006)
6. Yıldız, B., Ergenç, B.: Comparison of Two Association Rule Mining Algorithms without Candidate Generation. In: 10th IASTED International Conference on Artificial Intelligence and Applications, Innsbruck (2010)
7. Cheung, D., Han, J., Ng, V., Wong, C.: Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In: 12th International Conference on Data Engineering, pp. 106–114. IEEE Computer Society, Washington (1996)
8. Woon, Y.K., Ng, W.K., Das, A.: Fast online dynamic association rule mining. In: 2nd International Conference, vol. 1, pp. 278–287 (2001)
9. Amornchewin, R., Kreesuradej, W.: Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm. In: 6th International Conference on Information, Communications & Signal Processing, pp. 1–5 (2007)
10. Hong, T.P., Lin, C.W., Wu, Y.L.: Incrementally Fast Updated Frequent Pattern Trees. Expert Syst. Appl. 34(4), 2424–2435 (2008)
11. Muhaimenul, A.R., Barker, K.: Alternative Method for Incrementally Constructing the FP-tree. In: Chountas, P., Petrounias, I., Kacprzyk, J. (eds.) Intelligent Techniques and Tools for Novel System Architectures. SCI, vol. 109, pp. 361–377. Springer, Heidelberg (2008)
12. Jin, R., Agrawal, G.: An Algorithm for In-Core Frequent Itemset Mining on Streaming Data. In: 5th IEEE International Conference on Data Mining, pp. 210–217. IEEE Computer Society, Washington (2005)
13. Calders, T., Dexters, N., Goethals, B.: Mining Frequent Itemsets in a Stream. In: 7th IEEE Conference on Data Mining, pp. 83–92. IEEE Computer Society, Washington (2007)

14. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 Algorithms in Data Mining. Knowl. Inf. Syst. 14(1), 1–37 (2007)
15. Kotsiantis, S., Kanellopoulos, D.: Association Rules Mining: A Recent Overview Basic Concepts & Basic Association Rules Algorithms. Science 32(1), 71–82 (2006)
16. Park, J.S., Chen, M.S., Yu, P.S.: An Effective Hash-Based Algorithm for Mining Association Rules. SIGMOD Rec. 24(2), 175–186 (1995)
17. Cheung, D.W.L., Lee, S.D., Kao, B.: A General Incremental Technique for Maintaining Discovered Association Rules. In: 5th International Conference on Database Systems for Advanced Applications, pp. 185–194. World Scientific Press (1997)
18. Lin, C.-W., Hong, T.-P., Lu, W.-H., Chien, B.-C.: Incremental Mining with Prelarge Trees. In: Nguyen, N.T., Borzemski, L., Grzech, A., Ali, M. (eds.) IEA/AIE 2008. LNCS (LNAI), vol. 5027, pp. 169–178. Springer, Heidelberg (2008)

A Fast Algorithm for Frequent Itemset Mining Using Patricia* Structures

Jun-Feng Qu[1] and Mengchi Liu[2]

[1] State Key Lab of Software Engineering, School of Computer, Wuhan University, Wuhan 430072, China
cocoqjf@gmail.com

[2] School of Computer Science, Carleton University, Ottawa K1S 5B6, Canada
mengchi@scs.carleton.ca

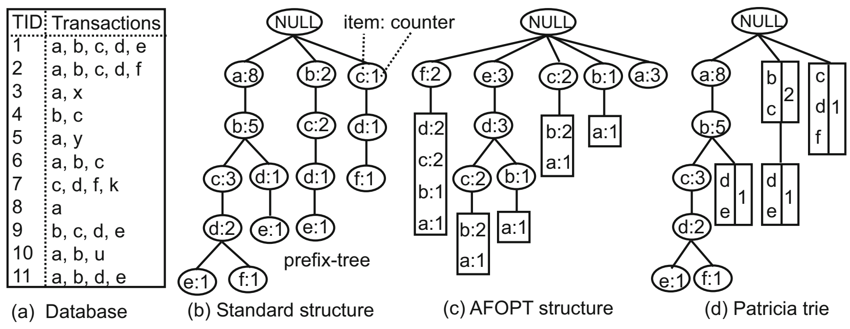
Abstract. Efficient mining of frequent itemsets from a database plays an essential role in many data mining tasks such as association rule mining. Many algorithms use a prefix-tree to represent a database and mine frequent itemsets by constructing recursively conditional prefix-trees from the prefix-tree. A (conditional) prefix-tree can be stored in various structures. The construction and traversal costs of prefix-trees, or rather their storage structures, take a large proportion in the whole cost for such algorithms. The PatriciaMine algorithm employs a Patricia trie to store a prefix-tree and shows good performance. In this study, we introduce an efficient Patricia* structure for storing a prefix-tree. A Patricia* structure is more compact and contiguous than a corresponding Patricia trie, and thus the construction and traversal costs of the former are less than those of the latter. Previous prefix-tree-based algorithms adopt a similar mining procedure, in which most nodes in a prefix-tree are repeatedly accessed when the prefix-tree is processed. The paper presents a novel mining procedure in which node accesses for a prefix-tree are greatly reduced. We propose the PatriciaMine* algorithm that is the combination of the Patricia* structure with the proposed procedure. Experimental data show that PatriciaMine* outperforms not only PatriciaMine but also several fast algorithms, such as FPgrowth* and dEclat, for various databases.

Keywords: Data mining, frequent itemset, prefix-tree.

1 Introduction

Since the introduce of frequent itemsets by Agrawal et al. [1], they have been applied to association rule mining, inductive databases, classification, and so on. Many problems [2, 7] derived from frequent itemsets have also been proposed. Therefore, mining efficiently frequent itemsets from a database has been a core problem in data mining area [3].

Let $\mathcal{I}$ be a set of items, each subset of $\mathcal{I}$ is an itemset. An itemset containing k items is called a k-itemset. For a transaction database, each transaction has a unique identifier (TID) and is also a subset of $\mathcal{I}$. A transaction *satisfies* an



TID	Transactions
1	a, b, c, d, e
2	a, b, c, d, f
3	a, x
4	b, c
5	a, y
6	a, b, c
7	c, d, f, k
8	a
9	b, c, d, e
10	a, b, u
11	a, b, d, e

Fig. 1. Database and prefix-tree storage structure (*minimum support=2*)

itemset if the transaction contains all the items of the itemset. The number (or percentage) of transactions satisfying an itemset is the *support* of the itemset. An itemset is *frequent* if its support exceeds a user-specified minimum support threshold. Given a database and a minimum support, the frequent itemset mining problem is to identify all frequent itemsets.

Prefix-tree-based algorithms [4, 5, 8, 10, 12] are thought of as ones of the fastest mining algorithms [3]. These algorithms employ prefix-trees to represent databases. After identifying all the frequent items in a database and initializing an empty tree, for each transaction, the algorithms generate a branch composed of the frequent items in the transaction and subsequently insert the branch into the prefix-tree. The frequent items of a branch are usually sorted in frequency-descending order, which can increase the shared paths in a prefix-tree. Fig. 1(a) shows a database, and the corresponding prefix-tree stored in a standard structure is depicted in Fig. 1(b) when the minimum support is 2. Each node in the structure contains an *item* and a *counter*. The counter value of a node registers the number of transactions containing the itemset composed of the items in the path from the node to the root. For item i, the paths from all the nodes containing i to the root constitute the conditional database of i. (The conditional database of i can also be defined as the sub-trees rooted at all the nodes containing i, which depends on algorithms.) The algorithms traverse the paths to identify the frequent items in the conditional database, and append each frequent item to i to generate a frequent 2-itemset. After that, the conditional prefix-tree of i is constructed from the conditional database and is processed recursively.

A prefix-tree can be stored in various structures. Fig. 1(b) shows the standard structure, and the famous FP-Growth algorithm [5] employs this structure. The AFOPT algorithm [8] uses an AFOPT structure to store a prefix-tree. The algorithm sorts items in frequency-ascending order, and any single branch ending with a leaf in a prefix-tree is stored in a node in an AFOPT structure. Fig. 1(c) depicts the AFOPT structure constructed from the database in Fig. 1(a). The PatriciaMine algorithm [10] adopts a Patricia trie [6] to store a prefix-tree. For a standard structure, if each node that has only one child node with the same counter value as that in the node is merged with its child node, the structure becomes a Patricia trie. The prefix-tree in Fig. 1(b) can also be stored in the Patricia trie in Fig. 1(d). For a prefix-tree-based algorithm, the prefix-tree storage structure significantly influences its performance, because the algorithm usually constructs and processes a very large number of conditional prefix-trees [9]. Another important factor relevant to the performance of an algorithm is its mining procedure. For example, using the FP-array technique, the efficient FPgrowth* algorithm [4] merges the counting operation with the construction operation and thereby distinctly improves the performance of the FP-Growth algorithm.

The contributions of this work are as follows. Firstly, we improve the Patricia trie used in the PatriciaMine algorithm and introduce a Patricia* structure for storing a prefix-tree. For a prefix-tree, the construction and traversal costs of a Patricia* structure are less than those of a Patricia trie. Secondly, an efficient mining procedure is introduced, in which node accesses are greatly reduced when a prefix-tree is processed, compared with the previous universal mining procedure. We propose the PatriciaMine* algorithm that is the combination of the improved procedure with the Patricia* structure. Thirdly, extensive experimental results show that PatriciaMine* outperforms not only PatriciaMine but also several fast algorithms for various databases. The paper is organized according to the three points aforementioned in Section 2, 3, and 4. Our work is summarized in Section 5.

2 Patricia* Structure

In a prefix-tree storage structure, each node contains two pieces of information: one related to the structure itself, namely various pointers, and the other related to frequent itemsets, namely item(s) and counter(s). When mining frequent itemsets, an algorithm only makes use of the second piece of information for which the algorithm has to access the first piece of information. No matter what structure a prefix-tree is stored in, the amount of the information about the structure itself can be measured in terms of the number of nodes in the structure. The smaller the number of nodes in a storage structure is, the less the construction and traversal costs of the structure are. In the following, we introduce an improved Patricia trie, namely a Patricia* structure, for storing a prefix-tree, and illustrate how to construct such a structure from a database.

2.1 One-Child Node

In a standard structure storing a prefix-tree, a node that has only one child node is called a "*one-child*" node. Given a prefix-tree, although the nodes in the Patricia trie storing it are usually fewer than those in the standard structure storing it, it does not make sense to construct the Patricia trie if there are a very small number of one-child nodes in the standard structure. Therefore, we first researched various prefix-trees constructed from the databases in the FIMI repository (http://fimi.cs.helsinki.fi/). These databases used extensively in previous studies hold various characteristics, and their statistical information

Database	Size(bytes)	#Trans	#Items	AvgLen	MaxLen
accidents	35509823	340183	468	33	51
chess	342294	3196	75	37	37
pumsb	16689761	49046	2113	74	74
retail	4167490	88162	16470	10	76
T10I4D100K	4022055	100000	870	10	29
T40I10D100K	15478113	100000	942	40	77

Fig. 2. Statistical information about experimental databases

DB(Minsup)	#Node	#One-child	Percentage
accidents(3%)	3834840	3277063	85%
chess(20%)	33930	28167	83%
pumsb(50%)	108227	88276	82%
retail(7)	659668	565895	86%
T10(0.002%)	714737	601798	84%
T40(0.08%)	3562638	3427217	96%

Fig. 3. Numbers of one-child nodes in standard structures storing prefix-trees

is showed in Fig. 2. For each database, the size, the number of transactions, the number of distinct items, the average transaction length, and the maximal transaction length are listed.

Fig. 3 gives the numbers of one-child nodes in the standard structures constructed from the above databases. An important fact is that "there are usually a very large number of one-child nodes in a standard structure storing a prefix-tree". For example, when the minimum support is 20%, the standard structure storing the prefix-tree from database *chess* contains 33930 nodes in which there are 28167 one-child nodes, and the percentage of one-child nodes amounts to 83%. The percentages of one-child nodes for all the databases exceed 80%, and the percentage for *T40I10D100K* is even larger than 95%.

Since a very large number of one-child nodes exist in a standard structure storing a prefix-tree, merging a one-child node with its child node is an efficient method of reducing the number of nodes in a prefix-tree storage structure. A Patricia* structure is a modification of a standard structure, in which each one-child node in the standard structure is merged with its child node. Namely, each maximal chain of one-child nodes in a standard structure is merged into a node in a corresponding Patricia* structure.

Fig. 4 shows the Patricia* structure storing the same prefix-tree in the standard structure in Fig. 1(b). Please note the differences between the Patricia trie in Fig. 1(d) and the Patricia* structure here. In a Patricia trie, each maximal chain of one-child nodes is merged into a node on condition that these nodes hold the

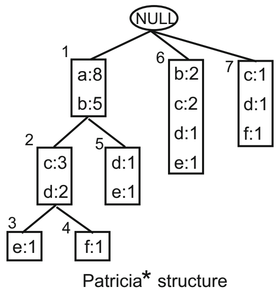


Fig. 4. Core structure

DB(Minsup)	# Generation	# Truncation	# Extension
accidents(3%)	120570	218603	170
chess(20%)	592	2585	0
pumsb(50%)	2064	8943	575
retail(7)	67210	13281	507
T10(0.002%)	63410	24764	311
T40(0.08%)	64438	35491	0

Fig. 5. Numbers of operations for constructing Patricia* structures

same counter value. In a Patricia* structure, each maximal chain of one-child nodes is unconditionally merged into a node. Therefore, the nodes in a Patricia* structure are fewer than those in a corresponding Patricia trie. To store the prefix-trees representing the database in Fig. 1(a), the standard structure, the AFOPT structure, the Patricia trie, and the Patricia* structure respectively contain 16, 14, 11, and 8 nodes. The Patricia* structure contains the smallest number of nodes, which means the construction and traversal costs of the Patricia* structure are less than those of the other structures.

2.2 Constructing a Patricia* Structure

In the process of constructing a Patricia* structure, besides the merging operation, the other operations are generating a node, extending a node, and truncating a node. One node is generated in each generation operation, and two nodes are generated in each truncation operation. Extension operations do not increase the number of nodes. By constructing Patricia* structures from experimental databases, we observed that generation operations and truncation operations are performed frequently while extension operations rarely happen. Fig. 5 presents the numbers of these operations.

Fig. 6 demonstrates how the Patricia* structure in Fig. 4 is constructed from the database in Fig. 1(a). For example, when branch $< abcde >$ is inserted into the structure, the node numbered 1 containing these items is generated. The node is truncated when branch $< abcdf >$ is inserted, and the node numbered 2 containing the truncated part and the node numbered 3 containing the remaining part of the branch are generated after the shared items are merged. When branch $< bcde >$ is inserted into the structure, the node numbered 4 is extended for

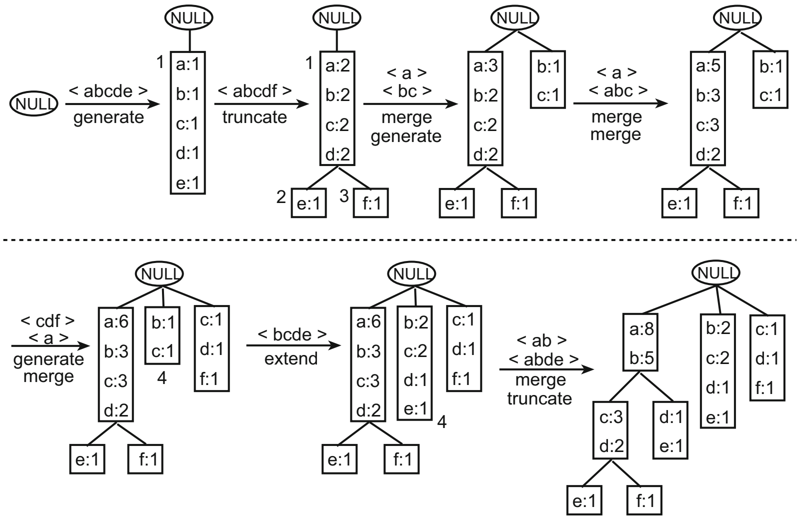


Fig. 6. Patricia* structure construction

Procedure 1. PreviousMiner(F, T)

```
Input: $F$ is a frequent itemset, initially empty;
       $T$ is the conditional prefix-tree of $F$.
Output: all the frequent itemsets with $F$ as prefix
1 foreach $item_i$ in $T$ do
2     $F_i = F \cup item_i$;
3     output $F_i$;
4     identify the frequent items in the conditional database of $item_i$;
5     construct $item_i$'s conditional prefix-tree $T_i$;
6     PreviousMiner($F_i$, $T_i$);
7 end
```

items d and e. Note that although extension operations do not increase the number of nodes in a Patricia* structure, an extension operation leads to a memory allocation and a memory release. Fortunately, the numbers of extension operations are small for constructing real Patricia* structures. For example, in Fig. 5, there are 120570 generation operations, 218603 truncation operations, and only 170 extension operations when the Patricia* structure for database *accidents* is constructed (The minimum support is 3%.).

3 Mining Frequent Itemsets Using Patricia* Structures

In this section, we improve the previous universal mining procedure and combine the Patricia* structure with the improved procedure.

3.1 Previous Mining Procedure

No matter what structure previous algorithms store a prefix-tree in, they hold a similar mining procedure, which is showed in Procedure 1. For (conditional) prefix-tree T, all the items in T are frequent, and they are processed one by one as follows. Firstly, a new frequent itemset F_i, composed of an item in T denoted by $item_i$ and prefix itemset F, is created and outputted (lines 2-3). Secondly, the paths from all the nodes containing $item_i$ to the root constitute the conditional database of $item_i$. To identify the frequent items in the conditional database of $item_i$, these paths are traversed and the items in the paths are counted (line 4). Thirdly, these paths are traversed again for constructing $item_i$'s conditional prefix-tree T_i, in which the only frequent items in the conditional database are considered (line 5). At last, T_i is processed recursively.

Consider a path in a prefix-tree stored in a standard structure, suppose it contains items i_1, i_2, …, $i_{(n-1)}$, and i_n from the root to a leaf node. When i_n is processed, the nodes containing i_1, i_2, …, and $i_{(n-1)}$ are accessed twice (one for counting, the other for constructing); when $i_{(n-1)}$ is processed, the nodes containing i_1, i_2, …, and $i_{(n-2)}$ are accessed twice; … Therefore, there are $n(n-1)$ node accesses for the path in the procedure. Even for FPgrowth*

Procedure 2. ImprovedMiner(F, T)

```
Input: $F$ is a frequent itemset, initially empty;
       $T$ is the conditional prefix-tree of $F$.
Output: all the frequent itemsets with $F$ as prefix
1 identify the frequent items in the conditional databases of all the items in $T$;
2 construct the conditional prefix-trees of all the items in $T$;
3 foreach $item_i$ in $T$ do
4     $F_i = F \cup item_i$;
5     output $F_i$;
6     ImprovedMiner($F_i$, $T_i$);
7 end
```

that merges the counting operations (line 4) into the construction operations (line 5), there are $n(n-1)/2$ node accesses for the path.

3.2 Improved Mining Procedure

The previous mining procedure can be improved as showed in Procedure 2. The main modification is that both the counting operations and the construction operations of Procedure 1 are moved out of the loop. By one depth-first traversal of T, the items in all the conditional databases are counted, which can be thought of as the process of counting for all the 2-itemsets in T. Using a work stack that stores the items in the path from the currently visited node to the root, when a node containing $item_i$ is visited, each 2-itemset composed of $item_i$ and an item in the stack is counted according to the counter value in the node. After T is traversed, the frequent items in any conditional database can be identified. Subsequently, all the conditional prefix-trees are simultaneously constructed by the second depth-first traversal of T. When a node containing $item_i$ is visited, the items in the stack that are also frequent in the conditional database of $item_i$ are picked out. These items are sorted and inserted into the conditional prefix-tree of $item_i$. At last, each conditional prefix-tree is recursively processed.

The advantage of the improved procedure is that all the nodes in a prefix-tree are accessed only twice when the prefix-tree is processed. Therefore, the path aforementioned is traversed twice, and namely there are only $2n$ node accesses for the path. In fact, node accesses in the procedure are far fewer than those in the previous procedure in consideration of that many paths are overlapped in a prefix-tree.

3.3 PatriciaMine* Algorithm

We refer to the combination of the improved mining procedure with the Patricia* structure as the PatriciaMine* algorithm. After an initial Patricia* structure is constructed from a database, PatriciaMine* can mine all the frequent itemsets from the structure. As an example, each node in the Patricia* structure in Fig. 4 is numbered, and PatriciaMine* processes this structure as follows.

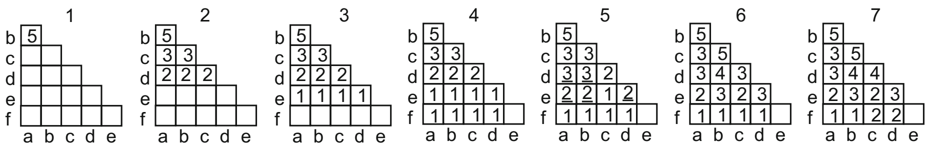


Fig. 7. Counting operations

Firstly, all the 2-itemsets in the Patricia* structure are counted during one depth-first traversal of the structure. When a node is visited, each combination of an item in the node and an item before the item (either in the node or in an ancestor node of the node) is counted according to the counter value corresponding to the item in the node. For example, when the node numbered 5 in Fig. 4 is visited, the items in the node are d and e, and the items in its ancestor node numbered 1 are a and b. Then 2-itemsets da, db, ea, eb, and ed are counted. Note that items a and b have been stored in the work stack when the above counting operations are performed, and thus the node numbered 1 is not accessed. The counting operations on the Patricia* structure are demonstrated in Fig. 7. After the depth-first traversal of the structure, the frequent items in each conditional database can be identified, and they are respectively: $\{a\}$ in D_b (representing the conditional database of item b), $\{b, a\}$ in D_c, $\{b, c, a\}$ in D_d, $\{b, d, a, c\}$ in D_e, and $\{c, d\}$ in D_f. (The minimum support is 2, and the frequent items in a conditional database are sorted in frequency-descending order.)

Secondly, PatriciaMine* traverses the structure again to construct all the conditional Patricia* structures. We denote the conditional Patricia* structure of item i by P_i. The updated P_is are depicted in Fig. 8 when PatriciaMine* processes each node. For example, when the node numbered 5 is processed, P_d and P_e are updated. Branch $<ba$: 1$>$ is inserted into P_d and branch $<bda$: 1$>$ is inserted into P_e.

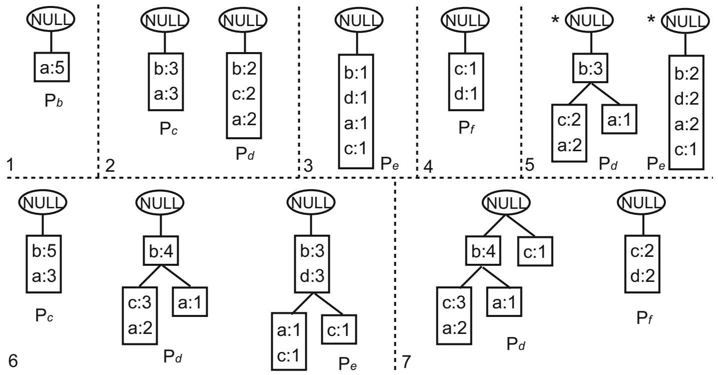


Fig. 8. Construction operations

At last, each item in the structure in Fig. 4 is outputted and the item with its conditional Patricia* structure is recursively processed.

4 Experiments and Performance Study

All the experiments were performed on a 2.83GHz Intel Core2 PC with 4GB of memory, running on a Debian (Linux 2.6.26) operation system. We implemented the PatriciaMine* algorithm. PatriciaMine* was compared with several famous prefix-tree-based algorithms including FP-Growth [5], FPgrowth* [4], AFOPT [8], PatriciaMine [10], in which FPgrowth* is the fastest algorithm in IEEE ICDM Workshop on frequent itemset mining implementations (FIMI'03). We also tested the significant dEclat algorithm [11, 13], in which each itemset holds a TID-list that indicates the transactions satisfying the itemset and the length of the TID-list is the support of the itemset. The dEclat algorithm mines frequent itemsets by concatenating two frequent itemsets and intersecting their TID-lists. To avoid implementation bias, the implementation of FP-Growth was downloaded from http://adrem.ua.ac.be/~goethals/software/, and the implementations of the other algorithms were downloaded from http://fimi.cs.helsinki.fi/. All of the codes were written in C++ and compiled using gcc (version 4.3.2) with standard optimization flags. The databases showed in Fig. 2 were used in our experiments.

4.1 Numbers of Nodes Generated during Mining Processes

The purpose of the experiment is to evaluate the total number of nodes in the prefix-tree storage structures constructed by an algorithm for a mining task. Fig. 9 shows the experimental data. Note that FP-Growth, FPgrowth*, PatriciaMine, and PatriciaMine* construct the same prefix-trees for a mining task, no matter what structure the prefix-trees are stored in and no matter when/how the prefix-trees are constructed. AFOPT sorts items in frequency-ascending order when constructing prefix-trees that are different from the above prefix-trees.

The total numbers of nodes generated by FP-Growth/FPgrowth*, AFOPT, PatriciaMine, and PatriciaMine* are labeled with *Standard*, *AFOPT*, *Patricia*, and *Patricia** in Fig. 9 respectively. FP-Growth and FPgrowth* employ standard structures to store prefix-trees, and thus the nodes generated by FP-Growth are the same as those generated by FPgrowth* for a mining task. Because each maximal chain of one-child nodes is unconditionally merged in a Patricia* structure, PatriciaMine* generates the smallest number of nodes for any mining task compared with the other algorithms. Another observation is that the nodes generated by AFOPT are more than those generated by FP-Growth/FPgrowth* in some cases, e.g., for *accidents*. Although any single branch ending with a leaf is merged in an AFOPT structure, the frequency-ascending item order leads to the decrease in the proportion of shared paths in the AFOPT structure.

The construction and traversal costs of a structure depend on the number of nodes in the structure to a great extent. For example, FP-Growth performs

accidents	3%	6%	9%	12%	15%	18%
Standard	599935311	123572739	44713716	20849286	11407958	6970059
AFOPT	761073376	153243380	54346857	25014577	13317273	8099201
Patricia	455339982	92347737	32379738	14584470	7574911	4439166
Patricia* (RR)	354735548 (22%)	72764825 (21%)	25745317 (20%)	11711303 (20%)	6131116 (19%)	3647871 (18%)
chess	20%	25%	30%	35%	40%	45%
Standard	247946947	88054121	34844557	14744303	6558958	3020348
AFOPT	212089968	79702157	33063414	14533059	6671473	3156128
Patricia	112674303	43106304	18186645	8121762	3784212	1815962
Patricia* (RR)	99952007 (11%)	37298169 (13%)	15418326 (15%)	6773611 (17%)	3114066 (18%)	1477437 (19%)
pumsb	50%	55%	60%	65%	70%	75%
Standard	159827210	49249021	19377015	8341505	3155347	970099
AFOPT	153610729	48267458	18595211	8350498	3428882	1133563
Patricia	82261358	26450324	10225970	4624710	1942992	662855
Patricia* (RR)	69518537 (15%)	22121107 (16%)	8605444 (16%)	3825549 (17%)	1555279 (20%)	515720 (22%)
retail	3	4	5	6	7	8
Standard	18405707	5378063	3050542	2243873	1848767	1595451
AFOPT	12912793	3964410	2278174	1658928	1339906	1129360
Patricia	6869517	2279767	1376145	1028615	843533	718617
Patricia* (RR)	6517756 (5%)	2047910 (10%)	1207413 (12%)	897059 (13%)	735540 (13%)	626753 (13%)
T10I4D100K	0.002%	0.003%	0.004%	0.005%	0.006%	0.007%
Standard	21372692	10162125	7118152	5660772	4814589	4263162
AFOPT	15957239	7603852	5101456	3864764	3146854	2689454
Patricia	8911605	4530823	3165817	2439237	1989200	1688830
Patricia* (RR)	8209643 (8%)	4071747 (10%)	2822568 (11%)	2173082 (11%)	1780158 (11%)	1521368 (10%)
T40I10D100K	0.08%	0.09%	0.1%	0.11%	0.12%	0.13%
Standard	283288570	245853060	218351933	195894118	152020719	132570561
AFOPT	160656129	140251343	125562805	112791043	83540565	71070469
Patricia	101291580	88842249	79895766	72086939	55366682	47495297
Patricia* (RR)	90224019 (11%)	79171089 (11%)	71130406 (11%)	64168538 (11%)	48787132 (12%)	41784013 (12%)

RR (Reducing Rate) = (|Patricia| - |Patricia*|) / |Patricia| $\times$ 100%.

Fig. 9. Numbers of nodes generated during mining processes

152020719 memory allocations to construct standard structures for mining frequent itemsets from database *T40I10D100K* when the minimum support is 0.12%; FP-Growth needs 152020719 pointer dereferences for traversing these structures one time. To perform the same mining task, for PatriciaMine*, there are only 48787132 memory allocations for constructing Patricia* structures and only 48787132 pointer dereferences for traversing the structures one time. Therefore, the construction and traversal costs of PatriciaMine* are less than those of the other algorithms according to the experimental data in Fig. 9.

4.2 Performance Comparison

Fig. 10 depicts the running time of these algorithms. Running time was recorded by *time* command and contains input time, computational time, and output time. Output was directed to */dev/null*. It can be observed that PatriciaMine* performs the best for almost all the databases and minimum supports.

The performance curves for PatriciaMine, FPgrowth*, and AFOPT are overlapped in many cases, for example, in Fig. 10(b), (c), (e), and (f). AFOPT is slower than PatriciaMine and FPgrowth* for database *accidents* as showed in

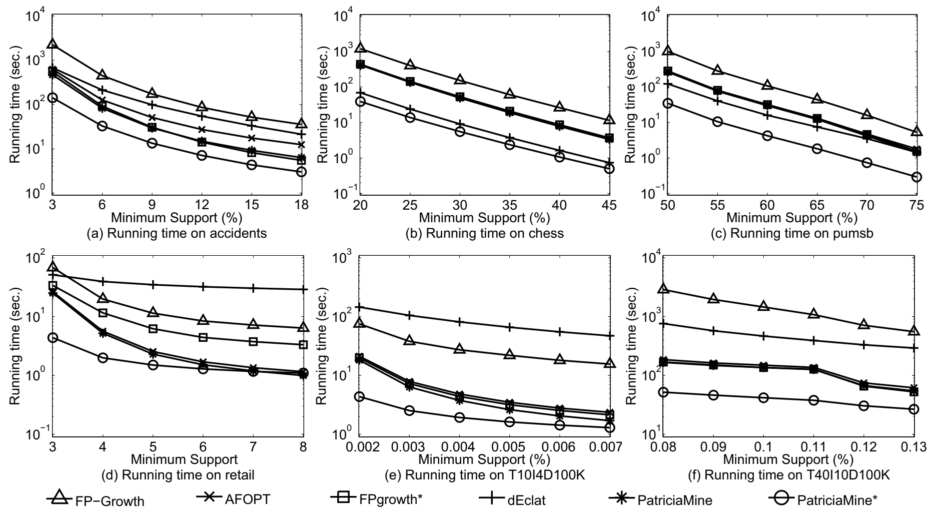


Fig. 10. Performance Comparison

Fig. 10(a), and FPgrowth* is slower than PatriciaMine and AFOPT for database *retail* in Fig. 10(d). PatriciaMine is more stable than FPgrowth* and AFOPT. We can observe from Fig. 9 that the nodes generated by PatriciaMine are always fewer than those generated by AFOPT or FPgrowth*, and thus the construction and traversal costs of PatriciaMine are less than those of AFOPT or FPgrowth*.

PatriciaMine* distinctly outperforms PatriciaMine. The performance improvement of PatriciaMine* results from two factors: (1) PatriciaMine cuts a node off as long as the counter values corresponding to any two items in the node are different from each other, but PatriciaMine* does not. Therefore, PatriciaMine* avoids a very large number of operations for cutting nodes off. For example, according to Fig. 9, PatriciaMine generates 455339982 nodes while PatriciaMine* generates 354735548 nodes for mining frequent itemsets from database *accidents* when the minimum support is 3%, which means that PatriciaMine cuts off 100604434 (=455339982-354735548) nodes while PatriciaMine* doesn't perform these operations. (2) The nodes generated by PatriciaMine* are always fewer than those generated by PatriciaMine, and thus the construction and traversal costs of PatriciaMine* are less than those of PatriciaMine.

Except for FP-Growth, prefix-tree-based mining algorithms outperform dEclat in many cases. However, dEclat is faster than FPgrowth*, AFOPT, and PatriciaMine for databases *chess* and *pumsb* as depicted in Fig. 10(b) and (c). It can be learned from Fig. 2 that the numbers of transactions in *chess* and *pumsb* (3196 and 49046 respectively) are small. Therefore, the TID-lists constructed from the databases are very short and TID-list intersections can be performed fast in dEclat. Even so, PatriciaMine* distinctly outperforms dEclat. Another performance advantage of PatriciaMine* derives from its efficient mining procedure. For all the nodes in a prefix-tree storage structure, previous prefix-tree-based algorithms access most of them many times for processing the prefix-tree while PatriciaMine* accesses any of them only twice. The traversal cost of PatriciaMine* is less than that of previous algorithms.

5 Conclusion and Future Work

The paper presents a novel algorithm, PatriciaMine*, for frequent itemset mining. The advantages of PatriciaMine* over other prefix-tree-based algorithms are: (1) PatriciaMine* stores a prefix-tree in a Patricia* structure that is more compact and contiguous than other storage structures. (2) PatriciaMine* adopts the improved mining procedure and can efficiently process any prefix-tree generated during its mining process. Extensive experimental data show that PatriciaMine* achieves significant performance improvement over previous works. For all the conditional Patricia* structures from a Patricia* structure, they are independent from one another. Therefore parallel mining of these structures are feasible, which is a further research direction.

References

[1] Agrawal, R., Imieliński, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: Proc. ACM SIGMOD, pp. 207–216 (1993)
[2] Calders, T., Garboni, C., Goethals, B.: Approximation of Frequentness Probability of Itemsets in Uncertain Data. In: Proc. IEEE ICDM, pp. 749–754 (2010)
[3] Ceglar, A., Roddick, J.F.: Association Mining. ACM Comput. Surv. 38(2), 1–42 (2006)
[4] Grahne, G., Zhu, J.: Fast Algorithms for Frequent Itemset Mining Using FP-Trees. IEEE Trans. Knowl. Data Eng. 17(10), 1347–1362 (2005)
[5] Han, J., Pei, J., Yin, Y., Mao, R.: Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*. Data Min. Knowl. Disc. 8(1), 53–87 (2004)
[6] Knuth, D.: The Art of Computer Programming, vol 3: Sorting and Searching. Addison Wesley, Reading (1973)
[7] Lam, H.T., Calders, T.: Mining Top-K Frequent Items in a Data Stream with Flexible Sliding Windows. In: Proc. ACM SIGKDD, pp. 283–292 (2010)
[8] Liu, G., Lu, H., Lou, W., Xu, Y., Yu, J.X.: Efficient Mining of Frequent Patterns Using Ascending Frequency Ordered Prefix-Tree. Data Min. Knowl. Disc. 9(3), 249–274 (2004)
[9] Liu, G., Lu, H., Yu, J.X., Wang, W., Xiao, X.: Afopt: An Efficient Implementation of Pattern Growth Approach. In: Proc. IEEE ICDM Workshop FIMI (2003)
[10] Pietracaprina, A., Zandolin, D.: Mining Frequent Itemsets Using Patricia Tries*. In: Proc. IEEE ICDM Workshop FIMI (2003)
[11] Schmidt-thieme, L.: Algorithmic Features of Eclat. In: Proc. IEEE ICDM Workshop FIMI (2004)
[12] Tsay, Y.J., Hsu, T.J., Yu, J.R.: FIUT: A New Method for Mining Frequent Itemsets. Inf. Sci. 179(11), 1724–1737 (2009)
[13] Zaki, M.J., Gouda, K.: Fast Vertical Mining Using Diffsets. In: Proc. ACM SIGKDD, pp. 326–335 (2003)

Multi-objective Optimization for Incremental Decision Tree Learning

Hang Yang, Simon Fong, and Yain-Whar Si

Department of Science and Technology, University of Macau,
Av. Padre Tomás Pereira Taipa, Macau, China
ya97404@gmail.com, {ccfong,fstasp}@umac.mo

Abstract. Decision tree learning can be roughly classified into two categories: static and incremental inductions. Static tree induction applies greedy search in splitting test for obtaining a global optimal model. Incremental tree induction constructs a decision model by analyzing data in short segments; during each segment a local optimal tree structure is formed. Very Fast Decision Tree [4] is a typical incremental tree induction based on the principle of Hoeffding bound for node-splitting test. But it does not work well under noisy data. In this paper, we propose a new incremental tree induction model called incrementally Optimized Very Fast Decision Tree (iOVFDT), which uses a multi-objective incremental optimization method. iOVFDT also integrates four classifiers at the leaf levels. The proposed incremental tree induction model is tested with a large volume of data streams contaminated with noise. Under such noisy data, we investigate how iOVFDT that represents incremental induction method working with local optimums compares to C4.5 which loads the whole dataset for building a globally optimal decision tree. Our experiment results show that iOVFDT is able to achieve similar though slightly lower accuracy, but the decision tree size and induction time are much smaller than that of C4.5.

Keywords: Decision Tree, Classification, Incremental Optimization, Stream Mining.

1 Introduction

How to extract knowledge efficiently from massive data has been a popular research topic. A decision tree, which presents the knowledge in a tree-like format, can be easily understood by both human and machine. Due to the high degree of comprehensibility, considered as one of the most important methods for classification.

In general, there are roughly two approaches for decision tree learning. The first approach loads full data, multi-scanning and analyzing them. This process builds a static tree model by greedy search, i.e. ID3 [1], C4.5 [2], CART [3]. When new data come, the whole data (including historical and fresh data) is re-loaded to update algorithm. The second approach only requires loading a small part of samples in terms of Hoeffding bound and comparing the best two values of heuristic function for node-splitting test, i.e. VFDT [4] (which will be introduced in Section 2) and its extensions [7,9,12,13]. Besides, static decision tree provides a global optimal model because it computes across the full samples by greedy search. Incremental tree maintains a local optimal model because it computes on a sufficient part of samples.

One challenge to decision tree learning is associated with noise, which generally renders a data stream "imperfect". The size of a decision tree model will grow excessively large under noisy data, so is an undesirable effect known as over-fitting. The imperfection significantly impairs the accuracy of a decision tree classifier through the confusion and misclassification prompted by the inappropriate data.

For static decision tree learning, pruning algorithms help keep the size of the decision tree in check, although the majority are post-pruning techniques that remove relevant tree paths after a whole model has been built from a stationary dataset [5, 6]. For incremental decision tree learning, post-pruning is not suitable because no extra time is available for stopping tree building and pruning the branches under high-speed data streams environment. It is said that the excessive invocation of tie breaking can cause significant decline in VFDT performance on complex and noise data [12], even with the additional condition by the parameter τ. MVFDT [7] uses an adaptive tie-breaking to reduce tree size for incremental tree.

In this paper, we propose a new incremental decision tree induction inheriting the usage of Hoeffding bound in splitting test, so called Incrementally Optimized Very Fast Decision Tree (iOVFDT). It contains a multi-objective incremental optimization mechanism so as to maintain a small tree size and comparable accuracy, even for imperfect data. For higher accuracy, four types of functional tree leaf are integrated with iOVFDT. In the experiment, we compare iOVFDT to a classical static tree induction C4.5 and pre-pruning incremental tree induction MVFDT. The objective of this paper is to shed light into the following research questions. What are the significant differences between static (global optimum) and incremental (local optimum) decision tree? The answer can be found in experiment and discussion sections, which also show the superior performance of our new algorithm.

The remainder of this paper is organized as follows. In the next section, we describe the classification problem for decision tree. In Section 3, we define the optimization problem for decision tree. Our new algorithm iOVFDT is presented in Section 4. Moreover, we provide the experimental comparison and discussion in Section 5. Finally, Section 6 concludes this paper.

2 Optimization Problem for Decision Tree

Suppose D is the full set of data samples with the form (X, y), where X is a vector of d attributes and y is the actual discrete class label. Attribute X_i is the i^{th} attribute in X and is assigned a value of $X_{i1}, X_{i2} \ldots X_{ij}$, where j is the range of attribute X_i, $|X_i| = j$ and $1 \le i \le d$. Class y_k is the k class in y and is assigned a value of $y_1, y_2 \ldots y_k$, where K is the total number of discrete classes. The classification problem for decision tree is defined as follows: construct a decision tree classifier $DT(X)$ so as to satisfy a classifying goal $DT(X) \leftarrow \hat{y}$, which uses the attribute vector X to provide a predicted class $\hat{y}$. The tree induction builds a tree model from a set of alternatives, minimizing the error between predicted class and actual class (1).

$$\text{minimize} \sum_{k=1}^{K} |Error_k|, where\ Error_k = \begin{cases} 1, & if\ \widehat{y_k} \neq y_k \\ 0, & otherwise \end{cases} \quad (1)$$

The static decision tree learning, i.e. ID3, C4.5, CART, etc., looks for an attribute with the best value of heuristic function *H(.)* as splitting-attribute by greedy search. Post-pruning mechanism removes noisy branches so as to minimize an error-based cost function after full tree built. Hence, it improves accuracy by reducing tree size. The constructed tree model *DT* searches a global optimal solution from the entire dataset *D* so far timestamp *t*, where $D = \sum_1^t D_i$. When new data D_{t+1} comes at timestamp *t+1*, it re-computes on full data D' to update *DT*, where $D' = \sum_1^{t+1} D_i$.

$$HB = \sqrt{\frac{R^2 \ln(\frac{1}{\delta})}{2n}}. \quad (2)$$

where *R* is the range of classes distribution and *n* is the number of instances which have fallen into a leaf, δ is the confidence to support this evaluation. Different from static tree learning, incremental decision tree learning operates continuously arrival data $D_1, D_2, \ldots, D_t$. In the *t th* splitting test, it only scans the newly received data D_t and update the sufficient statistics by Hoeffding bound (*HB*) in (2). Hence incremental decision tree is also called Hoeffding tree (*HT*). Let X_{ja} be the attribute X_j with the highest value of *H(.)*, X_{jb} be the attribute with the second-highest *H(.)*. $\Delta H = H(X_{ja}) - H(X_{jb})$ is the difference between the two top quality attributes. If $\Delta H > HB$ with *n* samples observed in leaf, while the *HB* states with probability $1\text{-}\delta$, that X_{ja} is the attribute with highest value in *H(.)*, then the leaf is converted into a decision node which splits on X_{ja}.

The constructed tree is a local optimum that satisfies the data *D* at timestamp *t*. Said it a local optimum because we never know what are the new arrival data at timestamp *t+1*, even if they contains imperfect values. Let p_l be the probability that an example that reaches level *l* in a decision tree falls into a leaf at that level. If HT and DT use the same heuristic function for node-splitting evaluation, the possibility that the $HT_\delta \neq DT$ is not greater than δ/p, where $E(HT \neq DT) \leq \delta/p$ [4]. Therefore, we can know that: for the full data *D*, where $D = \sum_1^t D_i$ at timestamp *t*, an incremental tree HT_t uses the same *H*(.) with *DT* that tree-branches of HT_t should be a subset of *DT* that $HT_t \subset DT$ with probability δ/p at least.

3 Incrementally Optimized Very Fast Decision Tree (iOVFDT)

3.1 Metrics

Here section will provide iOVFDT in detailed. The model is growing incrementally so as to update an optimal decision tree under continuously arriving data. Suppose that a decision tree optimization problem Π is defined as a tuple (X, HT, Φ). The set *X* is a collection of objects to be optimized and the feasible Hoeffding tree HT solutions are subsets of *X* that collectively achieve a certain optimization goal. The set of all feasible solutions is $HT \subseteq 2^X$ and $\Phi: HT \rightarrow \mathbb{R}$ is a cost function of these solutions. The optimal decision tree *HT** exists if *X* and Φ are known, and the subset S is the set of solutions meets the objective function where *HT** is the optimum in this set.

Therefore, the incremental optimization functions can be expressed as a sum of several sub-objective cost functions:

$$\Phi(HT_x) = \bigcup_{D=1}^{M} \Phi_D(HT_x) \tag{3}$$

where $\Phi_m : HT \to \mathbb{R}$ is a continuously differentiable function and M is the number of objects in the optimization problem. The optimization goal is given in (4):

$$minimize\ \Phi(HT_x)\ subject\ to\ HT_x \in X \tag{4}$$

iOVFDT uses $HT(X) \to \hat{y}$ to predict the class when a new data sample (X, y) arrives. So far timestamp t, the prediction accuracy $accu_t$ defined as:

$$accu_t = \frac{\sum_{i=1}^{t} Predict(D_i)}{|D_t|} \tag{5}$$

$$Predict(D_i) = \begin{cases} 1, if\ \widehat{y_k} = y_k \\ 0, if\ \widehat{y_k} \neq y_k \end{cases} \tag{6}$$

To measure the utility of the three dimensions via the minimizing function in (4), the measure of prediction accuracy is reflected by the prediction error in (7):

$$\Phi_1 = erro_t = 1 - accu_t \tag{7}$$

iOVFDT is a new methodology for building a desirable tree model by combining with an incremental optimization mechanism and seeking a compact tree model that balances the objects of tree size, prediction accuracy and learning time. The proposed method finds an optimization function $\Phi(HT_x)$ in (3), where $M = 3$. When a new data arrive, it will be sorted from the root to a leaf in terms of the existing HT model.

When a leaf is being generated, the tree size grows. A new leaf is created when the tree model grows incrementally in terms of newly arrival data. Therefore, up to timestamp t the tree size can be defined as:

$$\Phi_2 = size_t = \begin{cases} size_{t-1} + 1 & , if\ \Delta\overline{H} > HB \\ size_{t-1} & , otherwise \end{cases} \tag{8}$$

iOVFDT is a one-pass algorithm that builds a decision model using a single scan over the training data. The *sufficient statistics* that count the number of examples passed to an internal node are the only updated elements in the one-pass algorithm. The calculation is an incremental process, which tree size is “plus-one” a new splitting-attribute appears. It consumes little computational resources. Hence, the computation speed of this “plus one” operation for a new example passing is supposed as a constant value R in the learning process. The number of examples that have passed within an interval period of in node splitting control determines the learning time. n_{min} is a fixed value for controlling interval time checking node splitting.

$$\Phi_3 = time_t = R \times (n_{y_k} - n_{min}) \tag{9}$$

Suppose that n_{y_k} is the number of examples seen at a leaf y_k and the condition that checks node-splitting is $n_{y_k} mod\ n_{min} = 0$. The learning time of each node splitting is the interval period – the time defined in (9) – during which a certain number of examples have passed up to timestamp t.

Returning to the incremental optimization problem, the optimum tree model is the HT_x structure with the minimum $\phi(x)$. A triangle model is provided to illustrate the relationship amongst the three dimensions – the prediction accuracy, the tree size and the learning time. The three dimensions construct a triangle utility function shown in Figure 1. A utility function computes the area of this triangle, reflecting a relationship amongst the three objects in (10):

$$\Phi(HT_x) = \frac{\sqrt{3}}{4} \cdot (Erro_x \cdot Size_x + Erro_x \cdot Time_x + Size_x \cdot Time_x) \quad (10)$$

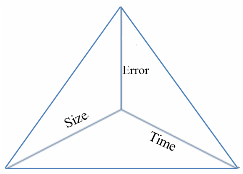


Fig. 1. A multiple objectives optimization model

The area of this triangle $\Phi(HT_x)$ changes when node splitting happens and the HT updates. A min-max constraint of the optimization goal in (4) controls the node splitting, which ensures that the new tree model keeps a $\Phi(HT_x)$ within a considerable range. Suppose that $Max.\Phi(HT_x)$ is a HT model with the maximum utility so far and $Min.\Phi(HT_x)$ is a HT model with the minimum utility. The optimum model should be within this min-max range, near $Mean.\Phi(HT_x)$:

$$Mean.\Phi(HT_x) = \frac{\text{Max}.\Phi(HT_x) - Min.\Phi(HT_x)}{2} \quad (11)$$

According to the Chernoff bound [8], we know:

$$|\text{Opt}.\Phi(HT_x{}^*) - Mean.\Phi(HT_x)| \leq \sqrt{\frac{\ln(1/\delta)}{2n}} \quad (12)$$

where the range of $\Phi_x(HT_x)$ is within the min-max model and $\text{Min}.\Phi(HT_x) < \text{Opt}.\Phi(HT_x{}^*) < \text{Max}.\Phi(HT_x)$. Therefore, if $\Phi(HT_x)$ goes beyond this constraint, the existing HT is not suitable to embrace the new data input and the tree model should not be updated. Node-splitting condition is adaptively optimized in iOVFDT such that: $\Delta\overline{H} > HB$ or $\text{Opt}.\Phi(HT_x{}^*) > Max.\Phi(HT_x)$ or $\text{Opt}.\Phi(HT_x{}^*) < Min.\Phi(HT_x)$,

3.2 Functional Tree Leaf Integration

Functional tree leaf [9], can further enhance the prediction accuracy via the embedded Naïve Bayes classifier.. In this paper, we embed the functional tree leaf to improve the performance of prediction by HT model. When these two extensions – an optimized node-splitting condition ($\Delta\overline{H} > HB$ or $\text{Opt}.\Phi(HT_x{}^*) > Max.\Phi(HT_x)$ or $\text{Opt}.\Phi(HT_x{}^*) < Min.\Phi(HT_x)$) and a refined prediction using the functional tree leaf – are used together, the new decision tree model is able to achieve unprecedentedly good performance, although the data streams are perturbed by noise and imbalanced class distribution.

For the actual classification, iOVFDT uses a decision tree model HT to predict the class label $\widehat{y_k}$ with functional tree leaf when a new sample (X, y) arrives, defined as $HT(X) \rightarrow \widehat{y_k}$. The predictions are made according to the observed class distribution (OCD) in the leaves called functional tree leaf $\mathcal{F}$. Originally in VFDT, the prediction uses only the majority class $\mathcal{F}^{MC}$. The majority class only considers the counts of the class distribution, but not the decisions based on attribute combinations. The naïve Bayes $\mathcal{F}^{NB}$ computes the conditional probabilities of the attribute-values given a class at the tree leaves by naïve Bayes network. As a result, the prediction at the leaf is refined by the consideration of each attribute's probabilities. To handle the imbalanced class distribution in a data stream, a weighted naïve Bayes $\mathcal{F}^{WNB}$ and an error-adaptive $\mathcal{F}^{Adaptive}$ are proposed in this paper. These four types of functional tree leaves are discussed in following paragraphs.

Let Sufficient statistics n_{ijk} be an incremental count number stored in each node in the iOVFDT. Suppose that a node $Node_{ij}$ in HT is an internal node labeled with attribute x_{ij} and k is the number of classes distributed in the training data, where $k \geq 2$. A vector V_{ij} can be constructed from the sufficient statistics n_{ijk} in $Node_{ij}$, such that $V_{ij} = \{n_{ij1}, n_{ij\,2} \ldots n_{ij\,k}\}$. V_{ij} is the OCD vector of $Node_{ij}$. OCD is used to store the distributed class count at each tree node in iOVFDT to keep track of the occurrences of the instances of each attribute.

***Majority Class Functional Tree Leaf*:** In the OCD vector, the majority class $\mathcal{F}^{\mathrm{MC}}$ chooses the class with the maximum distribution as the predictive class in a leaf, where $\mathcal{F}^{\mathrm{MC}}$: arg max $r = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, and where $0<r<k$.

***Naïve Bayes Functional Tree Leaf*:** In the OCD vector $V_{i,j} = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, where r is the number of observed classes and $0<r<k$, the naïve Bayes $\mathcal{F}^{\mathrm{NB}}$ chooses the class with the maximum possibility, as computed by the naïve Bayes, as the predictive class in a leaf. $n_{ij,r}$ is updated to $n'_{i,j,r}$ by the naïve Bayes function such that $n'_{i,j,r} = \mathrm{P}(X|C_f) \cdot \mathrm{P}(C_f) \,/\, \mathrm{P}(X)$, where X is the new arrival instance. Hence, the prediction class is $\mathcal{F}^{\mathrm{NB}}$: arg max $r = \{ n'_{i,j,1}, n'_{i,j,2} \ldots n'_{i,j,r} \ldots n'_{i,j,k} \}$.

***Weighted Naïve Bayes Functional Tree Leaf*:** In the OCD vector $V_{i,j} = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, where k is the number of observed classes and $0<r<k$, the weighted naïve Bayes $\mathcal{F}^{\mathrm{WNB}}$ chooses the class with the maximum possibility, as computed by the weighted naïve Bayes, as the predictive class in a leaf. $n_{i,j,r}$ is updated to $n'_{i,j,r}$ by the weighted naïve Bayes function such that $n'_{i,j,r} = \omega_r \cdot \mathrm{P}(X|C_f) \cdot \mathrm{P}(C_f) \,/\, \mathrm{P}(X)$, where X is the latest received instance and the weight is the probability of class i distribution among all the observed samples, such that $\omega_r = \prod_{r=1}^{k} (v_r / \sum_{r=1}^{k} v_r)$, where $n_{i,j,r}$ is the count of class r. Hence, the prediction class is $\mathcal{F}^{\mathrm{WNB}}$: arg max $r = \{ n'_{i,j,1}, n'_{i,j,2} \ldots n'_{i,j,r} \ldots n'_{i,j,k} \}$.

Adaptive Functional Tree Leaf: In a leaf, suppose that $V_{\mathcal{F}}{}^{\mathrm{MC}}$ is the OCD with the majority class $\mathcal{F}^{\mathrm{MC}}$; suppose $V_{\mathcal{F}}{}^{\mathrm{NB}}$ is the OCD with the naïve Bayes $\mathcal{F}^{\mathrm{NB}}$ and suppose that $V_{\mathcal{F}}{}^{\mathrm{WNB}}$ is the OCD with the weighted naïve Bayes $\mathcal{F}^{\mathrm{WNB}}$. Suppose that y is the true class of a new instance X and $E_{\mathcal{F}}$ is the prediction error rate using a $\mathcal{F}$. $E_{\mathcal{F}}$ is calculated by the average $E=error_i/n$, where n is the number of examples and $error_i$ is the number of examples mis-predicted using $\mathcal{F}$. The adaptive Functional Tree Leaf chooses the class with the minimum error rate predicted by the other three strategies, where $\mathcal{F}^{\mathrm{Adaptive}}$: arg min $\mathcal{F} = \{E_{\mathcal{F}}{}^{\mathrm{MC}}, E_{\mathcal{F}}{}^{\mathrm{NB}}, E_{\mathcal{F}}{}^{\mathrm{WNB}}\}$.

3.3 Tree-Building Process

In this section, a pseudo code summaries the process of tree-growth presented in previous parts. When new data stream comes, it will be sorted by current HT and given a predictive class label. The OCD, which is stored on those pass-by internal nodes, is updated. Comparing the predicted class to the actual class, the prediction error is updated (Line 1 – 5). If the number of samples seen so far is greater than the pre-defined interval number, the node-splitting evaluation should be performed (Line 7 – 19). If node-splitting condition that the difference of best two values of heuristic function *H(.)* is greater than *HB*, or the value of optimization function is out of a min-max range, the attribute with the highest *H(.)* value should split to a new leaf (Line 12 – 17). Meanwhile, new model size and learning time are updated by (8) and (9).

```
Input:
  $D_t$: a data stream (X,y) arriving at timestamp t;
  $H(.)$: the heuristic function for splitting test;
  $\mathcal{F}$: a strategy of functional tree leaf;
  $n_{min}$: the minimum interval between node-splitting tests;
  $\delta$: a desired probability for Hoeffding bound;
Output: Incremental decision tree HT
1.A data stream $D_t = (X,y)$ arrives;
2.If HT isn't initialized, let HT be a tree with a single leaf
    l (the root);
3.Sort $D_t$ from the root to a leaf by HT, using • to give a
    predicted class $\widehat{y_k} \leftarrow HT(X)$;
4.Update OCD on each pass-by node;
5.Compare predicted class $\widehat{y_k}$ to actual class $y_k$, and update
    error in (7);
6.Let $n_k$ be the number of instances seen at the leaf with class
    $y_k$.
7.If all instances seen so far at leaf $k$ don't belong to the
    same class, and ($n_k \, mod \, n_{min} = 0$){
8.   Update the learning time in (9);
9.   Let $X_a$ and $X_b$ the attributes with highest and the 2nd
     highest heuristic function H(.).
10.    Let $\Delta H = H(X_a) - H(X_b)$;
11.    Compute HB and update $\Phi(HT_t)$ in (3);
12.    If ($\Delta H > HB, or\ \Phi(HT_t) > Max.\Phi(HT), or\ \Phi(HT_t) < Min.\Phi(HT)${
13.        Replace leaf k by a node splits on $X_a$;
14.        Update the tree size in (8);
15.        If $\Phi(HT_t) > Max.\Phi(HT)$ then $Max.\Phi(HT) = \Phi(HT_t)$
16.        If $\Phi(HT_t) < Min.\Phi(HT)$ then $Min.\Phi(HT) = \Phi(HT_t)$
17.    } End-if;
18.    Reset OCD on the new leaf;
19. } End-if;
20. Return $HT_t$
```

4 Experiment

4.1 Setup

In this section, we compared C4.5, MVFDT to iOVFDT tree inductions in order to show the difference between global (C4.5) and local (MVFDT, iOVFDT) search optimal tree models. The heuristic function for splitting attribute is information gain. The experimental platform was built on Java. For C4.5, WEKA [10] tree classification J48 was used, both un-pruning and pruning mechanisms; for MVFDT, there are loose and strict pruning mechanisms, both of which have been verified to outperform VFDT described in [7]; for iOVFDT, it was programmed and integrated with MOA [11]. Majority Class (MC), Naïve Bayes (NB), Weighted Naïve Bayes (WNB) and Hybrid Adaptive (ADP) functional tree leaves are integrated in iOVFDT.

The running environment was a Windows 7 PC with an Intel 2.8GHz CPU and 8G RAM. Besides, un-pruned and pruned C4.5 algorithms were applied in this experiment so as to analyze the tree size. The heuristic evaluation of node-splitting was information gain in both methods.

4.2 Datasets

The datasets, including discrete, continuous and mixed attributes, were either synthetically generated by MOA generator, or downloaded from UCI repository.

Table 1. The description of experimental datasets

Data Name	#Nominal	#Numeric	#Class	Source	Size
LED24	24	0	10	Synthetic	10^6
Waveform	0	21	3	Synthetic	10^6
Cover Type	42	12	7	UCI	581,012

Synthetic Data *LED data* was generated by MOA. We added 10% noisy data to simulate imperfect data streams. The LED24 problem used 24 binary attributes to classify 10 different classes. *Waveform* was generated by the MOA generator. The goal of this task was to differentiate between three different classes of Waveform. It had 21 numeric attributes contained noise. **UCI Data** *Cover Type* was used to predict forest cover types from cartographic variables. It is a typical imbalanced class distribution data that all are real life samples.

4.3 Result Analysis

The performance measurements were evaluated in three aspects: accuracy, tree size and learning speed. The measurement of accuracy was 10 folds cross-validation. The number of leaves in the tree mode computed tree size. Learning speed was reflected by the time taken to build decision tree.

Discrete Data. Obviously, un-pruned C4.5 resulted lowest accuracy, biggest tree size and slowest speed (Figure 2). Hence un-pruned C4.5 had the worst performance in this test. Compared with iOVFDT, pruned C4.5 had better accuracy for small data size. But when the data size grew, iOVFDT outperformed pruned C4.5, because of the smaller tree size and the faster speed. iOVFDT with Hybrid Adaptive function tree leaf obtained the best accuracy but similar speed with the others. Both strict and loose pruning MVFDT had lower accurate than iOVFDT.

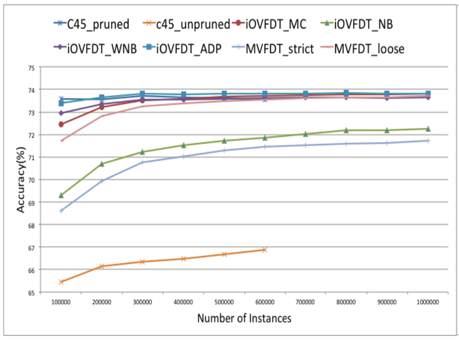


Fig. 2a. Prediction Accuracy for LED24 data

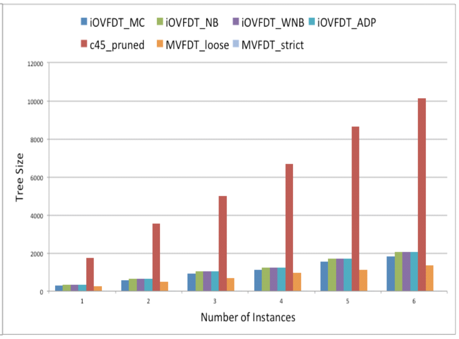


Fig. 2b. Tree size for LED 24

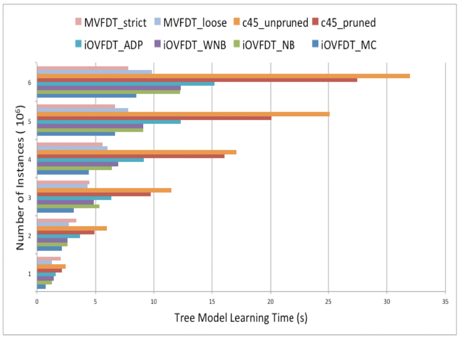


Fig. 2c. Learning time for LED 24

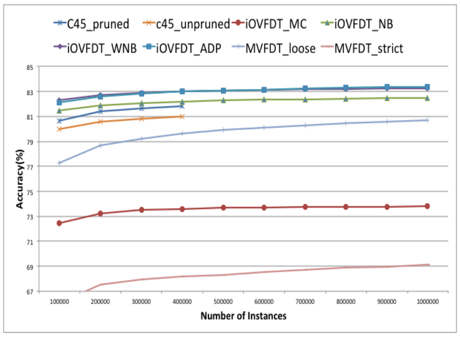


Fig. 3a. Prediction Accuracy for Waveform data

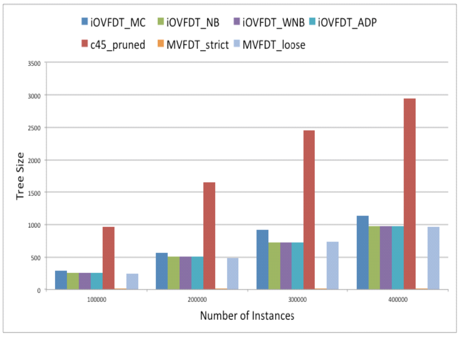


Fig. 3b. Tree size for Waveform

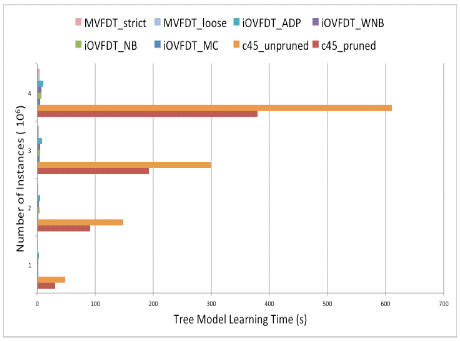
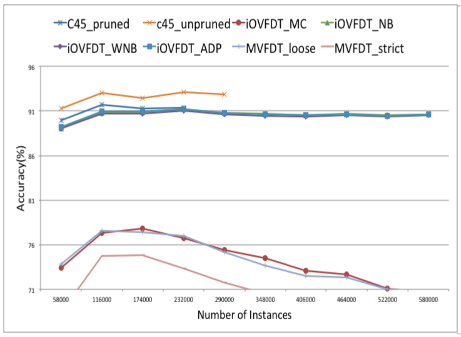


Fig. 3c. Learning time for Waveform



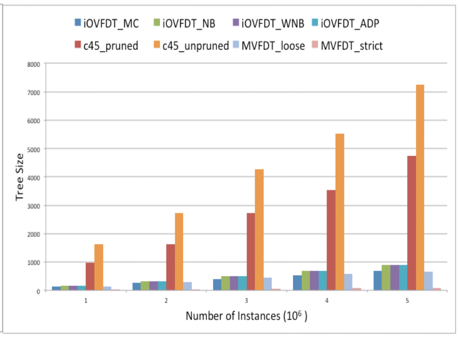


Fig. 4a. Prediction Accuracy for Cover Type data **Fig. 4b.** Tree size for Cover Type

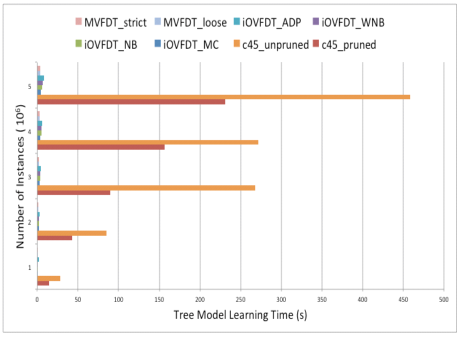


Fig. 4c. Learning time for Cover Type

Continuous Data. iOVFDT$_{MC}$ and MVFDT used majority class in leaf that didn't have the mechanism dealing with numeric data, so the accuracy of them was the worst in this test. For large data size, iOVFDT$_{ADP}$ had the best accuracy and smaller tree than the others. The speed of C4.5 was dramatically longer than iOVFDT (Figure 3).

Mixed Data. Although C4.5 had a little higher accuracy than iOVFDT, it took much longer learning time. iOVFDT$_{ADP}$ had similar accurate with pruned C4.5 but smaller model size. iOVFDT$_{MC}$ had worse performance than other iOVFDTs, and its accurate declined with more data arrived. In addition, the result shows iOVFDT had an obviously better accurate than MVFDT (Figure 4).

4.4 Discussion

For noise-included data, we designed an experiment to show the effects for the different decision trees. 10% noisy data were inserted into LED24 dataset, alternatively, at every other ten thousand instances. Full data had one million instances. C4.5 loaded the full data for training the decision tree and obtained a global accuracy around 91%. iOVFDT loaded the data incrementally, and trained the decision tree progressively. As it can be seen from Figure 5, the local accuracy of iOVFDT was affected by the noisy data. The accuracy dropped whenever noise appeared at the section of data, and it bounced back as soon as the next section of noise-free data emerged. The fluctuation continued but gradually diminished over the staggering input data with a damping effect. As it was shown in Figure 6, the performance curve of iOVFDT was converging when the total data volume increased. In order to verify the steady-state of the iOVFDT model without the need of using an exceedingly large amount training data, a fitting curve was derived from the existing iOVFDT curve by using Single Moving Average algorithm. The forecasted performance was estimated to be 88.05% from the fitted curve, with the reliability of mean absolute percentage error being 0.98%, Thiel's U value at 0.5118 (which is far better than random guess) and Durbin-Watson value at 1.21 that supported the fact of autocorrelation exhibited by the fitted curve.

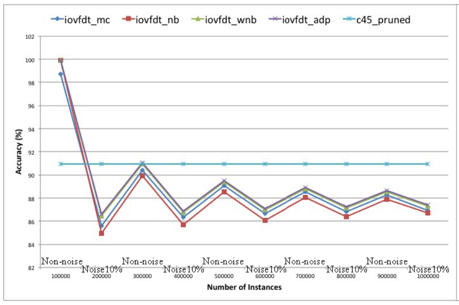


Fig. 5. Performance of global (C4.5) and local (iOVFDT) optimal trees for noise-included data

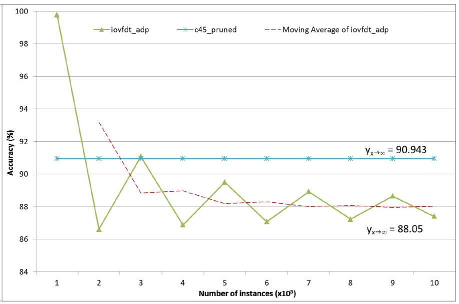


Fig. 6. Longitudinal accuracy of global (C4.5) and local (iOVFDT) optimal trees

In other words, when the training data size approaches infinity the expected accuracy of the iOVFDT model should be 88.05% as implied by the fitted curve. This observation essentially meant that iOVFDT could achieve an accuracy of 88.05% in a long run; and it would do so by incrementally updating its decision tree while new input data continuously streamed in. In contrast, C4.5 required first accepting all the data for learning a decision tree for obtaining a globally optimal accuracy value. The performance difference of 2.893% gained by C4.5 was done at the cost of loading in the full dataset that sometimes may be impractical. Our experiment in this case has shown that the accuracy of iOVFDT that was affected by a series of local optimums was still reasonably substantial in long run. And iOVFDT outperformed C4.5 by the merits of compact tree size, fast learning process and the incremental ability to handle infinite data streams.

5 Conclusion

In this paper, we propose a new incremental decision tree learning so called incrementally Optimized Very Fast Decision Tree (iOVFDT). Different from static tree induction, iOVFDT updates the decision model without scanning the full data in infinite data streams scenario. An incremental optimization, which considers prediction accuracy, model size and learning time, is also used for node-splitting constraint in tree growing. In addition, four types of functional tree leaf are also embedded to improve the prediction accuracy.

In our experiment, the comparison of static decision tree and incremental decision tree is presented, using the same heuristic function for splitting test. The experimental result compares iOVFDT to C4.5 and MVFDT, and we found that: iOVFDT has significantly smaller tree size and faster learning speed than C4.5. It also has higher accuracy than MVFDT in terms of the three-object incremental optimization mechanism. In the second experiment, the dataset is partitioned by interleaving segments of perfect and noisy data, so as to simulate existence of local optimums and to test the difference between global and local optimal decision trees. In general, global optimal decision tree, which was constructed by loading full data to find out a global optimal model in C4.5 tree induction, was able to obtain a high accuracy for small scale data. Because of multi-scanning over the full dataset, the size of decision tree model was very large even pruning was applied and the entire process was relatively slow. Local optimal decision tree, which was built by iOVFDT incrementally, was demonstrated to be applicable for large or, even infinite datasets. The proposed functional tree leaf of Hybrid Adaptive had the best performance in synthetic data. Compact tree size and short learning time make incremental model practical in real-time applications. The accuracy though may not be the highest, is on par with C4.5. By sacrificing slight accuracy, iOVFDT can be effectively used to handle infinite streams as well as to build an optimized tree within a short time.

Reference

1. Quinlan, R.: Induction of Decision Trees. Machine Learning 1(1), 81–106 (1986)
2. Quinlan, R.: C4.5: Programs for Machine Learning. MorganKaufmann, San Francisco (1993)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey (1984)
4. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc of 6th ACM SIGKDD, pp. 71–80 (2000)
5. Elomaa, T.: The Biases of Decision Tree Pruning Strategies. In: Hand, D.J., Kok, J.N., Berthold, M. (eds.) IDA 1999. LNCS, vol. 1642, pp. 63–74. Springer, Heidelberg (1999)
6. Bradford, J., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.: Pruning Decision Trees with Misclassification Costs. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 131–136. Springer, Heidelberg (1998)
7. Yang, H., Fong, S.: Moderated VFDT in Stream Mining Using Adaptive Tie Threshold and Incremental Pruning. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 471–483. Springer, Heidelberg (2011)
8. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. Annals of Mathematical Statistics 23, 493–507 (1952)
9. Gama, J., Ricardo, R.: Accurate decision trees for mining high-speed data streams. In: Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 523–528. ACM (2003)
10. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann, San Francisco (2000)
11. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive Online Analysis. Journal of Machine Learning Research 11, 1601–1604 (2000)
12. Geoffrey, H., Richard, K., Bernhard, P.: Tie Breaking in Hoeffding trees. In: Gama, J., Aguilar-Ruiz, J.S. (eds.) Proceeding Workshop W6: Second International Workshop on Knowledge Discovery in Data Streams, pp. 107–116 (2005)
13. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, pp. 97–106 (2001)

Mining Contextual Preference Rules for Building User Profiles

Sandra de Amo[1], Mouhamadou Saliou Diallo[2,3], Cheikh Talibouya Diop[3], Arnaud Giacometti[2], Haoyuan D. Li[2], and Arnaud Soulet[2]

[1] Universidade Federal de Uberlândia, Brazil
deamo@ubu.br
[2] Université de Tours, France
forename.surname@univ-tours.fr
[3] Université Gaston Berger de Saint-Louis, Sénégal
forename.surname@ugb.edu.sn

Abstract. The emerging of ubiquitous computing technologies in recent years has given rise to a new field of research consisting in incorporating context-aware preference querying facilities in database systems. One important step in this setting is the *Preference Elicitation* task which consists in providing the user ways to inform his/her choice on pairs of objects with a minimal effort. In this paper we propose an automatic preference elicitation method based on mining techniques. The method consists in extracting a *user profile* from a set of user preference samples. In our setting, a profile is specified by a set of contextual preference rules verifying properties of soundness and conciseness. We evaluate the efficacy of the proposed method in a series of experiments executed on a real-world database of user preferences about movies.

1 Introduction

Elicitation of preferences consists basically in providing the user a way to inform his/her preferences on objects belonging to a dataset, with a minimal effort for him/her. It can be achieved by following different strategies: (a) by using a query interface where users are asked to express their preferences [3], or (b) by capturing implicit user's choices and applying preference mining algorithms [9]. The first alternative is not efficient since the users in general are not able to express their preferences in an exact and consistent way. This paper is focused on the second alternative for preference elicitation. We assume our data is constituted by pairwise comparisons. We do not discuss in this paper the way the user informed his/her choices, knowing that different strategies can be applied [15]. Our method simply assume that *pairs of objects* expressing the user preferences have been collected somehow. The running example below illustrates the preference mining problem we tackle in this paper. In this example we assume that the user preferences are informed by means of the number of clicks on certain tags.

Motivating Example. A web service regularly provides recommendation about movies to its subscribers. In order to capture my preferences on films without being too annoying and intrusive, the service offers me a trial period during which I can freely access information about films. I indicate the films I am interested in by clicking on different tags. For instance, I can click on tags *Action*, *Spielberg* and *War* to indicate that I am interested in obtaining information on films directed by Steve Spielberg, with a script based on a war story, and having a lot of action. My clicks are automatically collected during the trial period. The relation $\mathcal{D}$ depicted on Table 1 presents some of my access during the trial period. Tags A, B, C, D and E stands for *Spielberg*, *Tom Hanks*, *Action*, *Leonardo di Caprio* and *War* respectively. Each t_i $(i = 1, ...5)$ represents the set of tags I selected each time I accessed the service. They are called *transactions.* Let us suppose that during the trial period I accessed the service ten times by clicking on the set of tags t_1 and only five times by clicking on the set of tags t_3. Thus, I implicitly indicated that I am more interested on films associated to tags t_1 than to tags t_3 as indicated by the first pair (t_1, t_3) in relation $\mathcal{P}$ depicted on Table 1. Notice that both t_1 and t_3 contain the tags A and C. Between them I prefer the one containing the tag D than the one containing the tag B. So, the following *contextual preference rule* can be inferred: Between two *action* movies directed by *Spielberg* I prefer the one played by Leonardo di Caprio than the one played by Tom Hanks. Tags *Action* and *Spielberg* constitute the *context* of the rule. Notice that some pairs of transactions (for instance, (t_1, t_2)) do not appear in relation $\mathcal{P}$, indicating that the number of clicks on each of these sets of tags is the same or differs by a *negligible* amount of clicks (below a given threshold).

$\mathcal{D}$

Tid	Transactions
t_1	$A\ \ \ C\ D$
t_2	$A\ B\ \ \ D$
t_3	$A\ B\ C\ \ \ E$
t_4	$C\ D$
t_5	$A\ B$

$\mathcal{P}$

Pid	Preference
p_1	$\langle t_1, t_3 \rangle$
p_2	$\langle t_2, t_3 \rangle$
p_3	$\langle t_2, t_4 \rangle$
p_4	$\langle t_3, t_4 \rangle$
p_5	$\langle t_4, t_5 \rangle$

$$\begin{tikzcd}
t_1 : ACD \arrow[d, "p_1"'] & t_2 : ABD \arrow[dl, "p_2"'] \arrow[d, "p_3"] & \\
t_3 : ABCE \arrow[r, "p_4"'] & t_4 : CD \arrow[r, "p_5"'] & t_5 : AB
\end{tikzcd}$$

Fig. 1. Preferences on Transactions

In this paper we propose a method for building the profile of a user from a sample of his/her preferences previously captured by the system. A user's profile is specified by a set of *contextual preference rules* [1] satisfying some interestingness criteria, namely *soundness* and *conciseness.* The *soundness* property guarantees that the preference rules specifying the profiles are in agreement with a large set of the user preferences, and contradicts a small number of them. On the other hand, *conciseness* implies that profiles are small sets of preference rules. We argue that this approach has many advantages if compared to other preference models found in the literature. The model is easy to understand and manage due to its conciseness and its *qualitative* aspect (it is constituted by a set of preference rules and it does not employ score function explicitly assigning grades to each transaction [5,8,11,16]). Moreover, the soundness property guarantees that our method builds user profiles with good predictive properties.

This paper is organized as follows. In Section 2 we discuss some related work. In Section 3 we rigorously define the mining problems we treat in this paper. Section 4 is dedicated to present the preference rule mining algorithm, whereas Section 5 presents the user profile construction algorithm. In Section 6 we describe and analyze experimental results on real datasets.

2 Related Work

Methods for Preference Learning can be categorized following different criteria such as *Preference Specification* (*qualitative* or *quantitative*) and *Preference Semantics* (the pareto model, conditional preference model). The techniques presented in this section are inherently distinct. Nevertheless they have a common main goal: given a pair of objects, to predict which one is the most preferred.

In a *qualitative approach*, preferences are specified by a compact set of preference rules from which a preference relation can be inferred. The method we propose in this paper follows a qualitative approach. Some other qualitative approaches are [9,10]. In [9] the authors propose a technique for mining user preferences whose underlying model is the *pareto preference model*. Such preference rules are obtained from log data generated by the server when the user is accessing a web site. Another approach to preference mining is presented in [10]. In this work the authors propose using preference samples provided by the user to infer an order on any pair of tuples in the database. Such samples are classified into two categories, the *superior* and *inferior* samples and contain information about some preferred tuples and some non-preferred ones. From these rules, an order is inferred on the tuples. The underlying preference model is the *pareto preference model* as in [9]. In this model, preferences are not conditional or contextual, that is, preferences on values of attributes do not depend on the values of other attributes. Our contextual preference model is more expressive.

In contrast with the above papers, where preferences are specified following a *qualitative* approach, in [5] and [7] algorithms for mining *quantitative* preferences are proposed. In these works preferences are specified by a score function and the main goal is to find automatically a prediction rule which assigns a score to each tuple of the database. The mining task in this approach is sometimes called *learning to rank*. Several efficient methods for learning to rank have been proposed so far in the information retrieval domain, including Rank SVM [11], RankBoost [8], RankNet [5] and AdaRank [16]. In all these methods, the learning task is formalized as classification of object pairs in two classes: correctly or incorrectly ranked. Different classification techniques are employed such as SVM (Rank SVM), Boosting (AdaRank, RankBoost) and Neural Network trained by a Gradient Descent algorithm (RankNet). In comparison, our method has the advantage of making explicit the preferences of the user through the profile.

3 Problem Formalization

3.1 Preference Database and Contextual Preference Rules

Let $\mathcal{I}$ be a set of distinct literals called *items* (or *tags*), an itemset is a subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L} = 2^{\mathcal{I}}$. A transactional dataset $\mathcal{D}$ is a multi-set of itemsets in $\mathcal{L}$. Each itemset, usually called *transaction*, is a database entry. Figure 1 presents a transactional dataset $\mathcal{D}$ where 5 transactions denoted by $t_1, \ldots, t_5$ are described by 5 items denoted by $A, \ldots, E$.

A *preference database* $\mathcal{P} \subseteq \mathcal{D} \times \mathcal{D}$ is a set of pairs of transactions representing a sample of user preferences over the dataset $\mathcal{D}$. Intuitively, a user preference $\langle t, u \rangle \in \mathcal{P}$ means that the user prefers the transaction t to the transaction u. Given a user preference $\langle t, u \rangle \in \mathcal{P}$, t is said to be the *preferred* transaction (according to the user). Figure 1 shows a set of 5 user preferences labeled $p_1, \ldots, p_5$. The preference database plus the transactions are also synthesized by a graph as illustrated in Table 1[1]. We emphasize that in general $\mathcal{P}$ is not necessarily *transitive* as in our running example, since in this particular case the user preferences have been obtained by comparing the number of clicks on each set of tags.

The main objective of this paper is to extract a user profile from a preference database provided by the user. A user profile is specified by a set of preference rules verifying some interesting properties.

Definition 1 (Contextual preference rule [1]). *A contextual preference rule is of the form $i^+ \succ i^- \,|\, X$ where X is an itemset of $\mathcal{L}$, i^+ and i^- are items of $\mathcal{I} \setminus X$.*

The left-hand side of a preference rule specifies the choice while the right-hand side is the context. For instance, $D \succ E \,|\, AB$ means that the context AB leads to choose the item D to the item E. $\mathcal{CP}(\mathcal{L})$ denotes the set of the contextual preference rules based on $\mathcal{L}$ (we often omit the language when it is implicit in the context). Of course, the interest behind $i^+ \succ i^- \,|\, X$ is its ability to compare transactions. A transaction t is preferred to u according to $\pi : i^+ \succ i^- \mid X$, denoted by $t \succ_\pi u$ if $(Xi^+ \subseteq t) \wedge (Xi^- \subseteq u) \wedge (i^- \not\in t) \wedge (i^+ \not\in u)$. For instance, ACD is preferred to $ABCE$ according to the contextual preference rule $D \succ E \,|\, A$ i.e., $ACD \succ_{D \succ E|A} ABCE$.

Naturally, a given contextual preference rule π can *agree with* a user preference $\langle t, u \rangle \in \mathcal{P}$ (i.e. $t \succ_\pi u$) or *contradict* $\langle t, u \rangle \in \mathcal{P}$ (i.e. $u \succ_\pi t$). In both cases, we say that the contextual preference rule *covers* the user preference $\langle t, u \rangle$. For instance, the user preference $p_1 = \langle t_1, t_3 \rangle$ is covered by both $D \succ E \,|\, A$ (agreement) and $B \succ D \,|\, C$ (contradiction).

3.2 The Contextual Preference Rule Mining Problem

Basically, we adapt the support-confidence framework of association rules by considering that the context X and the preference $i^+ \succ i^-$ corresponds respectively to the antecedent and the consequent of an association rule. Thereby, we

[1] For the sake of simplifying the presentation, some arrows obtained by transitivity are not depicted in the graph (for instance the arrow between t_1 and t_4).

analogically define the concept of support, confidence and minimality as interestingness criteria for filtering out non relevant contextual preference rules.

Definition 2 (Support). *The support of a contextual preference rule π in $\mathcal{P}$ is defined as:* $supp(\pi, \mathcal{P}) = \frac{|\{\langle t,u\rangle \in \mathcal{P} \mid t \succ_\pi u\}|}{|\mathcal{P}|}$

The support of a contextual preference rule π (ranged from 0 to 1) estimates the probability that π agrees with a pair of $\mathcal{P}$. The interest of a contextual preference rule increases with its support. For instance, as $ACD \succ_{D \succ E|A} ABCE$ and $ABD \succ_{D \succ E|A} ABCE$, we obtain $supp(D \succ E \mid A, \mathcal{P}) = |\{p_1, p_2\}|/|\mathcal{P}| = 0.4$. Similarly, $supp(D \succ E \mid B, \mathcal{P}) = |\{p_2\}|/|\mathcal{P}| = 0.2$. So, $D \succ E \mid A$ is more interesting than $D \succ E \mid B$.

Now we also need to evaluate the disagrement between a contextual preference rule and the preference database. To this end, the *confidence* of a contextual preference rule π measures the proportion of user preferences in agreement with π among pairs covered by π:

Definition 3 (Confidence). *The confidence of a contextual preference rule π in $\mathcal{P}$ is defined as:* $conf(\pi, \mathcal{P}) = \frac{|\{\langle t,u\rangle \in \mathcal{P} \mid t \succ_\pi u\}|}{|\{\langle t,u\rangle \in \mathcal{P} \mid t \succ_\pi u \vee u \succ_\pi t\}|}$

In other words, the confidence evaluates whether a contextual preference rule contradicts many user preferences. This criterion shows that $D \succ E \mid A$ is more valuable than $D \succ E \mid \emptyset$ because $conf(D \succ E \mid A, \mathcal{P}) = 2/2 = 1$ whereas $conf(D \succ E \mid \emptyset, \mathcal{P}) = 2/3$. The set of all contextual preference rules exceeding a minimal support threshold σ and a minimal confidence threshold κ is denoted by $\mathcal{CP}_{\sigma,\kappa}(\mathcal{L}, \mathcal{P})$ (or $\mathcal{CP}_{\sigma,\kappa}$ in brief).

At this point, the support and the confidence discard respectively the infrequent and unreliable contextual preference rules. But, the redundancies between several contextual preference rules of $\mathcal{CP}_{\sigma,\kappa}$ are not detected. Given the example of Figure 1, we observe that $D \succ E \mid B$ and $D \succ E \mid AB$ have the same support and the same confidence. Intuitively, the contextual preference rule $D \succ E \mid B$ is more relevant than $D \succ E \mid AB$ because its context is smaller. For this purpose, we introduce the notion of *minimal* contextual preference rule:

Definition 4 (Minimal Preference Rule). *A contextual preference rule $i^+ \succ i^- \mid X$ is minimal in $\mathcal{P}$ iff there is no contextual preference rule $i^+ \succ i^- \mid Y$ such that $Y \subset X$ and $supp(i^+ \succ i^- \mid Y, \mathcal{P}) = supp(i^+ \succ i^- \mid X, \mathcal{P})$ and $conf(i^+ \succ i^- \mid Y, \mathcal{P}) = conf(i^+ \succ i^- \mid X, \mathcal{P})$.*

Following on, $\mathcal{MCP}_{\sigma,\kappa}(\mathcal{L}, \mathcal{P})$ (or $\mathcal{MCP}_{\sigma,\kappa}$) denotes the whole set of minimal contextual preference rules having its support and confidence respectively greater than σ and κ. In practice, this minimality criterion drastically reduces the number of contextual preference rules.

Given a sample preference database, the first problem that we consider deals with the extraction of all *interesting* preference rules, i.e. those rules which are minimal and have acceptable support and confidence. More precisely:

Problem 1 (Preference Rule Mining). Given a preference database $\mathcal{P}$, a minimal support threshold σ and a minimal confidence threshold κ, find the set $\mathcal{MCP}_{\sigma,\kappa}$ of minimal contextual preference rules.

Obviously, a naive enumeration of all preference rules for computing $\mathcal{MCP}_{\sigma,\kappa}$ is unfeasible and some pruning criteria are necessary for reducing the search space. In Section 4 we present ContPrefMiner, a levelwise algorithm inspired on Apriori [2] which takes advantage of the downward closure of $\mathcal{MCP}_{\sigma,0}$ to reduce the search space.

3.3 The User Profile Construction Problem

In our approach, a *user profile* is specified by a set of contextual preference rules which is both *concise* and *sound* with respect to the preference samples he/she has previously provided. Roughly speaking, the *conciseness* of a set of preference rules is evaluated by means of its cardinality. On the other hand, the *soundness* of a set of preference rules is evaluated by means of two standard measures, *precision* and *recall* (see Definition 5).

We have to precise how two transactions can be compared according to a set S of contextual preference rules to evaluate the ability of a user profile to make good predictions. First, we say that two transactions are *comparable* with respect to a set S of preference rules if they can be compared by at least one rule in S. Then, one important issue is when two transactions can be compared in different ways using different rules in S. In this paper, we define a total order on the set of contextual preference rules (see Definition 6), and propose to select the *best* preference rule to decide which transaction is the preferred one. More precisely, we say that a transaction $t \in \mathcal{L}$ is preferred to $u \in \mathcal{L}$ according to a user profile S, denoted by $t \succ_S u$, it there exists a preference rule $\pi \in S$ such that $t \succ_\pi u$ and π is the best rule in S that can be used to compare t and u.

In order to evaluate the predictive quality of a user profile, we now introduce the *precision* and *recall* measures as follows:

Definition 5 (Precision and recall). *Given a preference database $\mathcal{P}$ and a set of contextual preference rules S, the precision of $\succ_S$ with respect to $\mathcal{P}$, denoted $Prec(\succ_S, \mathcal{P})$, is defined by:*

$$Prec(\succ_S, \mathcal{P}) = \frac{|\{\langle t, u\rangle \in \mathcal{P} | t \succ_S u\}|}{|\{\langle t, u\rangle \in \mathcal{P} | t \succ_S u \vee u \succ_S t\}|}$$

Moreover, the recall of $\succ_S$ with respect to $\mathcal{P}$, denoted $Rec(\succ_S, \mathcal{P})$, is defined by:

$$Rec(\succ_S, \mathcal{P}) = \frac{|\{\langle t, u\rangle \in \mathcal{P} | t \succ_S u\}|}{|\mathcal{P}|}$$

Notice that if S is a singleton then the precision and recall of S coincide with the confidence and support of the single rule in S.

Using Definition 5, we can now define precisely the second main problem we consider in this paper, i.e. the construction of a user profile that is concise and sound with respect to a set of user preferences.

Problem 2 (User profile construction). Given a preference database $\mathcal{P}$ and a set of contextual preference rules S, select $\Pi \subseteq S$ that maximizes precision and recall with respect to $\mathcal{P}$ and that is as concise as desired. Π is called the *user profile* associated to $\mathcal{P}$.

Notice that in this problem statement, S can be any set of preference rules. In practice, S will be the set of all interesting minimal contextual preferences rules, as defined in problem 1. It is also important to note that with large datasets, the construction of the smallest set of preference rules that maximizes recall and precision is a hard problem. Indeed, it can be shown that this problem is closely related to the red-blue set cover problem that is NP-complete [6][2]. To cope with this difficulty, we propose in Section 5 a heuristic approach based on the same ideas used by associative classification methods such as CBA [12]. More precisely, given a preference database, a set of interesting preference rules and a parameter k (called *minimal agreement threshold*) that allows to control the size of the user profile returned, we present an iterative algorithm called ProfMiner that maximizes precision and recall.

4 Discovery of Contextual Preference Rules

As indicated in the previous section, we cope with Problem 1 by using pruning criteria stemming from anti-monotone constraints that reduce the search space $\mathcal{CP}$. Before detailing the proposed algorithm, let us recall that a constraint q is anti-monotone iff whenever $i^+ \succ i^- \mid X$ satisfies q, any generalization of $i^+ \succ i^- \mid X$ (i.e., $i^+ \succ i^- \mid Y$ such that $Y \subseteq X$) also satisfies q. Such constraints like the minimal support provide powerful pruning conditions of the search space [13]. Interestingly, the minimality leads to another anti-monotone constraint: whenever a contextual preference rule $i^+ \succ i^- \mid X$ is minimal, all the contextual preference rules $i^+ \succ i^- \mid Y$ satisfying $Y \subseteq X$ are also minimal. As an example, let us consider $r : D \succ E \mid AB$ with $supp(r, \mathcal{P}) = 0.2$ and $conf(r, \mathcal{P}) = 1$. Since r is not a minimal contextual rule (because $supp(r, \mathcal{P}) = supp(D \succ E \mid B, \mathcal{P})$ and $conf(r, \mathcal{P}) = conf(D \succ E \mid B, \mathcal{P})$), we are sure that there is no more minimal rule concluding on $D \succ E$ containing AB in its context. Such pruning technique drastically reduces the search space in a levelwise mining method as Algoritm 1.

Now we detail CONTextual PREFerence rule MINER where the set $\mathcal{C}and_i$ (resp. $\mathcal{MCP}_i$) contains all the candidates (resp. minimal contextual rules) whose context has a cardinality i. Basically, Line 1 initializes the candidates with rules having an empty context. For this purpose, all the pairs of items (i_1, i_2) are considered. While there are candidates of context length i, Line 4 computes all the minimal contextual preference rules of length i satisfying the constraint $supp(r, \mathcal{P}) \geq \sigma$ (test step). Line 5 generates the new candidates of length $i + 1$

[2] Given a finite set of "'red" elements R (here, $\langle u, t\rangle$ such that $\langle t, u\rangle \in \mathcal{P}$), a finite set of "blue" elements B (here, $\mathcal{P}$) and a family of $\mathcal{S} \subseteq 2^{R \cup B}$, the red-blue set cover problem is to find a subfamily $S \subseteq \mathcal{S}$ which covers all blue elements, but which covers the minimum possible number of red elements.

Algorithm 1. ContPrefMiner

```
Input: A preference database $\mathcal{P}$, a minimal support threshold $\sigma$, a minimal confidence
    threshold $\kappa$
Output: All the minimal contextual preference rules with support and confidence
    exceeding $\sigma$ and $\kappa$ respectively.
1: $\mathcal{C}and_0 := \{i_1 \succ i_2 \,|\, \emptyset \in \mathcal{CP}$ such that $(i_1, i_2) \in \mathcal{I} \times \mathcal{I}\}$
2: $i := 0$
3: while $\mathcal{C}and_i \neq \emptyset$ do
4:    $\mathcal{MCP}_i := \{r \in \mathcal{C}and_i$ such that $r$ is minimal and satisfies $supp(r, \mathcal{P}) \geq \sigma\}$
5:    $\mathcal{C}and_{i+1} := \{i_1 \succ i_2 \,|\, X \in \mathcal{CP}$ such that $|X| = i + 1$ and $\forall i \in X, i_1 \succ i_2 \,|\, X \setminus \{i\} \in$
      $\mathcal{MCP}_i\}$
6:    $i := i + 1$
7: od
8: return $\{(r, supp(r, \mathcal{P}), conf(r, \mathcal{P})) \mid r \in \bigcup_{j<i} \mathcal{MCP}_j \wedge conf(r, \mathcal{P}) \geq \kappa\}$
```

(generate step). Finally, Line 8 returns the complete collection of the minimal contextual preference exceeding a minimal confidence threshold (with the corresponding support and confidence).

5 User Profile Construction

Basically, the construction of the user profile iterates two main principles over the contextual preference rules returned by ContPrefMiner until all user preferences in the database are in agreement with at least one preference rule in the profile: (1) select the best contextual preference rule and (2) remove the unnecessary contextual preference rules. Indeed, even if the minimality criterion removes many redundant contextual preference rules, some superfluous contextual preference rules remain among those returned by ContPrefMiner. For instance, in our running example the preference rule $D \succ B \mid A$ (only in agreement with p_1) can be removed from $\mathcal{MCP}_{0.2,0.6}$ (see Table 1) since $D \succ E \,|\, A$ already agrees with p_1 and has a better support (with the same confidence). More generally, a contextual preference rule π provides a substantial value if it agrees with user preferences of $\mathcal{P}$ that are not in agreement with other better preference rules. Note that such a kind of iterative process for building a model is quite classical in the literature [4].

5.1 Ordering Contextual Preference Rules

The main strategy of the algorithm ProfMiner responsible for building user profiles is the ability of selecting the *best* contextual rule to decide which transaction is the preferred one. The following definition introduces a total order on the set of contextual preference rules $\mathcal{MCP}$.

Definition 6 (Best rule order). *The best rule order on $\mathcal{MCP}$, denoted by $>_{best}$, is a total order defined for any contextual preferences π and π' as:*

$$\pi >_{best} \pi' \Leftrightarrow \begin{cases} \mathit{conf}(\pi) > \mathit{conf}(\pi') \text{ or} \\ \mathit{conf}(\pi) = \mathit{conf}(\pi') \text{ and } \mathit{supp}(\pi) > \mathit{supp}(\pi') \text{ or} \\ \mathit{conf}(\pi) = \mathit{conf}(\pi') \text{ and } \mathit{supp}(\pi) = \mathit{supp}(\pi') \\ \qquad \text{and } |\mathit{context}(\pi)| < |\mathit{context}(\pi')| \text{ or} \\ \mathit{conf}(\pi) = \mathit{conf}(\pi') \text{ and } \mathit{supp}(\pi) = \mathit{supp}(\pi') \\ \qquad \text{and } |\mathit{context}(\pi)| = |\mathit{context}(\pi')| \text{ and } \pi <_{\mathcal{CP}} \pi' \end{cases}$$

As the profile should contradict at most a very small number of user preferences (in order to have a high precision), the confidence is the most important criterion. The support criterion naturally comes in second place, followed by the size of the context. The fourth criterion (where $<_{\mathcal{CP}}$ is an arbitrary total order) is only used to definitely decide between two indistinguishable rules.

Table 1 (the left part) illustrates the best rule order $>_{best}$ over the minimal contextual preference rules with $\sigma = 0.2$ and $\kappa = 0.6$ on our running example. Note that the arbitrary order $<_{\mathcal{CP}}$ justifies to arrange $A \succ C | D$ before $A \succ D | C$ and $B \succ C | D$ as well as $D \succ B | A$ before $D \succ E | AC$.

Table 1. Rules of $\mathcal{MCP}_{0.2,0.6}$ ordered according $>_{best}$ and profile construction ($k = 1$)

$\mathcal{MCP}_{0.2,0.6}$				Profile construction			
Cont. pref.	*supp*	*conf*	Agreement	step 1	step 2	step 3	step 4
$D \succ E \| A$	0.4	1	p_1,p_2	✔			
$D \succ C \| \emptyset$	0.2	1	p_2	✗			
$A \succ C \| D$	0.2	1	p_3		✔		
$A \succ D \| C$	0.2	1	p_4			✔	
$B \succ C \| D$	0.2	1	p_3		✗		
$D \succ B \| A$	0.2	1	p_1	✗			
$D \succ E \| B$	0.2	1	p_2	✗			
$D \succ E \| AC$	0.2	1	p_1	✗			
$D \succ B \| \emptyset$	0.4	2/3	p_1,p_5				✔
$D \succ E \| \emptyset$	0.4	2/3	p_1,p_2	✗			

5.2 The Algorithm ProfMiner

Given a preference database $\mathcal{P}$, a set of contextual preference rules S, a minimal agreement threshold k, ProfMiner returns a user profile Π by selecting suitable contextual preference rules from S (see Algorithm 2). Note that the agreement threshold k enables us to adjust the size of the user profile as indicated in Problem 2. The greater the minimal agreement k, the smaller the profile.

After initializing the profile (Line 1), the main loop (Line 2-7) selects the best contextual preference rule according to $>_{best}$ (Line 3) and adds it to the profile (Line 4) until that S becomes empty (Line 2). This condition is ensured by the reduction of $\mathcal{P}$ (Line 5) and the reduction of S (Line 6). Indeed, a contextual preference rule π is unnecessary with respect to the profile in progress whenever

Algorithm 2. ProfMiner

```
Input: A preference database $\mathcal{P}$, a set of preference rules $S$, a minimal agreement $k$
Output: A user profile $\Pi$
1: $\Pi := \emptyset$
2: while $S \neq \emptyset$ do
3:     $\pi_{best} = \max_{>_{best}} S$
4:     $\Pi := \Pi \cup \{\pi_{best}\}$
5:     $\mathcal{P} := \{\langle t, u\rangle \in \mathcal{P} | t \not\succ_{\pi_{best}} u\}$
6:     $S := \{\pi \in S | supp(\pi, \mathcal{P}) \geq k/|\mathcal{P}|\}$
7: od
8: return $\Pi$
```

π does not agree with at least k remaining user preferences (i.e., not still in agreement with other preference rules of the profile).

Table 1 (the right part) illustrates ProfMiner on our running example (see Figure 1) with $S = \mathcal{MCP}_{0.2,0.6}$ and $k = 1$. At the first iteration, Line 3 selects $D \succ E \,|\, A$ (symbol ✔) as the best rule according to $>_{best}$ (see Table 1). Line 5 removes the user preferences p_1 and p_2 and then, Line 6 removes 5 contextual preference rules from S (symbol ✗). Note that $D \succ B \,|\, \emptyset$ is preserved because it also covers p_5. The second iteration adds $A \succ C \,|\, D$ to the profile because it is the best remaining contextual preference rule. This process stops at the end of the 4th iteration because S is empty (see Line 2 of ProfMiner). So, the final profile is $\{D \succ E | A, A \succ C | D, A \succ D | C, D \succ B | \emptyset\}$.

6 Experimental Results

This experimental study aims at evaluating the conciseness and the soundness of user profiles mined by our approach. Indeed, a comprehensive study of the effectiveness of our approach has been conducted on real world datasets based on the APMD-Workbench [14] built from MovieLens (`www.movielens.org`) and IMDB (`www.imdb.com`). The used datasets and detailed data preparation process are available on the CPrefMiner project repository (`www.lsi.ufu.br/cprefminer/`). All the tests were performed on a 3 GHz Intel processor with Windows XP operating system and 1 GB of RAM memory. The overall process of preference rule mining and user profile construction is performed in at most 113 seconds, for the largest preference database P30000 described below.

Basically, the datasets consist in 6 user preference databases about movies, one database per user. In each database, a user preference about movies is represented by a pair of movie records $\langle m_1, m_2\rangle$ meaning that "the user prefers the movie m_1 to the movie m_2". A movie record is based on a set of attributes such as Genre, Director, Years, Actor, etc. Genre, Director and Actor are multi-valued attributes. Hence, to apply our approach relying on contextual preference rules, we shall itemize each distinct attribute value so that each movie record becomes a transaction corresponding to our data model. Due to the space limitation, we

Database	Items ($\mathcal{I}$)	Trans. ($\mathcal{D}$)
P301	125	32
P3000	342	99
P30000	857	309

Fig. 2. Real world preference databases over movies

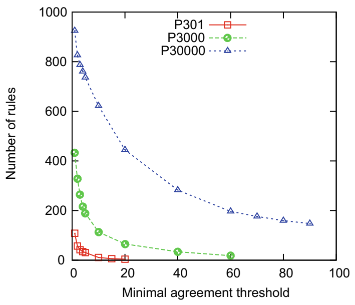


Fig. 3. Number of preference rules per profile w.r.t various k values

only present the experimental results of 3 preference databases named `P301`, `P3000`, and `P30000` as shown by Figure 2. The results on the 3 other preference databases are very similar. Each database is named by its number of user preferences, e.g., the database `P301` contains 301 user preferences corresponding to a set $\mathcal{D}$ of 32 distinct movie records described by a set $\mathcal{I}$ of 125 distinct items.

For each preference database, the user profile mining and preference prediction have been performed using a 10-fold cross-validation method, and the metric values (e.g., precision and recall) on the different iterations are averaged to yield an overall one. The minimal contextual preference rules are mined using ContPrefMiner with $\sigma = 0.001$ and $\kappa = 0.5$. Note that other minimal thresholds have been tested (not reported here due to space limitation) showing that the increase of σ systematically damages the quality of user profiles while the increase of κ has a lower impact. The user profile construction was done with ProfMiner by varying the minimal agreement threshold k.

Table 2. Top-10 preference rules discovered from the database P3000 ($k = 1$)

Contextual preference rule	Support	Confidence
1. LAN:German $\succ$ LAN:English | $\emptyset$	0.017	1.00
2. GEN:Fantasy $\succ$ GEN:War | GEN:Drama	0.015	1.00
3. GEN:Crime $\succ$ GEN:Adventure | GEN:Action	0.012	1.00
4. GEN:Crime $\succ$ GEN:Horror | GEN:Sci-Fi	0.012	1.00
5. GEN:Romance $\succ$ GEN:War | GEN:Drama	0.011	1.00
6. GEN:Crime $\succ$ GEN:Adventure | GEN:Sci-Fi	0.010	1.00
7. GEN:Crime $\succ$ GEN:Drama | $\emptyset$	0.010	1.00
8. GEN:Fantasy $\succ$ GEN:Action | GEN:Drama	0.009	1.00
9. LAN:German $\succ$ LAN:Vietnamese | GEN:War	0.009	1.00
10. GEN:Sci-Fi $\succ$ GEN:Western | GEN:Action	0.009	1.00

We start by analyzing the conciseness of the user profile according to the minimal agreement threshold. Figure 3 plots the number of preference rules when the minimal agreement threshold k varies from 1 to 90. Even with $k = 1$, the number of preference rules contained in the user profile is drastically reduced compared to the inital number of contextual preference rules: from 5319.4 to

108.7 for P301; from 4833.9 to 432.9 for P3000; and from 4913.3 to 925 for P30000. Moreover, Figure 3 shows that the size of the user profile rapidly decreases with k and then, the user profile can be as concise as desired by the user.

The preference prediction was performed using the orders induced by the profile. Figure 4 estimates the effectiveness of the user profiles according to their size. For facilitating comparisons between the different preference databases, the size of a user profile $|\Pi|$ is normalized by means of the *profile reduction rate* defined by $(|\Pi_{k=1}| - |\Pi|)/|\Pi_{k=1}|$ where $|\Pi_{k=1}|$ is the cardinality of the user profile obtained from $k = 1$.

Figure 4 reports the precision, the recall and F-measure (i.e., $2 \times precision \times recall/(precision + recall)$) for the profile when the profile reduction rate varies. The first important observation is that the predictive quality of the mined profiles can be very high. More precisely, the precision always remains very high, while the recall deeply depends on the size of the user profile.

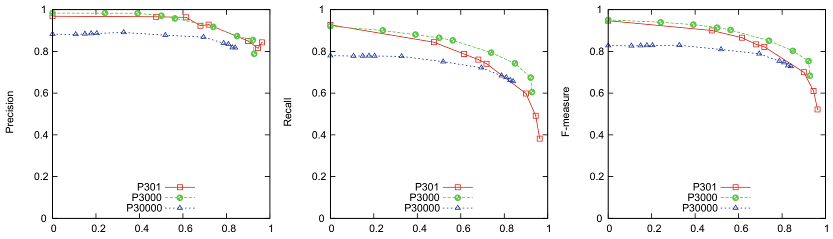


Fig. 4. Predictive quality of constructed profiles

In brief, this set of experiments demonstrates that the conciseness of user profiles is controlled by the minimal agreement threshold and that even with strong reduction, the soundness of the profile remains at an acceptable level. But what does the mined profiles look like? Table 2 lists the top-10 preference rules of a user profile discovered from the database P3000 with $k = 1$. It demonstrates that a mined profile is easy readable. For example, rules 1 and 9 means that the user prefers german movies than english or vietnamese movies. We can also see that the user especially enjoys crime movies (see rules 3, 4, 6 and 7). Between two drama movies, this profile finally shows that the user prefers fantasy movies than war movies (see rule 2) or action movies (see rule 8), that he/she prefers romance movies than war movies (see rule 5).

7 Conclusion and Future Work

In this paper we proposed the method ProfMiner for mining user profiles from preference databases. A set of experiments on a real-world database of user preferences about movies showed the efficiency of the method. More interestingly, our approach is the first one to build readable user profile based on the notion of contextual preference rules.

The overall aim of a profile is to order a set of transactions. Thus, it would be expected that the preference relation associated to the user profile be a strict partial order over transactions. However, this is not the case since the induced order is not transitive in general. Presently, we are developing two other methodologies for extracting a strict partial order from a given set of pairs of transactions, one based on Bayesian Network classifiers and other based on a voting system.

As future work ,we finally plan to compare the predictive quality of our method with well-known ranking methods as RankNet, Rank SVM, Ada Rank and RankBoost [11,8,5,16], knowing that existing prototypes that implement these methods have to be adapted (in order to take directly as input pairwise preferences, and not only quantitative preferences).

Acknowledgments. We thank the French Embassy of Dakar, the Brazilian Research Agencies CNPq, CAPES (SticAmSud Project 016/09) and FAPEMIG for supporting this work.

References

1. Agrawal, R., Rantzau, R., Terzi, E.: Context-sensitive ranking. In: SIGMOD Conference, pp. 383–394. ACM (2006)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
3. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. 21, 135–191 (2004)
4. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: ICDM, pp. 63–72. IEEE Computer Society (2007)
5. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.N.: Learning to rank using gradient descent. In: ICML, vol. 119, pp. 89–96. ACM (2005)
6. Carr, R.D., Doddi, S., Konjevod, G., Marathe, M.V.: On the red-blue set cover problem. In: SODA, pp. 345–353 (2000)
7. Crammer, K., Singer, Y.: Pranking with ranking. In: NIPS, pp. 641–647. MIT Press (2001)
8. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res. 4, 933–969 (2003)
9. Holland, S., Ester, M., Kießling, W.: Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 204–216. Springer, Heidelberg (2003)
10. Jiang, B., Pei, J., Lin, X., Cheung, D.W., Han, J.: Mining preferences from superior and inferior examples. In: KDD, pp. 390–398. ACM (2008)
11. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD, pp. 133–142. ACM (2002)
12. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: KDD, pp. 80–86 (1998)
13. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Min. Knowl. Discov. 1(3), 241–258 (1997)

14. Peralta, V., Kostadinov, D., Bouzeghoub, M.: APMD-workbench: A benchmark for query personalization. In: Proceedings of the CIRSE Workshop (2009)
15. Song, R., Guo, Q., Zhang, R., Xin, G., Wen, J.-R., Yu, Y., Hon, H.-W.: Select-the-best-ones: A new way to judge relative relevance. Information Processing and Management 47(1), 37–52 (2011)
16. Xu, J., Li, H.: AdaRank: a boosting algorithm for information retrieval. In: SIGIR, pp. 391–398. ACM (2007)

Polarities, Axiallities and Marketability of Items

Dan A. Simovici[1], Paul Fomenky[1], and Werner Kunz[2]

[1] University of Massachusetts Boston,
Department of Computer Science, Boston, MA 02125, USA
[2] University of Massachusetts Boston,
Department of Marketing, College of Management, Boston, MA 02125, USA

Abstract. We apply polarities, axiallities and the notion of entropy to the task of identifying marketable items and the customers that should be approached in a marketing campaign. An algorithm that computes the criteria for identifying marketable items and the corresponding experimental work is also included.

Keywords: polarity, axiallity, closure operator, entropy.

1 Introduction

Recommender systems (RS) aim to help users deal with the immensity of offers in electronic commerce by customizing the most adequate offerings for their specific needs. After almost two decades, this discipline is quite established following the initial publications [11,10] and [13]. The general framework of RSs involves a bipartite graph whose set of vertices is partitioned into customers and items (see Figure 1). Edges of the form (c, t), where c is a customer and t is an item are marked by numerical ratings $r(c, t)$. Recommender systems that use ratings generate item recommendations for customers or identify sets of customers suitable for sets of items and fit into one of the following broad three approaches: content-based, collaborative, or hybrid. An excellent survey of developments in recommender systems can be found in [3].

Further, more sophisticated types of RSs extend the user/item paradigm to take into account temporal and other contextual characteristics of customers and items (see [2]). All existing approaches have in common the use of additional data (rating, survey information) and many are based on hard-to-implement complex inference statistical approaches [1,7,9]. Furthermore, most of then aim to obtain optimal recommendations for the consumer and ignore the business perspective.

Formal concept analysis which makes use of the notion of polarity was applied in the study of RSs in [6] in the quest of simplifying the task of finding similar users or similar items without loss of accuracy or coverage. The purpose of this paper is distinct from the main topics of the area recommender systems. We investigate possibilities to identify items that should be the object of marketing campaigns and we intend to extend this approach to sets of items that can be co-marketed. Thus, we approach RSs from the position of the seller rather than from the prospective of the users. Instead of providing recommendations to the users we propose to suggest items or sets of items to retailers which could stimulate sales.

We use entropy (and some of its generalizations) as measures of scattering of partition blocks for partitions of finite sets. Namely, if $\pi = \{B_1, \ldots, B_n\}$ is a partition of a finite set S, its generalized b-entropy (see [5,12]) is defined as

$$\mathcal{H}_b = \frac{1}{1 - 2^{1-b}} \left(1 - \sum_{i=1}^{n} \left(\frac{|B_i|}{|S|} \right)^b \right),$$

where $b > 1$.

Two special cases of generalized entropy are particularly interesting. For $b = 2$ we have the Gini index of π given by

$$\mathsf{gini}(\pi) = 2 \left(1 - \sum_{i=1}^{n} \left(\frac{|B_i|}{|S|} \right)^2 \right).$$

The largest value of $\mathsf{gini}(\pi)$ is obtained when all blocks have equal size (and this, the elements of S are uniformly scattered in the blocks of π); in this case we have

$$\mathsf{gini}(\pi) = 2 \left(1 - \frac{1}{n} \right).$$

The lowest value, $\mathsf{gini}(\pi) = 0$, is obtained when π consists of one block.

The other interesting case of generalized entropy is obtained when b tends to 1. In this case

$$\lim_{b \to 1} \mathcal{H}_b(\pi) = - \sum_{i=1}^{n} \frac{|B_i|}{|S|} \log_2 \frac{|B_i|}{|S|},$$

which recaptures the well-known Shannon entropy.

The article is structured as follows. Section 2 introduces the notions of polarity and axiallity in the context of recommender systems. In Section 3 we define marketable item sets and formulate an algorithm for identifying these sets. Experimental work is described in Section 4.

2 Polarities, Axiallities and Recommender Systems

Let C and T be two finite sets, referred to as the *set of customers* and the *set of items*, respectively and let $\rho \subseteq C \times T$. Following the terminology of concept lattices [8] we shall refer to the triple $\mathcal{C} = (C, T, \rho)$ as a *recommendation context.* The fact that $(c, t) \in \rho$ means that the customer c has purchased the item t. Sets of customers will be denoted by letters from the beginning of the alphabet $D, E, K, \ldots$; sets of items will be denoted by letters from the end of the alphabet $U, V, W, \ldots$.

For a set S, the set of subsets of S is denoted by $\mathcal{P}(S)$. If S, T are sets, a function $f : \mathcal{P}(S) \longrightarrow \mathcal{P}(T)$ is *monotonic* if for every $X, Y \in \mathcal{P}(S)$, $X \subseteq Y$ implies $f(X) \subseteq f(Y)$; f is *anti-monotonic* if $X \subseteq Y$ implies $f(X) \supseteq f(Y)$.

Consider the mappings $\phi_\rho : \mathcal{P}(C) \longrightarrow \mathcal{P}(T)$, and $\psi_\rho : \mathcal{P}(T) \longrightarrow \mathcal{P}(C)$ given by

$$\begin{aligned}\phi_\rho(D) &= \{t \in T \mid (\forall d \in D)(d,t) \in \rho\},\\ \psi_\rho(U) &= \{c \in C \mid (\forall u \in U)(c,u) \in \rho\}\end{aligned}$$

for $D \in \mathcal{P}(C)$ and $U \in \mathcal{P}(T)$. In other words, $\phi_\rho(D)$ consists of items that were bought by all customers in D and $\psi_\rho(U)$ consists of customers who bought all items of U. As shown in [4] (Chapter V), the mappings ϕ_ρ and ψ_ρ are anti-monotonic. The pair $\mathsf{Pol}_\rho = (\phi_\rho, \psi_\rho)$ is the *polarity* of ρ,

Let $\overline{\rho} = (C \times T) - \rho$ be the relation that consists of all pairs (c,t) that are not in ρ. Another pair of functions defined by ρ is $(\alpha_\rho, \beta_\rho)$, where $\alpha_\rho : \mathcal{P}(C) \longrightarrow \mathcal{P}(T)$ and $\beta_\rho : \mathcal{P}(T) \longrightarrow \mathcal{P}(C)$ are given by

$$\alpha_\rho(D) = \phi_{\overline{\rho}}(\bar{D}) \text{ and } \beta_\rho(U) = \overline{\psi_{\overline{\rho}}(U)}$$

for $D \in \mathcal{P}(C)$ and $U \in \mathcal{P}(T)$, where $\bar{D} = C - D$.

By applying the definition of $\phi_{\overline{\rho}}$ we have

$$\begin{aligned}\alpha_\rho(D) &= \phi_{\overline{\rho}}(\bar{D}) = \phi_{\overline{\rho}}(C - D)\\ &= \{t \in T \mid (\forall d \in C - D)(d,t) \notin \rho\}.\end{aligned}$$

In other words, $\alpha_\rho(D)$ consists of those items $t \in T$ which were not bought by any customer who does not belong to D.

Similarly, by applying the definition of $\psi_{\overline{\rho}}$ we have

$$\begin{aligned}\beta_\rho(U) &= \overline{\psi_{\overline{\rho}}(U)} = C - \psi_{\overline{\rho}}(U)\\ &= C - \{c \in C \mid (\forall u \in U)(c,u) \notin \rho\}\\ &= \{c \in C \mid (\exists u \in U)(c,u) \in \rho\},\end{aligned}$$

which shows that $\beta_\rho(U)$ consists of customers who bought some items in U.

It is immediate that the functions α_ρ, β_ρ are monotonic. In other words, we have

$$\begin{aligned}D_1 \subseteq D_2 &\Rightarrow \alpha_\rho(D_1) \subseteq \alpha_\rho(D_2),\\ U_1 \subseteq U_2 &\Rightarrow \beta_\rho(U_1) \subseteq \beta_\rho(U_2),\end{aligned}$$

for $D_1, D_2 \in \mathcal{P}(C)$ and $U_1, U_2 \in \mathcal{P}(T)$.

The pair $\mathsf{Axl}_\rho = (\alpha_\rho, \beta_\rho)$ is the *axiallity* of ρ.

For the polarity mappings we have

$$D \subseteq \psi_\rho(\phi_\rho(D)) \text{ and } U \subseteq \phi_\rho(\psi_\rho(U))$$

for any set of customers $D \in \mathcal{P}(C)$ and every set of items $U \in \mathcal{P}(T)$. The mappings $\psi_\rho\phi_\rho$ and $\phi_\rho\psi_\rho$ are closure operators on $\mathcal{P}(C)$ and $\mathcal{P}(T)$, respectively.

For the axiallity mappings we have

$$\begin{aligned}\beta_\rho(\alpha_\rho(D)) &= \overline{\psi_{\overline{\rho}}(\alpha_\rho(D))}\\ &= \overline{\psi_{\overline{\rho}}(\phi_{\overline{\rho}}(\overline{D}))} \subseteq D,\end{aligned}$$

and

$$\begin{aligned}\alpha_\rho(\beta_\rho(U)) &= \alpha_\rho(\overline{\psi_{\overline{\rho}}(U)}) \\ &= \phi_{\overline{\rho}}(\psi_{\overline{\rho}}(U)) \supseteq U,\end{aligned}$$

which allows us to conclude that $\beta_\rho\alpha_\rho$ is an interior operator on sets of customers and $\alpha_\rho\beta_\rho$ is a closure operator on sets of items.

Let ITEMS_c be the set of items acquired by customer c and let CUST_t be the set of customers who bought item t. Also, define the sets P_c and R_t by

$$P_c = \{c\} \times \mathsf{ITEMS}_c \text{ and } R_t = \mathsf{CUST}_t \times \{t\}$$

for each customer c and item t (see Figure 1).

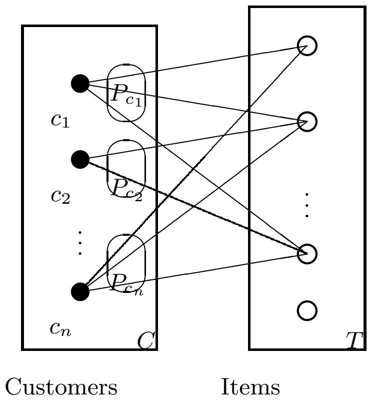


Fig. 1. Partitions of the set of purchases generated by customers

Both collections $\{P_c \mid c \in C\}$ and $\{R_t \mid t \in T\}$ are partitions of the set of purchases ρ. Also, observe that $P_c \cap R_t = \{(c,t)\}$ for $c \in C$ and $t \in T$.

For a set of customers $D \subseteq C$, the set of purchases is $\mathsf{pur}(D) = \bigcup_{c \in D} P_c$ and the collection of sets $\{P_c \mid c \in D, P_c \neq \emptyset\}$ is a partition of $\mathsf{pur}(D)$.

The entropy (or the Gini index) of the partition $\pi_D = \{P_c \mid c \in D, P_c \neq \emptyset\}$ captures the diversity of purchasing patterns for the customers in D. Note that

$$\begin{aligned}\mathcal{H}_1\left(\{P_c \mid c \in D, P_c \neq \emptyset\}\right) &= -\sum_{c \in C} \frac{|P_c|}{|\mathsf{pur}D|} \log_2 \frac{|P_c|}{|\mathsf{pur}D|} \\ &= \log_2 |\mathsf{pur}(D)| - \frac{1}{|\mathsf{pur}(D)|} \sum_{c \in D} |P_c| \log_2 |P_c|.\end{aligned}$$

We use the *specific entropy* $h_1(D)$ of a set of customers D defined as the ratio between the entropy of the purchases of customers in D and the size of D

$$h_1(D) = \frac{\mathcal{H}_1\left(\{P_c \mid c \in D, P_c \neq \emptyset\}\right)}{|D|}.$$

The specific entropy is intended to compensate the growth of the entropy of the partition $\{P_c \mid c \in D\}$ due to an increase in the size of the customer population D, and seems to be a better indicator of the diversity of the purchasing patterns of the population in D than the entropy of the partition π_D.

3 Marketable Items

We examine criteria for choosing items that should be the object of a marketing campaign. The reason for starting a marketing campaign involving an item t is that the set of users who purchased t is non-empty but small; in other words, $|\psi_\rho(\{t\})|$ does not exceed a threshold θ. Using the notions of polarity and axiallity that can be defined starting from the purchasing relation defined as a binary relation on the sets of customers and items we focus on items that satisfy several conditions:

1. the closure $\phi_\rho(\psi_\rho(\{t\})$, which consists of items that were bought by customers who bought t must be sufficiently large;
2. since β_ρ is a monotonic mapping, the set of customers who bought some item in the previously mentioned set of items, $\beta_\rho(\phi_\rho(\psi_\rho(\{t\}))$ will, in turn be large, and
3. the purchasing patterns of these customers must be sufficiently diverse, to ensure a reasonable chance that they will decide to buy t.

The target of the marketing campaign is the set of customers $\beta_\rho(\phi_\rho(\psi_\rho(\{t\})) - \psi_\rho(\{t\})$. These criteria are summarized in Algorithm 1.

The SQL procedure that implements the algorithm and includes the computation of the entropy of purchases is given next.

```
create procedure market1(item1 integer)
 begin
      # cleaning up tables for intermediate results
      call cleanup();

      # custforitem(userid) contains customers who bought item1

      insert into custforitem
          select userid from pur where item = item1;


      # items bought by every customer in custforitem
      # are stored in itemsbyallcust(item)

      insert into itemsbyallcust
          select distinct item from pur r where
              not exists(select * from custforitem where
                      not exists(select *
Data: A table pur of purchases, a minimum and a maximum number of
      purchases minpurand maxpur, respectively
Result: a set of customers targeted for the marketing campaign
Place in table selitems items from pur bought by at least minpur customers
      but not more that maxpurcustomers ;
foreach item t in selitems do
    retrieve in table custforitem customers who bought $t$,
        $\psi_\rho(\{t\}) = \beta_\rho(\{t\}) \rightarrow$ custforitem;
    retrieve in table itemsbyallcust items bought by every
        customer in custforitem,
        $\phi_\rho(\psi_\rho(\{t\}) \rightarrow$ itemsbyallcust;
    retrieve in table custwhoboughtsome customers who
        bought some item in itemsbyallcust,
        $\beta_\rho(\phi_\rho(\psi_\rho(\{t\})) \rightarrow$ custwhoboughtsome;
    retrieve in targetitem customers targeted for marketing
        $\beta_\rho(\phi_\rho(\psi_\rho(\{t\})) - \psi_\rho(\{t\}) \rightarrow$ targetitem;
    compute the entropy (or the Gini index) for the purchases
        made by customers targeted for marketing;
end
```

Algorithm 1. Algorithm for computing the target set of a marketing campaign

```
                    from pur where
                    userid = custforitem.userid
                    and item = r.item));

# custwhoboughtsome(userid) contains customers who bought
# some item in itemsbyallcust

insert into custwhoboughtsome
    select distinct userid from pur
    where item in (select item from itemsbyallcust);

# targetitem(userid) contains customers targeted for marketing

insert into targetitem
   select userid from custwhoboughtsome where
      not exists(select * from custforitem
                     where userid = custwhoboughtsome.userid);

select 'Customers targeted for marketing';

select userid from targetitem;

# calculation of entropy for the customers in targetitem
# follows
```

```
    insert into purcust(userid, noitem)
        select userid, numitemcust(userid) from targetitem;

    insert into custsq
        select userid,noitem*log(2,noitem) from purcust;

    insert into results
        select sum(c),(select sum(noitem) from purcust)
            from custsq;


   select log(2,ct) - (1/ct) *s from results;
end
```

4 Experimental Study

We used the MovieLens data set that contains 100,000 anonymous ratings of 1,682 movies made by 943 customers (referred to as users in the documentation of the data set). This data set was obtained from the University of Minnesota GroupLens Research `www.movielens.org` and was processed using `mySQL`. The main characteristics of the attributes of this data set are specified below.

– UserIDs are integers;
– MovieIDs range between 1 and 1,682;
– Ratings are made on a 5-star scale (whole-star ratings only);
– Timestamp is represented in seconds;
– Each user has at least 20 ratings.

The relation pur was extracted by projecting the data set on the attributes UserId and MovieId (referred to as item). The customer sets for the analyzed items consisted between 5 and 10 individuals, as shown in the second column of Table 1. D_t is the set of customers targeted for the marketing campaign for t,

$$D_t = \beta_\rho(\phi_\rho(\psi_\rho(\{t\})) - \psi_\rho(\{t\}).$$

The customer population targeted for these campaigns varied between 116 and 880 individuals, as shown in Table 1. It is not a surprise that the size of the targeted customer population D_t has a strong positive correlation with the entropy of the partition of purchases of this set of customers. For example, for the sample of movies we experimented the correlation coefficient is 0.91.

As it can be seen from Figure 2, the larger the targeted population of customers, the larger the entropy is and therefore, we have a better chance that some of these customers will buy the item t, which is the focus of the marketing campaign. This explains the strong positive correlation between the size of the targeted population and the entropy.

Table 1. Size and Entropy for several items viewed by five to ten users

item t	cust_for_item $\|\psi_\rho(t)\|$	cust_targeted $\|D\|$	entropy $\mathcal{H}_1(\pi_{D_t})$	spec_entropy $h(D_t))$
34	7	880	9.28	19.50
37	8	731	9.09	22.45
74	7	661	8.98	24.11
75	5	714	9.06	22.58
104	5	684	9	22.53
113	9	116	6.44	16.40
247	5	546	8.68	22.02
296	6	670	8.87	18.12
314	5	798	9.11	18.18
390	10	600	8.82	22.57
437	5	556	8.96	57.37
438	6	655	8.96	23.66
439	5	556	8.96	57.37
446	9	447	8.51	29.93

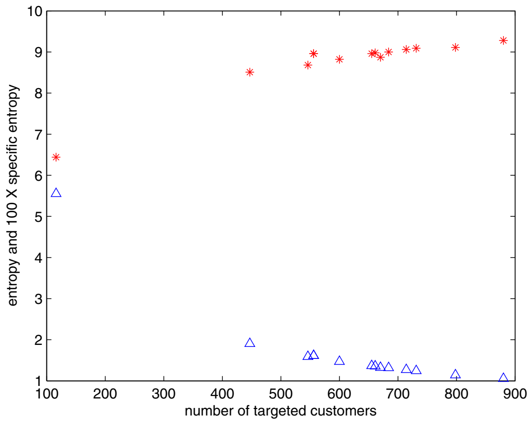


Fig. 2. Entropy ($*$) and Specific Entropy per Customer ($\triangle$) vs. Size of Set of Targeted Customers

For a targeted population D_t the maximum entropy of the partition of purchases is $\log_2 |D_t|$. The specific entropy defined as

$$h(D_t) = \frac{\log_2 |D_t|}{\log_2 |D_t| - \mathcal{H}_1(\pi_{D_t})}$$

is a better indicator of the diversity of purchases because it takes into account the relative size of the customer population targeted. Thus, the best targets for a marketing campaign are the items 437 and 439 for which $h_1(D_t)$ has a relatively high value (57.37).

5 Further Work

In this paper we introduced an approach that can be used by companies to market item sets to customer groups that possess a very likely high preference for these products. The algorithm is easy to implement and use in a daily managerial life. As input data, solely the past purchase data of all customers are needed, which is usually available in a company today. Our proposed approach can be used towards two directions. It can be used to segment the customers according to their preference fit for an individual item (set) as well as to build groups of items and rank them according their productivity for the existing customer base. In future research more simulation studies are needed to show that the proposed approach is not only easier to implement but is also equally (or better) suited to forecast customer product fit as well as optimize profit for the implementing company.

The productivity $\mathsf{prod}(t)$ of a marketing campaign for an item t can be measured by the ratio between the size of the target population of customers and the size of the set of customers who bought t:

$$\mathsf{prod}(t) = \frac{|\beta_\rho(\psi_\rho(\phi_\rho(\{t\})))|}{\phi_\rho(\{t\})}$$

This function can be extended to a set of items U by defining

$$\mathsf{prod}(U) = \frac{|\beta_\rho(\psi_\rho(\phi_\rho(U)))|}{\phi_\rho(U)}$$

for $U \subseteq T$. Note that this function is monotonic with respect to U; in other words, $U_1 \subseteq U_2$ implies $\mathsf{prod}(U_1) \leq \mathsf{prod}(U_2)$. We intend to explore criteria for marketing jointly sets of items using both productivity and the entropy of the set of customer purchases.

Incorporating the effect of the ratings of items by users will also be investigated in the future.

Acknowledgment. D. A. Simovici was supported by a S&T grant from the President's Office of the University of Massachusetts.

References

1. Ansari, A., Essegaier, S., Kohli, R.: Internet Recommendation Systems. Journal of Marketing Research 37, 363–377 (2000)
2. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems 23, 103–145 (2005)
3. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005)

4. Birkhoff, G.: Lattice Theory, 3rd (with corrections) edn. American Mathematical Society, Providence (1995)
5. Daróczy, Z.: Generalized information functions. Information and Control 16, 36–51 (1970)
6. du Boucher-Ryan, P., Bridge, D.: Collaborative recommending using formal concept analysis. Knowledge-Based Systems 19, 309–315 (2006)
7. Decker, R., Trusov, M.: Estimating aggregate consumer preferences from online product reviews. International Journal of Research Marketing 27, 293–307 (2010)
8. Denecke, K., Erné, M., Wismuth, S.L.: Galois Connection and Applications. Kluwer Academic Publishers, Dordrecht (2004)
9. Fader, P.S., Hardie, B.G.S.: Modeling consumer choice among SKUs. Journal of Marketing Research 33, 442–452 (1996)
10. Hill, W., Stead, L., Rosenstein, M., Furnas, G.: Recommending and evaluating choices in a virtual community of use. In: Proceedings of the Conference on Human Factors in Computer Systems, pp. 194–201. ACM, New York (1995)
11. Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of Computer Supported Cooperative Work, pp. 175–186. ACM, New York (1994)
12. Simovici, D.A., Djeraba, C.: Mathematical Tools for Data Mining. Springer, New York (2008)
13. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating 'word of mouth'. In: Proceedings of the Conference on Human Factors in Computer Systems, pp. 210–217. ACM, New York (1995)

RssE-Miner: A New Approach for Efficient Events Mining from Social Media RSS Feeds

Nabila Dhahri[1], Chiraz Trabelsi[1], and Sadok Ben Yahia[1,2]

[1] Faculty of Sciences of Tunis, University Tunis El-Manar, 2092 Tunis, Tunisia
[2] Department of Computer Science, TELECOM SudParis, UMR 5157 CNRS Samovar, 91011 Evry Cedex, France
dhahri.nabila@gmail.com,
{chiraz.trabelsi,sadok.benyahia}@fst.rnu.tn

Abstract. Most of the new social media sites such as Twitter and Flickr are using RSS Feeds for sharing a wide variety of current and future real-world events. Indeed, RSS Feeds is considered as a powerful real-time means for real-world events sharing within the social Web. Thus, by identifying these events and their associated user-contributed social media resources, we can greatly improve event browsing and searching. However, a thriving challenge of events mining processes is owed to an efficient as well as a timely identification of events. In this paper, we are mainly dealing with event mining from heterogenous social media RSS Feeds. Therefore, we introduce a new approach, called RssE-Miner, in order to get out these events. The main thrust of the introduced approach stands in presenting a better trade-off between event mining accuracy and swiftness. Specifically, we adopted the probabilistic Naive Bayesian model within the exploitation of the rich context associated with social media Rss Feeds contents, including user-provided annotations (e.g., title, tags) and the automatically generated information (e.g., time) for efficiently mining future events. Carried out experiments over two real-world datasets emphasize the relevance of our proposal.

Keywords: Event Identification, Social Media, Real Time, Really Simple Syndication, News Mining, Unstructured Data.

1 Context and Motivations

The task of event identification embraces two well-known approaches: linguistic approach and statistical one. The former constitute a hindrance since linguistic expert uses rules for manually defining event patterns [11]. In this paper, we do some explorations on the latter direction.

In this context, the topic detection and tracking (TDT) event detection task [16,14], which is similar to our event identification task, was studied on a continuous stream of news documents with the aim to identify news events and organize them. Most techniques mainly focused on formal text stream data. However, social media RSS Feeds, which are the focus of this paper, are substantially different from formal text stream data as the text is so noisy that its handling becomes more difficult [12]. In this respect, searching or browsing through the web becomes difficult and inaccurate.

Recently, RSS Feeds have become ubiquitous with the evolution of the web. Especially, social RSS Feeds are defined as a collection of informal data that arrives over time and each RSS Feed item is associated with some social attributes such as title, description, tags. These features contain information about real world events that are manually added by users. The key challenges that we address are: the identification of events and their associated news items over social media sites(e.g.,Flickr, Youtube, and Facebook).

It can happen that you think to attend the carnival in Rio de Janeiro. Hence, you make your way to the computer to seek information about Rio Carnival with the goal to purchase a ticket as soon as possible. Unfortunately, there are too many photographs dealing with this event. You sift through many different photographs interleaved with photographs of the same event happened at the last year. At the end, you are lost among these results and there is a possibility that you miss the deadline for buying tickets. The above scenario is a likely and perhaps frequent occurrence on the Web today.

Many research studies have then attempted to identify events from social media sites [4,8]. Most of them opt for the popular Support Vector Machine as a machine learning technique. This classifier is memory-intensive and time consuming. That's why, we think about Naive Bayes as an accurate and fast classifier that makes real-time use possible and does well if you have fairly little data [9] as the case in social media. Thus, social media documents contain little textual narrative, usually in the form of short description, title, or tags.

We apply an appropriate algorithm, namely Naive Bayes, for the event identification task in social media RSS Feeds trading off runtime performance and classification accuracy. We apply our approach to two real-world datasets derived from Flickr. We refer to these datasets as Upcoming and Last.fm, as the labels have been extracted from these sites.

This paper is structured as follows. First, Section 2 discusses the related work. Section 3 elaborates in the proposed approach. Then, in Section 4 the approach is evaluated. Finally, Section 5 concludes the paper and provides future research directions.

2 Related Work

We describe related work in four areas namely event identification or detection, RSS Feeds studies, social media analysis, and Naive Bayes applications.

The event detection task [16,14] was studied on a continuous stream of news documents with the aim to identify news events and organize them. This is one of the important tasks considered by the topic detection and tracking (TDT) [2]. However, our work is distinguished by the fact that we are interested in event identification in social media RSS Feeds where the text is so noisy that its handling becomes more difficult [12]. In our current endeavors, we aim to have a fully automated application for processing social media events, fetched from

Really Simple Syndication(RSS) Feeds in such a way that the essence of the social media resources is extracted and captured in events that are represented in a machine-understandable way .

There are several studies that have focused on the RSS Feeds processing: We can talk about SPEED (Semantics-Based Pipeline for Economic Event Detection) [11] which is a framework that aims at extracting financial events from news articles(announced through RSS Feeds) and annotated these with meta-data at a speed that makes real-time use possible. It's modeled as a pipeline that reused some of the ANNIE GATE components and develop new ones. We can found also SemNews [13] which is a Semantic Web-based application that aims at accurately extracting information from heterogenous sources. It seeks to discover the meaning of news items. These items are retrieved from RSS Feeds and are processed by the NLP engine OntoSem. Our work also involves the processing of RSS Feeds but using the statistical approach and not the linguistic approach. The latter constitute a hindrance since linguistic expert uses rules for manually defining event patterns, which is prone to errors.

Several efforts have focused on social media analysis [7,3]. There are those which have focused on Flickr tags as significant descriptors [17,1]. Others are interested in the wealth of available context [4,8] including date and location and concluded that despite the noise that clutters the social media, take advantage of all its context can provide relevant information about the event. Most techniques mainly focused on context and ignore time needed for the context analysis. In this respect, we explore the rich context of social media with respect to the time constraint providing top-tier accuracy with a fraction the training time of alternative methods.

There are various applications of the Naive Bayes algorithm. It has been applied for automatic categorization of email into folders [5]. Email arrives in a stream over time. It was mentioned that Naive Bayes is the fastest algorithm compared to, respectively, MaxEnt, SVM and Winnow. Naive Bayes was also used as a pre-trained model for real-time network traffic classification [6]. Naive Bayes represents,yet, one of the most popular machine learning models applied in the spam filtering domain [18]. Importantly, the learning process of Naive Bayes is extremely fast compared with current discriminative learners, which makes it practical for large real-world applications. Since the training time complexity of Naive Bayes is linear to the number of training data, and the space complexity is also linear in the number of features, it makes Naive Bayes both time and storage efficient for practical systems. This led us to opt for the choice of the Naive Bayes algorithm for social media RSS Feeds processing.

3 RssE-Miner: Efficient Events Mining from Social Media RSS Feeds

We elaborate on event identification in social media RSS Feeds. To fulfill this task, we propose our approach for efficient events mining from social media RSS Feeds (c.f., figure 1). Indeed, The Naive Bayes classifier is our choice in the

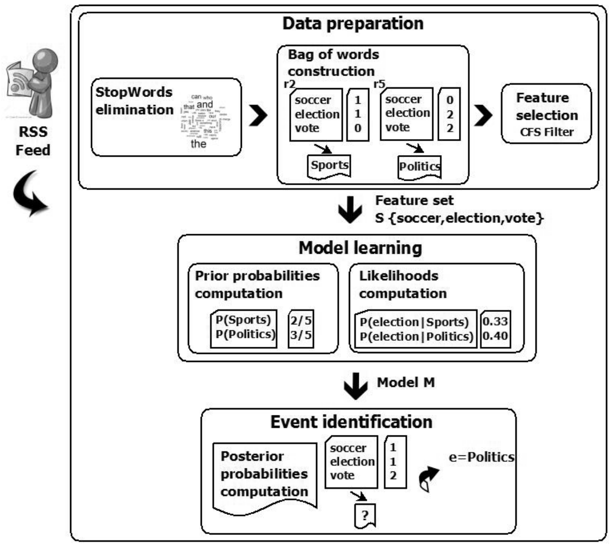


Fig. 1. The RssE-Miner approach

classification-based technique to predict when RSS Feed items correspond to the same event. The classifier presented in this section is based on a probabilistic model simply trying to give each resource r the label of the event e which is most likely. Hence, our proposed approach has three steps: **Data preparation**, **Model learning** and **Event identification**. In the following, we describe these three steps in more details.

3.1 Data Preparation

Common Refinements such as Removing Stop Words and Symbols are applied to the data. Each RSS Feed item, which is our resource, is represented as a bag of words *w1*,*w2*,*w3* etc. There are many design choices in the feature construction. In this paper, we use the traditional bag-of-words document representation. We do not apply stemming. Finally, we apply feature selection by identifying the most salient features for learning. In fact, CFS(Correlation based Feature Selection) filter is used[10]. It is a simple filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function:

$$M_S = \frac{k\overline{h_{cf}}}{\sqrt{k + k(k-1)\overline{h_{ff}}}}. \tag{1}$$

where M_S is the heuristic "merit" of a feature subset S containing k features, $\overline{h_{cf}}$ is the mean feature-class correlation ($f \in S$), and $\overline{h_{ff}}$ is the average feature-feature inter-correlation. In this paper, Best first heuristic search is used.

3.2 Model Learning

The Naive Bayes classifier is provided by a simple theorem of probability known as Bayes' rule [15]:

$$P(e|r) = P(e).\frac{P(r|e)}{P(r)}. \tag{2}$$

Hence, the Naive Bayes classifier need to compute P(e) and P($r|e$). Since our resource is represented as a bag of words, required P(e_j) and P($w_k|e_j$) terms are calculated for each e_j.

The former is given by:

$$P(e_j) = \frac{|r_j|}{|r|}. \tag{3}$$

where $|r_j|$ is a subset of resources for which the target event is e_j.

The latter,for each word w_k in the vocabulary, is given by:

$$P(w_k|e_j) = \frac{n_k + l}{n + l|Vocabulary|}. \tag{4}$$

where n_k is the total number of occurrences of w_k where target event is e_j, n is the total number of words in all training examples whose target value is e_j and l is the Laplacian smoothing.

The likelihoods P($w_1, w_2, ..., w_n|e_j$) are computed using the (naive) independence assumption. A common strategy is to assume that the distribution of $w_1, w_2, ..., w_n$ conditional on e_j can be decomposed in this fashion for all e_j:

$$P(w_1, w_2, ..., w_n|e_j) = \prod_i P(w_i|e_j). \tag{5}$$

The following section will explain the event identification stuff in detail.

3.3 Event Identification

Once the model learning step is performed, we proceed with testing the model for identifying the appropriate event. Classification will occurs when event probability is calculated. To classify, we must find the class label e which is most likely to generate r. Then, we choose e which gives r the best score according to P($e|r$):

$$g(r) = \arg\max_e P(e)P(r|e). \tag{6}$$

And according to the equation 5 and the equation 6, we have:

$$e = \arg\max_{e_j \in E} P(e_j) \prod_i P(w_i|e_j). \tag{7}$$

Typically, the denominator in equation 2 is not explicitly computed since it is the same for all e_j.

3.4 RssE-Miner: Application

Assuming that our data is passed through the Data preparation step, Table 1 represents the output of this step. In this example, we set the Laplacian smoothing l equal to 1.

Table 1. Model learning example

w	soccer	election	vote	label
r1	1	0	0	Sports
r2	1	1	0	Sports
r3	0	0	1	Politics
r4	0	1	1	Politics
r5	0	2	2	Politics

Table 2. P($word|label$) calculation

word	label	P($word|label$)
election	Sports	0.33
election	Politics	0.40
soccer	Sports	0.50
soccer	Politics	0.10
vote	Sports	0.17
vote	Politics	0.50

The model computes the prior probabilities for each class label. Then, Probabilities for every word are calculated.

$$P(Sports) = 2/5. \tag{8}$$

$$P(Politics) = 3/5. \tag{9}$$

The word "election" occurs 1 times in "Sports" resources.

The total number of words in "Sports" resources = 1+1+1= 3. Then

$$P(\text{"}election\text{"}|Sports) = (1 + 1)/(3 + 3) = 1/3. \tag{10}$$

The word "election" occurs 3 times in "Politics" resources.

Then

$$P(\text{"}election\text{"}|Politics) = (3 + 1)/(7 + 3) = 2/5. \tag{11}$$

Table 2 resumes this stuff.

We try to treat the same example mentioned above to test our model. Hence, Table 3 provides a resource that we seek the corresponding event.

Table 3. Event identification

w	soccer	election	vote	label
r6	1	1	2	?

– e_j=Sports :

$$\begin{aligned} P(r6|Sports) &= P(\text{"}soccer\text{"}|Sports)P(\text{"}vote\text{"}|Sports)^2 P(\text{"}election\text{"}|Sports) \\ &= 0.0048 \end{aligned} \tag{12}$$

– e_j=Politics :

$$\begin{aligned} P(r6|Politics) &= P(\text{"}soccer\text{"}|Politics)P(\text{"}vote\text{"}|Politics)^2 P(\text{"}election\text{"}|Politics) \\ &= 0.010 \end{aligned} \tag{13}$$

Then the event with the highest posterior probability, is selected.

$$e_j = Politics. \tag{14}$$

For this example, the event is correctly identified.

4 Experimental Evaluation

In this section we describe how our approach is evaluated on two real world datasets derived from Flickr. We use traditional classification accuracy[1] as well as precision[2] and recall[3] as our evaluation measures. We elaborate on our choice of Naive Bayes classifier.

4.1 Experimental Settings

Dataset Collect. We collected our dataset from the online photo management and sharing application Flickr using the Flickr API[4]. It consists of RSS Feeds of two real world datasets Upcoming and Last.fm. In this section we describe these two datasets and provide some basic statistics on their content. The dataset used is a set of news items fetched from Flickr RSS Feeds from January to June 2006. The Upcoming dataset contains 5778 images spread over 362 unique events. The Last.fm dataset contains 3356 images spread over 316 unique events. In order to emulate real-world scenario, we order the items in the dataset by their time of upload.

Baseline Models. To the best of our knowledge, event identification (using Naive Bayes) in such social media sites have never been modeled before. Thus, for enhancing the efficiency as well as the effectiveness of our approach, we compare the results of our approach to three baselines:

- **Most popular events identified:** For each event, we counted in how many resources it occurs and used the resources ranked by event occurrence count. For each event, the resources are randomly selected.
- **Most popular event aware extracted:** Events are weighted by their co-occurrence with a given event. Then, resources are ordered without validation.
- **SMO:** we compare our approach vs the Becker et al.(2010) [4] approach (we only make use of its classification-based technique part). In fact, SVM was used to learn document similarity functions for social media. In other

[1] Ratio of correctly classified instances to the total number of instances in the test set.
[2] percentage of resources correctly predicted in the class C from those predicted in this class.
[3] Percentage of resources correctly predicted in the class C from those actually in this class.
[4] `http://www.flickr.com/services/api/`

words, Becker et al.(2010) used SVM as a classifier with similarity scores as features to predict whether a pair of documents belongs to the same event. They selected Weka's sequential minimal optimization implementation. In this respect, We implement Naive Bayes as a part of the Weka[5] software system.

4.2 Efficiency of Our Approach

We report in the following results averaged over 10 test runs. We empirically decide to use only description, title and tags features. Indeed, the presence of other features such as location is an indication of document dissimilarity.

Dataset Phenomena. We evaluate the performance of the proposed approach on a sparse and a dense datasets. The Upcoming dataset is a sparse data since it contains different kind of events which are of public interest. That's why, it is less likely to find two or more items that belong to the same event. thus, it includes fewer items per event. However, the Last.fm dataset is a dense data since it includes only events in the area of music. That's why, it is more likely to find many items per event. As mentioned in Figure 1, the behavior of the two classifiers is almost the same in the two datasets.

Figure 2 (Left), depicts averages of accuracy on the Upcoming sparse dataset. Figure 2 (Right), depicts averages of accuracy on the Last.fm dense dataset. On both datasets, SVM demonstrates the highest accuracies. In fact, SVM outperforms Naive Bayes - by notable 1% in the case of Upcoming dataset and by notable 1% in case of Last.fm dataset, but the difference is not statistically significant. However, the performance of Naive Bayes could likely be improved by applying a more sophisticated smoothing method than Laplace. Naive Bayes accuracy, for now, is acceptable since we seek a compromise between runtime performance and classification accuracy. Next, we elaborate in the runtime performance.

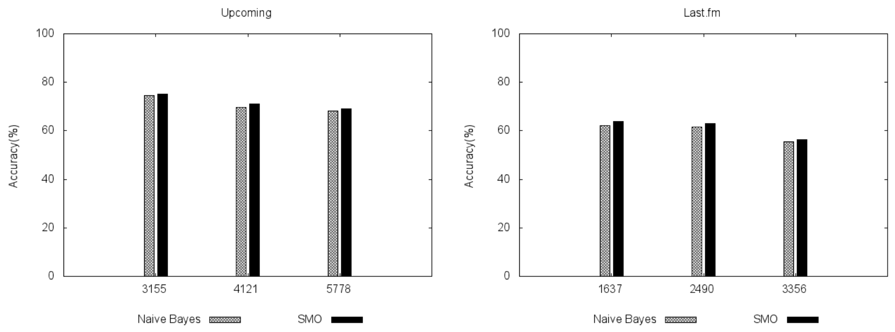


Fig. 2. Left: Average of Accuracy on the Upcoming dataset; **Right**: Average of Accuracy on the Last.fm dataset

[5] http://www.cs.waikato.ac.nz/ml/weka/

Running Time. As mentioned above, our main goal is to achieve a meaningful trade-off between runtime performance and classification accuracy. Table 4 shows that our approach has succeeded in fulfilling this task. Thus, according to the results given, we can point out that our approach outperforms baseline one. In fact, as expected, the Runtime of the SVM classifier are much slower than those achieved by our approach for both datasets. We note that Naive Bayes is by far the fastest classifier. It takes no more than 2.278 seconds on Upcoming dataset and no more than 0.739 seconds on Last.fm dataset. In particular, our approach outperforms the baseline one by a large and statistically significant margin. To this end, Naive Bayes makes real-time use possible. This is of paramount importance in the case of event identification in social media RSS Feeds as faster processing of data enables one to make better informed decisions.

In all, evaluation underlines fast and accurate performance by applying our approach. Results show that event identification using Naive Bayes model can work in near real-time without obvious decrease in accuracy.

Table 4. Average of Runtime on the Upcoming and Last.fm datasets

	Instances	Features	Events	Runtime(s)	
				Naive Bayes	SMO
Upcoming	3155	30	203	**0.596**	34.213
	4121	33	280	**1.215**	66.392
	5778	36	362	**2.278**	115.33
Last.fm	1637	21	171	**0.18**	24.571
	2490	27	243	**0.519**	50.143
	3356	23	316	**0.739**	85.829

4.3 Effectiveness of Our Approach

We present in figure 3 precision as well as recall measures in Upcoming and Last.fm datasets. Indeed, according to the sketched histograms, we can point out that our approach outperforms both baselines. On Upcoming dataset, the average recall achieves high percentage for higher value of N. Indeed, for N = 58, the average Recall is equal to 0.742, showing a drop of 98,38 % compared to the average Recall for N = 36. On Last.fm dataset, the average recall achieves high percentage for higher value of N. Indeed, for N = 46, the average Recall is equal to 0.878, showing a drop of 99,4 % compared to the average Recall for N = 41. In this case, for a higher value of N, by matching resources with their corresponding events, the proposed approach can achieve event identification task successfully. In addition, the percentage of precision for the proposed model outperforms the two baselines. On Upcoming dataset, our approach achieves the best results when the value of N is around 58. In fact, for N = 58, it has an average of 68,3% showing an exceeding about 4% against the first baseline and around 59,6% against the second one. On Last.fm dataset, our approach achieves the best results when the value of N is around 41. In fact, for N = 41, it has an average of 87,3% showing an exceeding about 12.9% against the first baseline and around 64,5% against the second one. These results highlight that the proposed approach can better improve event identification task even for a high number of extracted events.

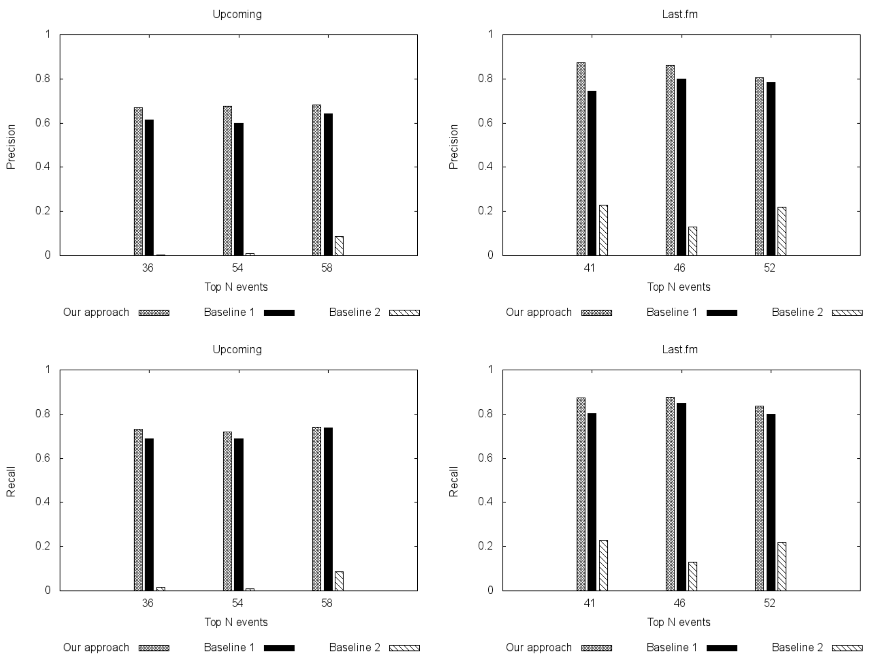


Fig. 3. Left: Precision and Recall on the Upcoming and Last.fm datasets

4.4 Online Evaluation

We present in figure 4 the runtime of RssE-Miner. Since it is hard to measure the exact runtime of the proposed approach, we simulated an online execution of our system among the Upcoming as well as the Last.fm datasets with different

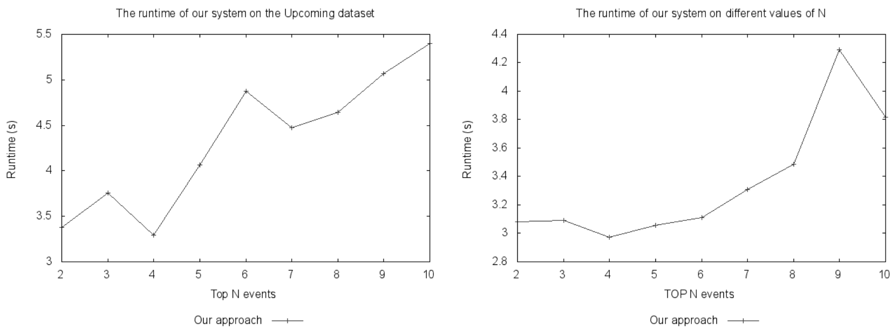


Fig. 4. Left: The runtime of our system online with different values of N on the Upcoming dataset; **Right**: The runtime of our system online with different values of N on the Last.fm dataset

values of N, i.e., the number of extracted events, ranging from 2 to 10. Hence, for each Flickr RSS Feed, we report the average runtime of the related top N events extracted. With respect to Figure 4, the maximum value of runtime is about 5.403(s) in the Upcoming dataset and about 4.292(s) in the Last.fm dataset, whereas the minimum value is around 3.292(s) in the Upcoming dataset and about 2.975(s) in the Last.fm dataset which is efficient and satisfiable.

5 Conclusion and Future Work

In this paper, we have tackled the task of event identification in social media RSS Feeds. We have formulated this task as a real-time problem and introduced a novel probabilistic approach for events mining from heterogenous social media RSS Feeds, called RssE-Miner, in order to get out these events. In particular, our approach relies on a better trade-off between event mining accuracy and swiftness by applying the probabilistic Naive Bayesian model to Flickr data. Our experiments suggest that our approach yields better performance than the baselines on which we build. To the best of our knowledge, event identification (using Naive Bayesian model) in such social media sites have never been modeled before. In future work, we will focus on further study other more sophisticated smoothing method than Laplace to improve Naive Bayes performance. Our future research will focus also on event ontology enrichment. Indeed, from these events, we aim to enrich an event ontology. Such an ontology is useful in providing accurate, up-to-date information in response to user queries.

Acknowledgments. This work was partially supported by the project Utique CMCU 11G1417.

References

1. Ahern, S., Nair, R., Kennedy, L., Naaman, M., Rattenbury, T.: How flickr helps us make sense of the world: context and content in community-contributed media collections. In: Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA 2007, pp. 631–640. ACM (2007)
2. Allan, J.: Introduction to topic detection and tracking. In: Topic Detection and Tracking: Event-Based Information Organization, pp. 1–16. Kluwer Academic Publishers (2002)
3. Amer-Yahia, S., Lakshmanan, L.V.S., Benedikt, M., Stoyanovich, J.: Efficient network aware search in collaborative tagging sites. In: Proc. VLDB Endow., vol. 1(1), pp. 710–721 (2008)
4. Becker, H., Naaman, M., Gravano, L.: Learning similarity metrics for event identification in social media. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 291–300. ACM (2010)
5. Bekkerman, R., Mccallum, A., Huang, G.: Automatic categorization of email into folders:benchmark experiments on enron and sri corpora. Technical Report, Computer Science department, IR-418. pp. 4–6

6. Dann Wei Li, R., Abdin, K., Moore, A.: Approaching real-time network traffic classification. Technical Report, RR-06-12, Department of Computer Science, Queen Mary, University of London (October 2006)
7. Donato, D., Gionis, A., Agichtein, E., Castillo, C., Mishne, G.: Finding high-quality content in social media. In: Proceedings of the First ACM International Conference on Web Search and Data Mining, WSDM 2008, pp. 183–194. ACM (2008)
8. Drumond, L., Buza, K., Reuter, T., Cimiano, P., Schmidt-Thieme, L.: Scalable event-based clustering of social media via record linkage techniques. In: Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, ICWSM 2011, pp. 313–320. AAAI Press (2011)
9. Forman, G., Cohen, I.: Learning from Little: Comparison of Classifiers Given Little Training. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 161–172. Springer, Heidelberg (2004)
10. Hall, M.A.: Correlation-based Feature Selection for Machine Learning. Doctoral thesis, The University of Waikato (April 1999)
11. Hogenboom, A., Hogenboom, F., Frasincar, F., Kaymak, U., van der Meer, O., Schouten, K.: Detecting Economic Events Using a Semantics-Based Pipeline. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011, Part I. LNCS, vol. 6860, pp. 440–447. Springer, Heidelberg (2011)
12. Hurst, M., Sayyadi, H., Maykov, A.: Event detection and tracking in social streams. In: ICWSM. The AAAI Press (2009)
13. Java, A., Finin, T., Nirenburg, S.: Semnews: A semantic news framework. In: Proceedings of the 21st National Conference on Artificial Intelligence, AAAI 2006, pp. 1939–1940. AAAI Press (2006)
14. Kumaran, G., Allan, J.: Text classification and named entities for new event detection. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2004, pp. 25–29 (2004)
15. Lewis, D.D.: Naive (Bayes) at Forty. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 4–15. Springer, Heidelberg (1998)
16. Papka, R., Allan, J., Lavrenko, V.: On-line new event detection and tracking. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1998, pp. 37–45. ACM (1998)
17. Ramage, D., Heymann, P., Garcia-Molina, H.: Social tag prediction. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, pp. 531–538. ACM (2008)
18. Song, Y., Kolcz, A., Lee Giles, C.: Better naive bayes classification for high-precision spam detection. Softw. Pract. Exper. 39(11), 1003–1024 (2009)

k Nearest Neighbor Using Ensemble Clustering

Loai AbedAllah and Ilan Shimshoni

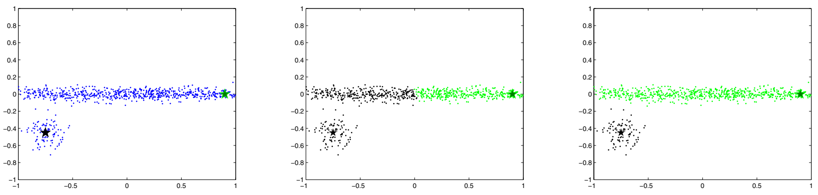
[1] Department of Mathematics, University of Haifa, Israel
Department of Mathematics, The College of Saknin, Israel
[2] Department of Information Systems, University of Haifa, Israel
`loai1984@gmail.com, ishimshoni@mis.haifa.ac.il`

Abstract. The performance of the k Nearest Neighbor (kNN) algorithm depends critically on its being given a good metric over the input space. One of its main drawbacks is that kNN uses only the geometric distance to measure the similarity and the dissimilarity between the objects without using any statistical regularities in the data, which could help convey the inter-class distance. We found that objects belonging to the same cluster usually share some common traits even though their geometric distance might be large. We therefore decided to define a metric based on clustering. As there is no optimal clustering algorithm with optimal parameter values, several clustering runs are performed yielding an ensemble of clustering (EC) results. The distance between points is defined by how many times the objects were not clustered together. This distance is then used within the framework of the kNN algorithm (kNN-EC). Moreover, objects which were always clustered together in the same clusters are defined as members of an equivalence class. As a result the algorithm now runs on equivalence classes instead of single objects. In our experiments the number of equivalence classes is usually one tenth to one fourth of the number of objects. This equivalence class representation is in effect a smart data reduction technique which can have a wide range of applications. It is complementary to other data reduction methods such as feature selection and methods for dimensionality reduction such as for example PCA. We compared kNN-EC to the original kNN on standard datasets from different fields, and for segmenting a real color image to foreground and background. Our experiments show that kNN-EC performs better than or comparable to the original kNN over the standard datasets and is superior for the color image segmentation.

Keywords: Clustering, Classification, Ensemble Clustering, Unsupervised Distance Metric Learning.

1 Introduction

The performance of many learning and data mining algorithms depend critically on there being given a good metric over the input space. Learning a "good" metric from examples may therefore be the key of a successful application of these algorithms. For instance, many researchers have demonstrated that k-nearest neighbor (kNN) [8] classification can be significantly improved by learning a distance metric from labeled examples [5,10,14,16]. However, like any classifier kNN has some drawbacks. One of its main drawbacks is that most implementations of kNN use only the geometric distance to measure the similarity and the dissimilarity between the objects without using any statistical regularities in the data. Thus, it does not always convey the inter-class distance. The following example illustrates this situation. Given the dataset in Figure 1(a) with two labeled points. When the classifier uses the Euclidean distance, it works "poorly" and many points belonging to the green class were classified to be black (b).



(a) (b) (c)

Fig. 1. Euclidean distance does not reflect the actual similarity

To overcome this problem we turned to clustering for defining a better metric. As there is no optimal clustering algorithm with optimal parameter values, several clustering runs are performed yielding an ensemble of clustering results. The distance between points is defined by how many times the points were not clustered together. This distance is then used within the framework of the kNN algorithm (kNN-EC). Returning to the previous example in Figure 1(a), we can see that it worked very well (c). Moreover, points that are always clustered together in the same cluster (distance= 0) are defined as members of an equivalence class. As a result, the algorithm now runs on equivalence classes instead of single points. In our experiments the number of equivalence classes is usually less than one tenth to one fourth of the number of points. This equivalence class representation is in effect a novel data reduction technique which can have a wide range of applications. It is complementary to other data reduction methods such as feature selection and methods for dimensionality reduction such as the well known Principal Component Analysis (PCA).

This paper is organized as follows: Related work on distance metric learning is discussed in Section 2. The distance metric using ensemble clustering is described in Section 3. Section 4 describes the ensemble clustering method using the mean shift and the k-means clustering algorithms. Experimental results are presented in Section 5. Finally, our conclusions are presented in Section 6.

2 Related Work

A large body of work has been presented on the topic of distance metrics learning, and we will just briefly mention some examples. Most of the work in distance metric learning can be organized into the following two categories: Supervised/semi-supervised distance metric learning and unsupervised distance metric learning.

Most supervised/semi-supervised distance metric learning attempt to learn metrics that keep data points within the same classes close, while separating data points from different classes [5,16]. Goldberger et. al [14] provided a distance metric learning method to improve the classification of the kNN algorithm. They use a gradient decent function to reduce the chance of error under the stochastic neighborhood assignments. Domeniconi et. al [10] proposed to use a locally adaptive distance metric for kNN classification such as the decision boundaries of SVMs. Shalev-Shwartz et. al [22] considered an online method for learning a Mahalanobis distance metric. The goal of their method is to minimize the distance between all similarity labeled inputs by defining margins and inducing hinge loss functions. Recently, a similar method was presented by Weinberger et. al [24] which uses only the similarly labeled inputs that are specified as neighbors in order to minimize the distance.

The unsupervised distance metric learning takes an input dataset, and finds an embedding of it in some space. Many unsupervised distance metric learning algorithms have been proposed. Gonzales and Woods [15] provided the well known PCA which finds the subspace which best maintains the variance of the input data. Tenenbaum et. al [23] proposed a method called ISOMAP which finds the subspace which best maintains the geodesic inter-point distances. Saul et. al [21] provided a locally linear embedding (LLE) method to establish the mapping relationship between the observed data and the corresponding low dimensional data. Belikin et. al [1] presented an algorithm called the Laplacian Eigenamp to focus on the maintenance of local neighbor structure.

Our method falls into the category of the unsupervised distance metric learning. Given an unlabeled dataset, a clustering procedure is applied several times with different parameter values. The distance between points is defined as a function of the number of times the points belonged to different clusters in the different runs.

A clustering based learning method was proposed in Derbeko, El-Yaniv, and Meir [9]. There, several clustering algorithms are run to generate several (unsupervised) models. The learner then utilizes the labeled data to guess labels for entire clusters (under the assumption that all points in the same cluster have the same label). In this way the algorithm forms a number of hypotheses. The one that minimizes the PAC-Bayesian bound is chosen and used as the classifier. They assume that at least one of the clustering runs produces a good classifier and that their algorithm finds it.

Our work is different from these techniques in several ways especially on the assumptions that they made. Unlike other techniques, we only assume that the equivalence classes, which were built by running the clustering algorithm several times, are quite pure. We also did not assume that at least one of the clustering runs produces a good classifier. Rather that the true classifier can be approximated quite well by a set of equivalence classes (i.e the points which always belong to the same clusters in the different clustering iterations will define an equivalence class) instead of single points and the distance metric defined between these equivalence classes.

3 Distance Metric Learning Using Ensemble Clustering

Consider the following paradigm. Let X be the *unlabeled data* - a set of unlabeled instances where each x_i is a vector in some space χ. Instances are assumed to be i.i.d. distributed according to some unknown fixed distribution ρ. Each instance x_i has a label $w_i \in \mathcal{W}$ (where in our case $\mathcal{W} = \{0,1\}$) distributed according to some unknown conditional distribution $P(w|x)$. Let $D = \{\langle x_i, f(x_i)\rangle : x_i \in X, i = 1, ..., N_D\}$ be the *training data*—a set of labeled examples already known.

The Euclidean distance does not always reflect the actual similarity or dissimilarity of the objects to be classified. We found that on the other hand points belonging to the same cluster usually share some common traits even though their geometric distance might be large.

The main problem with such an approach is that there is no known method to choose the best clustering. There have been several attempts to try to select the optimal parameters values of the clustering algorithms in supervised and unsupervised manners mainly within the range image and color image domain, but a general solution to this problem has not been found [19,3,25]. We therefore decided to run different clustering algorithms several times with different parameter values. The result of all these runs yields a cluster ensemble [11].

The clustering results are stored in a matrix denoted the *clusters matrix* $C \in Mat_{N\times K}$, where K is the number of times the clustering algorithms were run. The ith row consists of the cluster identities of the ith point in the different runs. This results in a new instance space $\chi_{cl} = \mathbb{Z}^K$ which contains the rows of the *clusters matrix*. Let X_{cl} be an *unlabeled training set*, a set of objects drawn randomly from χ_{cl} according to distribution ρ. Let $D_{cl} = \{\langle x_i, f(x_i)\rangle : x_i \in X, i = 1, ..., N_D\}$ be the *training data*—a set of labeled examples from X_{cl}.

The goal now is to adapt the kNN classifier to work with a distance function based on the new instance space. The new distance between points from this space should be defined in such a way as to reflect our intuitive notion on proximity among the corresponding points.

Given two points $x, y \in \chi_{cl}$ we define a new distance function d_{cl} as:

$$d_{cl}(x,y) = \sum_{i=1}^{K} dis(x_i, y_i), \tag{1}$$

where $dis(x_i, y_i) = \begin{cases} 1 & x_i \neq y_i \\ 0 & x_i = y_i \end{cases}$ be the metric of a single feature. This metric is known as the *Hamming distance.* Over this metric we define the following equivalence relation.

Let E be a binary relation on χ_{cl}, where E defined as

$$\forall x, y \in \chi_{cl}, (x, y) \in E \Leftrightarrow d_{cl}(x, y) = 0.$$

The relation E is an equivalence relation on χ_{cl}. By this relation, points which always belong to the same clusters in the different clustering iterations will define an equivalence class $[\cdot]_E$. Thus, all the equivalent points will be represented by a single point in the quotient set and we can work with X/E yielding $C' \in Mat_{M \times K}$, where $M = |X/E|$. On the other hand, points which always belong to different clusters in all the clustering iterations will be infinitely distant (i.e. $d_{cl}(x, y) = \infty$ if and only if x and y always belong to different clusters in all the clustering iterations). Thus, x is a neighbor of y if and only if $d_{cl}(x, y) < \infty$. The set of the neighbors of x will be defined as: $\mathfrak{N}_x = \{y | d_{cl}(x, y) < \infty\}$. For each $x \in X$ $\mathfrak{N}_x \neq \emptyset$, since by using the reflexive property of E, we get $d_{cl}(x, x) = 0 < \infty$, thus, $x \in \mathfrak{N}_x$.

This new metric is used in the *k*NN classifier instead of the Euclidean distance. In this setting all the unlabeled points in X_{cl} will be labeled according to a given training dataset D_{cl}. Experiments using this method are presented in Section 5.

The main presumption made by the algorithm is that *equivalent points have the same label* but this assumption does not always hold in practice. To overcome this hurdle several possible options exist. One possibility is that for each equivalence class $x_{cl} \in D_{cl}$ several points from its equivalence class are labeled and x_{cl} will then be labeled according to the majority voting. Another option is to label x_{cl} according to its center point. Thus, with high probability a point will be selected from the majority class of the equivalence class. It is also possible to run the clustering algorithms more times, increasing the number of the equivalence classes yielding smaller but hopefully purer equivalence classes.

4 Ensemble Clustering Using Mean Shift and *k* Means Algorithms

As mentioned above the main problem with an approach based on clustering is that there is no known method to choose the best clustering. It is unknown how many clusters should be, their shapes, which clustering algorithm is best, and which parameter values should be used? We therefore decided to run different clustering algorithms several times with different parameter values.

Our algorithm however is general and any good clustering algorithm could be used. We decided to work with the well known *k*-means algorithm [18] and the mean shift clustering algorithm [13,7] in order to build the clusters matrix.

For completeness we will now give a short overview of the mean shift algorithm. Mean shift is a non-parametric iterative clustering algorithm. The fact that mean shift does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters, makes it ideal for handling clusters of arbitrary shape and number. It is also an iterative technique, but instead of the means, it estimates the modes of the multivariate distribution underlying the feature space. The number of clusters is obtained automatically by finding the centers of the densest regions in the space (the modes). The density is evaluated using kernel density estimation which is a non-parametric way to estimate the density function of a random variable. This is also called the Parzen window technique. Given a kernel K, bandwidth parameter h, which is a smoothing parameter of the estimated density function, the kernel density estimator for a given set of d-dimensional points is:

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{2}$$

For each data point, a gradient ascent process is performed on the local estimated density until convergence. The convergence points represent the modes of the density function. All points associated with the same convergence point belong to the same cluster.

We worked with the two mean shift algorithm types; the simple, and the adaptive (more details in [13,7]). The simple mean shift works with a fixed bandwidth h. We chose 80 different values of h with fixed intervals from 0.1 to 0.9 of the space size. The adaptive mean shift algorithm is given the number of neighbors k as a parameter and the bandwidth is determined for each point in the data as the distance to its k'th neighbor. We chose 30 different values of k with fixed intervals between 1% to 30% of N (for more details see Section 5).

Some clustering algorithms work with continuous parameters, like the mean shift algorithm described above, or with continuous weights over the features, like the EDISON program which will be discussed in Section 5.2. In these cases the differences between two consecutive iterations might be small. There are two possibilities to deal with these similar clusterings: The first one is to eliminate the similar clustering results or simply take all of them. We preferred the second one because if a set of samples were together in several clustering runs it means that they might have some common features. So if we eliminate them we stand to loose this information. However, it is not efficient to preserve similar clustering runs. Therefore, we decided to join them, as a result the dimensionality of the data is reduced. We use the Rand index [20] which is a measure of similarity between two data clusterings. Let $C1, C2$ be two clustering iterations, then the measure between them is:

$$R(C1, C2) = \frac{\alpha + \beta}{\alpha + \beta + \gamma + \delta} = \frac{\alpha + \beta}{\binom{n}{2}}, \tag{3}$$

where α describes the number of pairs of elements in the instance space that are in the same set (i.e cluster) in $C1$ and in the same set in $C2$, β describes the number of pairs of elements in the instance space that are in the different set in $C1$ and in the different set in $C2$, γ describes the number of pairs of elements in the instance space that are in the same set in $C1$ and in the different set in $C2$ and δ describes the number of pairs of elements in the instance space that are in the different set in $C1$ and in the same set in $C2$.

Similar clusterings are represented by a single column, weighted by the number of clustering it represents. Accordingly, the metric function has become:

$$d_{cln}(x, y) = \sum_{i=1}^{q} n_i dis(x_i, y_i), \tag{4}$$

where $x, y \in \chi_{cln}$ are two points in the new weighted space, q is the dimension of χ_{cln}, and n_i is the weight of each representative column.

The advantage of this method is that it maintains the relation between the samples according to the clustering results, while maintaining a relatively small dimension of the clustering matrix.

This method worked quite well for mean shift clustering as the bandwidth acts as a smoothing parameter for the density estimation. However, for k-means the differences between consecutive runs of the algorithm were significant and thus columns could not be joined.

5 Experiments

To validate the efficiency of kNN-EC we conducted a series of experiments using standard datasets from different fields. An additional experiment was conduct on a color image, where the mission is to classify each pixel as a foreground or background pixel. We compare the performance of kNN-EC to that of kNN on the same datasets. Both algorithms were implemented in Matlab.

5.1 Datasets

In order to evaluate the performance of kNN-EC we ran experiments on four datasets; the image segmentation dataset (UCI Machine Learning Repository [12]), the breast cancer dataset (LIBSVM library [4]), Leo Breiman's ringnorm [2], and a real color image. The image segmentation dataset contains 2310 instances, which are divided into 7 classes. Since we choose to work with a binary kNN, the classes were joined to create two class labels (as was done in [17]) one corresponding to BRICKFACE, SKY and FOLIAGE and the other corresponding to CEMENT, WINDOW, PATH and GRASS. The breast cancer dataset contains 683 instances, which are divided into two class labels, such that 444 points are from the first class and the rest are from the second. Leo Breiman's ring norm dataset contains 7400 instances, two-class classification problem. Each class is drawn from a multivariate normal distribution. The last dataset is a color image. More details on this experiment will be described in Section 5.2. All these datasets were labeled, but this knowledge was used only to evaluate the quality rate of the resulting classifier. In all experiments the algorithm assumes that these datasets are unlabeled.

The mean shift algorithm was run with the k or h values, described above. For the breast cancer and ring norm datasets the mean shift algorithm did not yield good clustering (i.e one cluster or the same clustering for all runs). So we use the k-means algorithm for these two datasets. For the breast cancer the k-means algorithm was run with $k = 3..15$ and for the ring norm dataset the k-means algorithm was run with $k = 3..30$. The results are stored in the clusters matrix C. The equivalence relation E was employed to build the equivalence matrix C'. As can be seen in Table 1, the new space is usually smaller than the original space without the equivalence classes. The ratio between the sizes of the two spaces is given in the fourth column. Our algorithm assumes that points belonging to the same equivalence class have the same label. However, as can be seen from the last column the equivalence classes are not perfectly pure.

Table 1. Numerical Datasets properties

Dataset	Dataset size	Cluster matrix size	Equivalence matrix size	Ratio %	Purity
Image Segmentation	2310×19	2310×38	548×38	24%	97%
Breast Cancer	683×8	683×13	160×13	23%	98%
Ring Norms	7400×20	7400×28	7400×28	100%	100%

At the first stage of each algorithm a training set of size 20% to 40% of the dataset is randomly drawn and labeled. For each training dataset the algorithms run with different numbers of neighbor values (i.e $k = 3, 5, 7$). For each k the quality was evaluated by the ability of the classifier to label the rest of the unlabeled points. The results are averaged over 10 different runs on each dataset. A resulting curve was constructed for each dataset which evaluated how well the algorithm performed.

Results. As can be seen from Figure 2, the kNN-EC performs better than or comparable to the kNN with the Euclidean distance. The learning curves, which describe the accuracy for each classifier by computing the ratio of the correct classified instances to the whole unlabeled data, of the image segmentation dataset and that of the Breast Cancer datasets show that kNN-EC is comparable to the kNN while glancing at the learning curves of the ring norm dataset depict the superiority of kNN-EC. As Figure 2 shows the quality of kNN-EC is about 85% while the kNN quality is about 65%. Moreover we compute the runtime of the two algorithms when the training dataset includes 30% of the points, and $k = 5$. The runtime results are shown in Table 2.

Table 2. Runtime of the algorithms in seconds

Dataset	kNN	kNN-EC
Image Segmentation	0.45	0.15
Breast Cancer	0.15	0.01
Ring Norms	2.4	3.9

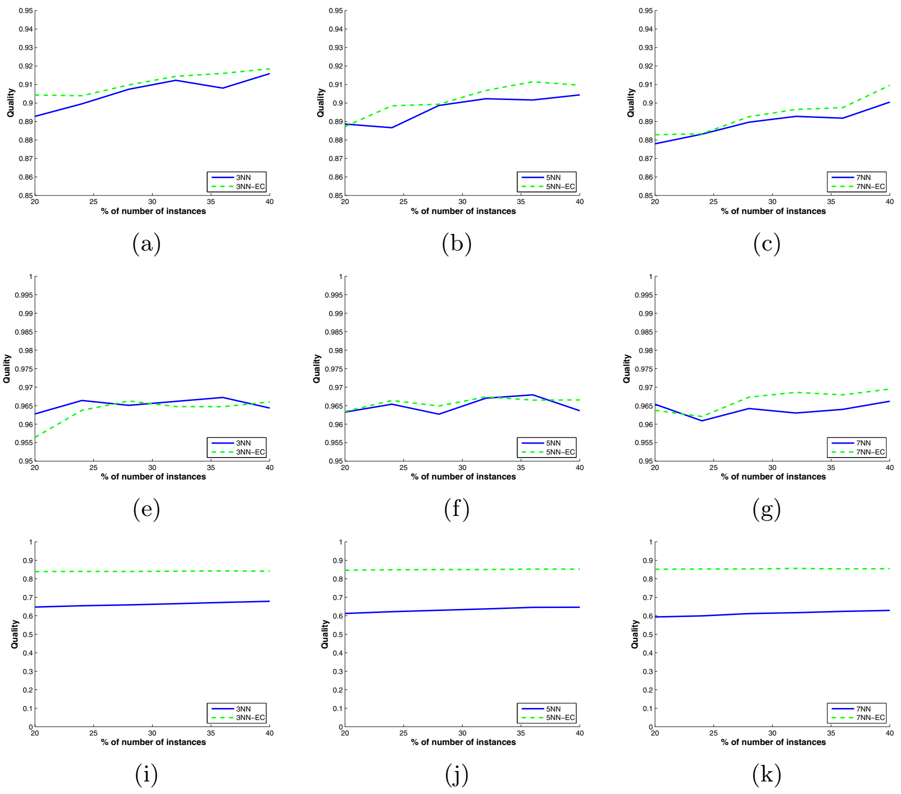


Fig. 2. Results of *k*NN and *k*NN-EC for the three datasets. The first row shows the learning curves of the Image Segmentation dataset, the second shows Breast Cancer dataset, and the third one shows the Ring Norm dataset. The columns show the learning curves for the different *k*'s values.

The Effect of the Purity of the Equivalence Classes. As shown in the previous subsection, the performance of the *k*NN-EC is not always superior. In this section we carried out an experiment in order to determine how much our algorithm depends on the purity of the equivalence classes. In the original experiments the equivalence classes were not pure, for instance the purity of the image segmentation dataset was 97%. In this experiment the classes were changed until the equivalence classes were pure (i.e 100%). As shown in this Figure 3 there is a linear trade off between the quality and the purity of the equivalence classes. The quality increased by about 3% while the purity increased from 97% to 100%.

The Effect of Number of the Clustering Iterations on the Performance of *k*NN-EC Algorithm. Another experiment was performed to determine how much our algorithm depends on the number of clustering iterations. In

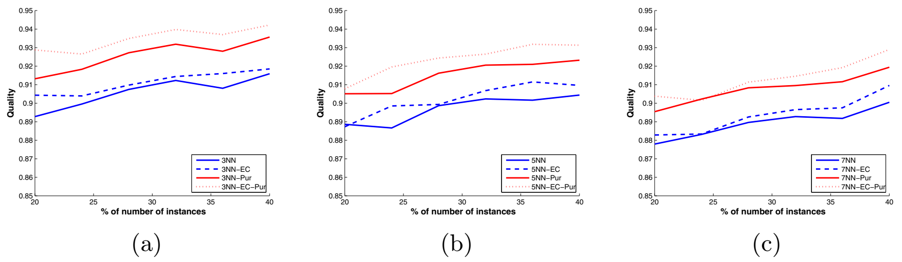


Fig. 3. Results of kNN and kNN-EC for the image segmentation dataset with the different k's values

this experiment we evaluate the performance of the 5NN-EC classifier on the ring norm dataset given 20% of the dataset as a training set. We run the k-means algorithm on three different ranges $k = 3..10$, $k = 3..20$ and $k = 3..30$ as shown in Table 3. As the number of the clustering runs increases, the purity of the equivalence classes increases, and the number of the equivalence classes increases dramatically. (When the clustering runs increased from 8 to 18 runs the purity increased from 93 to 99.8 and the equivalence classes increased from 3351 equivalence classes to 7171 equivalence classes). However, the performance of the algorithm preserves it stability (i.e the quality increased from 81% to 83%). This occurred because the distance function metric based on ensemble clustering is stable, and if for example an equivalence class was partitioned then the distance between the instances which were equivalent will be 1 instead of zero. Thus with hight probability they will still be classified to the same class.

Table 3. The effect of the number of the clustering iterations

k-Means	Equivalence matrix size	Purity %	Quality %
$k = 3..10$	3351×8	93	81
$k = 3..20$	7171×18	99.8	83
$k = 3..30$	7400×28	100	85

5.2 Experiments with Images

In a final set of experiments we tested our algorithm using a real color image. We use images for two reasons, first images provide large complex datasets and second that the results obtained by applying classification algorithms on images can be easily viewed and evaluated. This image contains three birds (shown in Figure 4(a)). It was manually segmented into two classes, the foreground (birds) and the background yielding the ground truth (as shown in Figure 4(b)).

The reader can appreciate that segmenting these images using a color based segmentation algorithm into foreground and background images will not be an easy task.

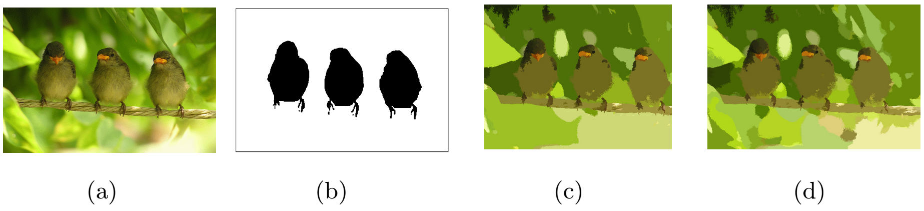


Fig. 4. (a) The original image with three birds. (b) The classified image (the goal). (c),(d) The output of the EDISON system.

We chose to work with the Edge Detection and Image Segmentation (EDISON) System. This program implements the mean shift image segmentation algorithm described in [6,7]. Each pixel in the image is represented by its two image coordinates and RGB color values yielding a 5D dataset. The user is asked to provide the algorithm with values for two bandwidths, one for the *spatial* domain h_s (the image coordinates) and the other for the *range* domain h_r (the RGB values). The output of this program is a clustered image. Each cluster was assigned a color, (i.e points in the same cluster have the same color). Figure 4 (c,d) shows some of these clustering results.

In our experiments we used the following values for the two bandwidths $h_s = \{5, 10, 20, 30\}$ and $h_r = \{10, 15, 20, 25, 30, 35\}$ yielding 24 clustered images. Results for which nearly the whole image belonged to a single cluster were automatically discarded. It is important to note that the original kNN classifier has to choose values for these bandwidths (or actually their ratio) in order to define the distance metric between points. As optimal values for these bandwidths are not available, it is not clear how this method can be compared to kNN-EC. In the experiments we therefore ran them using all 24 bandwidth pair values.

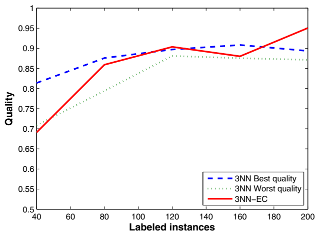
For each image the EDISON algorithm was run with the h_r and h_s values, described above, to build the clusters matrix C. We then joined clusterings with small Rand index measures and worked with the weighted C_w matrix. The equivalence relation E is employed to build the equivalence matrix C'. Table 4 summarizes the information about the three birds image.

In this table we can see that the new space is about 10 times smaller than the original space. As the complexity of the algorithm is $O(N^2)$, the running time of kNN-EC is two orders of magnitude smaller than the running time of kNN.

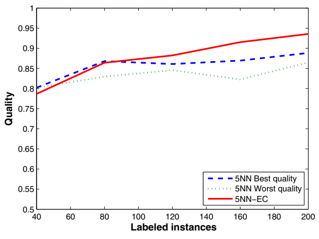
Table 4. Image properties

Dataset	Picture size	Cluster matrix size	Equivalence matrix size	Ratio %	Purity	Fg pixels
Three Birds	207×352	72864×24	8233×18	11%	100%	14407

The two classifiers were evaluated on several training sets of size 40 pixels to 200 pixels (less than 0.3% of the data), with different numbers of neighbors values (i.e $k = 3, 5$). As the optimal bandwidth parameters can not be found



(a)

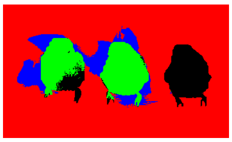


(b)

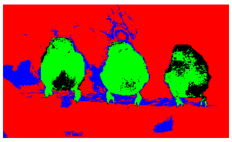
Fig. 5. Results of kNN and kNN-EC for the three birds image dataset with the different k's values

automatically in the kNN algorithm, then in the resulting curves we compared the kNN-EC with the best case and the worst case of the kNN for each training dataset which evaluated how well the algorithms perform (as shown in Figure 5).

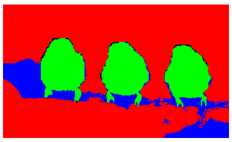
The experimental results shown in Figure 5 show that the kNN-EC performs better than the kNN with the Euclidean distance. Due to the learning curves presented in the figures below we see that our algorithm is superior, where its quality was around 95% while the best quality of the kNN algorithm was less than 90%. Figure 6 shows the superiority of the kNN-EC. It shows an example of running the kNN-EC and kNN (the best and the worst cases) algorithms with k=5, and with 120 labeled pixels as a training set.



(a)



(b)



(c)

Fig. 6. Results of 5NN and 5NN-EC for the three birds image dataset for given 120 labeled pixels as a training dataset. (a) The output for the worst case of kNN. (b) The output for the best case of kNN. (c) The output for the kNN-EC. The color of the pixels represents the results of the classifier. Red is background, green is birds, blue is wrong background and black is wrong birds pixels.

6 Conclusions and Future Work

In this work, we have presented a new unsupervised distance metric learning based on ensemble clustering and use it within the kNN classifier. Each data point is characterized by the identity of the clusters that it belongs to in several clustering runs. The distance metric is defined as the Hamming distance between these clustering results.

This new distance has two important contributions. The first contribution is that this distance is more meaningful than the Euclidean distance between points resulting in a better kNN classifier. The second and more general contribution results from the observation that all points which always belong to the same cluster form an equivalence relation. Thus, the algorithm only has to consider one member of each equivalence class. This reduces the complexity of the algorithm considerably (by at least two orders of magnitude in our case). Our algorithm however is only a private case of a more general concept, the concept of data reduction. This concept is orthogonal to other methods of data reduction such as feature selection or PCA which reduce the size of the representation of the data points but not their number.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15(6), 1373–1396 (2003)
2. Bias, L.: Variance and arcing classifiers. Tec. Report 460, Statistics department (1996)
3. Chabrier, S., Emile, B., Rosenberger, C., Laurent, H.: Unsupervised performance evaluation of image segmentation. EURASIP Journal on Applied Signal Processing, 1–12 (2006)
4. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), Software, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
5. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 26–33 (2005)
6. Christoudias, C., Georgescu, B., Meer, P.: Synergism in low level vision. In: Proceedings of International Conference on Pattern Recognition, pp. 150–155 (2002)
7. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5), 603–619 (2002)
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
9. Derbeko, P., El-Yaniv, R., Meir, R.: Explicit learning curves for transduction and application to clustering and compression algorithms. Journal of Artificial Intelligence Research 22(1), 117–142 (2004)
10. Domeniconi, C., Gunopulos, D., Peng, J.: Large margin nearest neighbor classifiers. IEEE Transactions on Neural Networks 16(4), 899–909 (2005)
11. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the Twenty-First International Conference on Machine Learning, pp. 36–43. ACM (2004)
12. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
13. Georgescu, B., Shimshoni, I., Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. In: Proceedings of the 9th International Conference on Computer Vision, pp. 456–463 (2003)
14. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood components analysis. In: Advances in Neural Information Processing Systems, vol. 17, pp. 513–520 (2004)

15. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
16. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(6), 607–616 (1996)
17. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. Machine Learning 54(2), 125–152 (2004)
18. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Symposium on Math., Statistics, and Probability, pp. 281–297 (1967)
19. Min, J., Powell, M., Bowyer, K.W.: Automated performance evaluation of range image segmentation algorithms. IEEE Transactions on Systems Man and Cybernetics-Part B-Cybernetics 34(1), 263–271 (2004)
20. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66, 846–850 (1971)
21. Saul, L.K., Roweis, S.T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. The Journal of Machine Learning Research 4, 119–155 (2003)
22. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proceedings of the Twenty-First International Conference on Machine Learning, pp. 94–102. ACM (2004)
23. Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for non-linear dimensionality reduction. Science 290(5500), 19–23 (2000)
24. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 10, 207–244 (2009)
25. Zhang, H., Fritts, J.E., Goldman, S.A.: Image segmentation evaluation: A survey of unsupervised methods. Computer Vision and Image Understanding 110(2), 260–280 (2008)

Collocation Pattern Mining in a Limited Memory Environment Using Materialized iCPI-Tree*

Pawel Boinski and Maciej Zakrzewicz

Poznan University of Technology Institute of Computing Science
{pawel.boinski,maciej.zakrzewicz}@cs.put.poznan.pl

Abstract. We consider the problem of executing collocation pattern queries in limited memory environments. Our experiments show that if the memory size is not sufficient to hold all internal data structures used by the iCPI-tree algorithm, its performance decreases dramatically. We present a new method to efficiently process collocation pattern queries using materialized, improved candidate pattern instance tree. We have implemented and tested the aforementioned solution and shown that it can significantly improve the performance of the iCPI-tree algorithm.

Keywords: spatial collocation, limited memory, pattern mining.

1 Introduction

Increasing popularity of devices with GPS modules, advances in sensing technology and decreasing cost of mass storage devices, results in rapid growth of spatial datasets. Spatial objects can be described by geometrical representation (e.g. point, rectangle etc.), a relative position as well as various non-spatial attributes. In this immense size of complex spatial data, valuable and interesting patterns can be hidden. *Spatial data mining* [11] is the process of extracting such patterns with respect to the location and spatial relationships between the objects. Similar techniques have been successfully applied in such 'classical' domains as marketing, customer relations, medicine etc. In the context of spatial data, one of the interesting tasks is to discover spatial collocation patterns.

The most common definition of a *spatial collocation pattern* (or in short a collocation) assumes that it is a subset of spatial features whose instances are frequently located together (in a spatial neighborhood). A *spatial feature* attribute has a boolean nature that provides information about occurrence of this feature in a particular location in space. Typical examples of spatial features include species, business types, point of interest (e.g. hospitals, airports) etc.

Collocation discovery problem can be defined as the process of searching for collocation patterns in datasets with spatial feature instances. It is substantially similar to the frequent itemset discovery problem [2], however, direct application

* This work was supported by grant 91-5169 from National Science Centre, Poland.

of the association mining algorithms (e.g Apriori [3]) is very challenging [10]. Difficulties arise from significant differences in the characteristics between classical market-basket data and spatial data. Particularly noteworthy is the difference in the concept of the sets from which instances of the frequent itemsets and collocations are constructed. Instances in the market-basket analysis are precisely nested in transactions, while instances of spatial features are embedded in a continuous space and share neighbor relationships. Therefore, the collocation discovery requires novel, specially designed algorithms to cope with this computationally demanding task.

Most of the existing algorithms for collocation discovery [4] [5] [12] [14] [15] are based on the method presented in [11]. Briefly, the proposed idea leverages the classical *Apriori* algorithm. This cyclical, iterative process consists in generating candidates and counting their usefulness. Due to the complexity of the spatial data, internal data structures used by collocation discovery algorithms can necessitate large amounts of system memory. Nanopoulos et al. consider in their work [9] real world database systems where OLTP and data mining queries for association rules discovery are executed simultaneously, imposing high requirements for hardware resources, including system memory and processor. In this paper, we focus on an efficient execution of the collocation discovery algorithm in a limited memory environment. For this purpose, we introduce materialization techniques and specially designed data access methods to improve performance by effective usage of the cache memory.

The structure of this paper is as follows. Section 2 contains basic definitions for the collocation discovery problem. In section 3 we present a brief description of the existing algorithm, our observations and a modification of the algorithm. Section 4 covers conducted experiments and their results.

1.1 Related Work

The problem of mining spatial association rules has been introduced in [8], where the authors proposed a method to find spatial features that are frequently associated with a particular, previously chosen feature. The main disadvantage was that this approach divided spatial dataset into artificial and possibly overlapping transactions. Therefore, computation of support measures could not be accurate.

In [7] and [11], the authors proposed a new measures of interest called *participation index* and *participation ratio* together with a concept of neighborhoods instead of the artificial transactions. A new algorithm, called *Co-location Miner*, has been introduced. Due to a join-based strategy of generating collocation instances, its performance decreases dramatically with increasing density of datasets. The problem has been addressed in [14], where the authors proposed a solution that requires to divide continuous spatial data into transactions, although spatial information has been kept in an additional structure. The number of expensive join-based operations experienced significant reduction, however the performance highly depended on the distribution of the spatial dataset.

A disparate approach to identifying collocation instances consists in dismissing a step for generating instances of candidates [15]. Instead of performing a computationally demanding join-based procedure, an additional data structure, called *star neighborhoods*, is introduced and serves as a base for instances generation. The number of potential collocation instances is dramatically reduced, however, some of them can form a star pattern instead of a clique, and therefore should be removed from the result set. In [12], the authors proposed a method that stores star neighborhoods in a tree structure. The modified procedure finds only proper collocation instances. Therefore, there is no need to perform an additional filtering step like in [15]. Recently, an *order-clique-based* algorithm [13] has been proposed to discover maximal collocations from input dataset. The advantage of this method consists in applying techniques based on special tree structures to mine maximal patterns rather than using the *Apriori*-like algorithm to increase the collocation size iteratively. However, one can be interested also in characteristics of non-maximal patterns in the process of knowledge discovery. The problem of limited memory has been addressed in [4] [5] where the efficient method for performing joinless algorithm has been proposed. Nonetheless, there is still need to perform additional step of filtering candidate instances.

2 Definitions and Problem Formulation

2.1 Basic Definitions

Definition 1. *Let f be a spatial feature. An object x is an* ***instance*** *of the feature f, if x is a type of f and is described by a location and unique identifier.*

Definition 2. *Let F be a set of spatial features and S be a set of their instances. Given a neighbor relation R, we say that the* ***collocation*** *C is a subset of spatial features $C \subseteq F$ whose instances $I \subseteq S$ form a clique with respect to the relation R.*

Definition 3. *The* ***participation ratio*** *$Pr(C, f_i)$ of a feature f_i in the collocation $C = \{f_1, f_2, \ldots, f_k\}$ is a fraction of objects representing the feature f_i in the neighborhood of instances of collocation $C - \{f_i\}$*

$$Pr(C, f_i) = \frac{\textit{Number of distincs objects of } f_i \textit{ in instances of } C}{\textit{Number of all objects of } f_i}.$$

Definition 4. *The* ***participation index*** *$Pi(C)$ of a collocation $C = \{f_1, f_2, \ldots, f_k\}$ is defined as $Pi(C) = \min_{f_i \in C}\{Pr(C, f_i)\}$.*

Lemma 1. *The participation ratio and participation index are monotonically non-increasing with increases in the collocation size.*

Definition 5. *Given a spatial object $o_i \in S$ whose feature type is $f_i \in F$ the* ***star neighborhood*** *of o_i is defined as a set of spatial objects:*

$$T = \{o_j \in S \parallel o_i = o_j \vee (f_i < f_j \wedge R(o_i, o_j))\}$$

where $f_i \in F$ is the feature type of o_j and R is a neighbor relationship.

Definition 6. *Let $I = \{o_1, \ldots, o_k\} \subseteq S$ be a set of spatial objects whose feature types $\{f_1, f_2, \ldots, f_k\}$ are different. If all objects in I are neighbors to the first object o_1, I is called a* ***star instance*** *of collocation $C = \{f_1, f_2, \ldots, f_k\}$.*

2.2 Problem Formulation

The collocation pattern mining is defined as follows. Given a set of spatial features $F = \{f_1, f_2, \ldots, f_k\}$ and a set of their instances $S = S_1 \cup S_2 \cup \ldots \cup S_n$ where $S\,(1 \leq i \leq n)$ is a set of instances of feature $f_i \in F$ and each instance that belongs to S contains information about its feature type, instance id and location; a neighbor relationship R over locations; a minimum prevalence threshold (*min_prev*) and minimum conditional probability threshold (*min_cond_prob*); a size of the available memory, find efficiently (with respect to the memory constraint) a correct and complete set of collocation rules with participation index $\geq$ *min_prev* and conditional probability $\geq$ *min_cond_prob*. We assume that relation R is a distance metric based neighbor relationship with a symmetric property and spatial dataset is a point dataset.

3 Materialized Improved Candidate Pattern Tree

A general approach to the collocation mining problem consists in three major steps that are executed iteratively: (1) generating a set of candidate collocations, (2) identifying candidate collocation instances, (3) accepting or rejecting candidates on the basis of their prevalance measure. These steps are repeated untill there are no new candidates for collocations.

The first step applies the *Apriori* strategy (with prunnig based on the monotonicity property). Due to the limited number of spatial features (a few dozens at most), this step is not computationally demanding as well as computing their prevalences (the third step). However, to calculate this measure, knowledge of all candidate instances is required. It is the most challenging part of the algorithm and can be executed in a variety of ways. In the next section, one of the most popular method is briefly described.

3.1 State of the Art Solutions

An efficient method for collocation discovery, called *joinless*, has been proposed in [15]. In this algorithm a concept of a *star neighborhood* has been introduced. In the beginning spatial data are transformed into the set of *star neighborhoods* and further processing is based only on that structure. In each iteration candidate collocation instances are constructed from *star neighborhoods* and then filtered out. Lack of expensive spatial join results in high efficiency of the algorithm. In [5], an algorithm based on the *joinless* method, called *BCM*, has been proposed. It uses specially designed structure and operations to cope with limited memory. However, in both algorithms, generating instances directly from star neighborhoods can lead to incorrect collocation instances that do not form a clique in the neighborhood graph. This imposes an additional step of filtering such instances. In [12], the authors addressed this problem and proposed a new structure called *iCPI-tree (improved Collocation Pattern Instance Tree)* that holds information about neighborhoods. The step of generating collocation instances is performed by recursively traversing the tree. A short description of this method is as follows.

Step 1. *Convert a spatial dataset to a set of spatial ordered neighbor relationships between instances* - in this step, star neighborhoods are created. Each star has a central object, i.e. an object which has a neighbor relationship with all other objects. After sorting (with respect to the spatial feature), star neighborhoods form a set of ordered spatial neighbor relationships between instances.

Step 2. *Generate the iCPI-tree of the set of spatial ordered neighbor relationships* - from the set of ordered spatial neighbor relationships, the *iCPI-tree* is iteratively created. For example, consider a star neighborhood instance $A2$, $B1$, $B3$, $C1$, $C3$ in Fig. 1. A neighbor relationship between $A2$ and the remaining elements is represented as a branch in the *iCPI-tree*. Sub-branches B and C determine neighbors with appropriate spatial feature ($B1$, $B3$ and $C1$, $C3$).

Step 3. *Iteratively mining collocation rules* - this step consists of: applying the *Apriori* strategy to generate candidate collocations, searching for their instances in the *iCPI-tree* and filtering candidate with respect to the minimal prevalence threshold. All one-element instances are considered as collocations with prevalence equal to 1. This iterative step lasts as long as there new candidates from *apriori_gen* method.

Step 4. *Generate collocation rules* - this step can be executed after each iteration in the third step or at the end of the algorithm. Collocations rules are generated from the discovered collocation patterns.

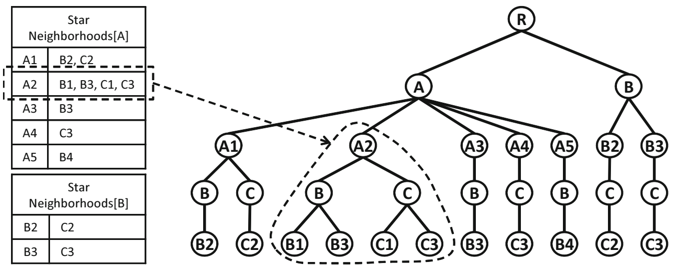
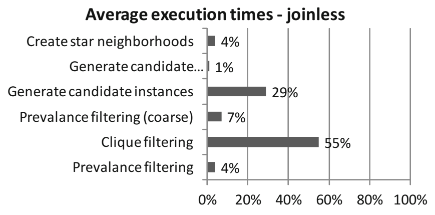


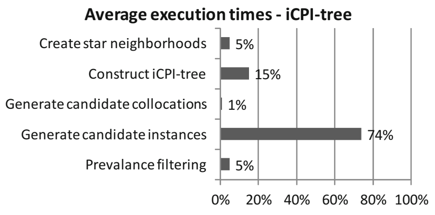
Fig. 1. Sample structure of the *iCPI-tree* (*Improved Candidate Pattern Instance Tree*)

3.2 Potential Bottlenecks in Joinless and iCPI-Tree Methods

For independent evaluation of the efficiency of current state of the art algorithms, we have implemented and tested both, the *joinless* and the *iCPI-tree* methods.



(a)



(b)

Fig. 2. Performance of the *joinless* and the *iCPI-tree* methods

The acquired results are shown in Fig. 2. Our attention has been focused on the performance of each part of the algorithms as well as their efficiency in the limited memory environment.

As we can see, in the original *joinless* method, two steps have major influence on the total processing time: *generating star instances* and *cliques filtering*. In contrast, in the *iCPI-tree* method, most of the time the algorithm spends on the step *generate candidate instances*. This can be explained in the following manner. In both methods, the first step is performed only once and there are efficient methods to create star neighborhoods. In our implementation, we used a *plane sweep* algorithm [6], which requires the data to be sorted (with respect to one of the dimensions in two-dimensional input dataset). Both, the sorting and the *plane sweep*, read entire dataset once. After this operation, in the *iCPI-tree* method, star neighborhoods are transformed into a tree (step 2). Candidate collocations are generated using the *Apriori* strategy. The number of possible combinations, unlike the size of such sets in the market-basket analysis, is very small. Typically, the number of spatial features is limited to several dozens, therefore this step has a minor influence on the total processing time. The biggest discrepancy is between times of generating candidate instances. While it is seemingly quite effortless step in the *joinless* method, the *iCPI-tree* algorithm spends almost three quarters of its processing time on this step. However, due to possible invalid instances (i.e. those that do not hold clique property) computationally demanding step of filtering cliques is applied before the prevalence filtering in the *joinless* method.

The second series of experiments have been conducted in order to observe, how the limited memory can decrease the efficiency of the aforementioned methods. In addition, we put results for the aforementioned *BCM* algorithm. Results are shown in Fig. 3. For the original *joinless* and *iCPI-tree* algorithms, we can see dramatic increase of the processing time. This is caused by the operating system preventing insufficient memory errors by a technique well known as *page swapping*. With the limited memory, it leads to a constant state of paging and therefore it results in a clear degradation of the execution time. Due to the huge number of invalid instances in the *joinless* algorithm, it reaches the memory limit much faster than the *iCPI-tree* method. Although the *iCPI-tree* algorithm performs better, finally the growing tree structure cannot be held in the system memory and efficiency falls down.

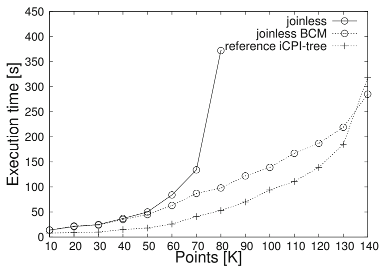


Fig. 3. Comparison of the performance of *joinless*, *BCM* and *iCPI-tree* methods

3.3 Materialization of the iCPI-Tree and a New Search Procedure

The following section presents a new method for processing collocation pattern queries in limited memory environment. A materialization step and an optimized tree search procedure are introduced.

During our tests we have observed that *iCPI-trees* can reach sizes bigger than the amount of available memory. The *iCPI-tree* structure is based on the star neighborhoods, which can also be bigger than the underlying database containing spatial objects. Moreover, the *iCPI-tree* requires huge number of pointers to maintain easy access to neighbors. Finally, the conclusion is that this structure requires more space than the corresponding set of star neighborhoods. There are many ways of representing tree-like structures on the disk. We propose to convert *iCPI-trees* into a two sets of objects stored (only when required) on disk. The first type of objects consists of a key that contains information about the object identifier and the feature type of particular neighbors. This type of key can be obtained e.g. by applying some hashing techniques. Additionally, each such element has a generated (and unique) pointer to the object representing one of the element's neighbors. The second type of objects consists of a key, which is a generated pointer and of a piece of information about the current and the next neighbor. Both types of objects share the same cache memory having size determined by the amount of the available system memory. In the ideal circumstances, the cache memory can hold all objects and no I/O transfer is required. When the memory size is not sufficient, there is necessity to materialize some nodes on the disk, to ensure that they can be retrieved if required. The best possible hit ratio (i.e. number of reads from memory in proportion to number of read from disk) is 100%, however, in a limited memory environment it can be unreachable. Although, we can more effectively design the step of generating candidate instances and therefore increase the hit ratio.

In Fig. 4(a) the original method for generating collocation instances has been shown. Let's assume that there is a candidate $\{A, B, C\}$. Instances for this candidate can be constructed from instances of the collocation $\{A, B\}$ discovered in the previous iteration of the algorithm. For each instance of the collocation $\{A, B\}$ we are searching for neighbors with feature C. If the same object is found for each element of particular $\{A, B\}$ instance, then a new candidate instance

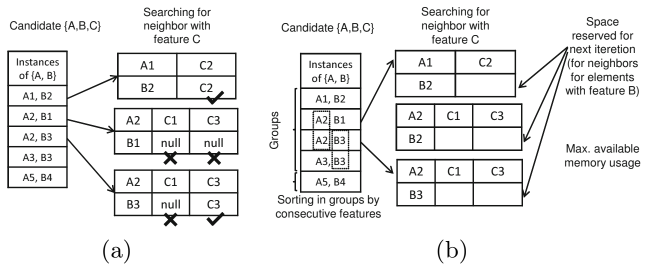

Candidate {A,B,C}
Searching for neighbor with feature C
Instances of {A, B}
A1, B2
A2, B1
A2, B3
A3, B3
A5, B4
A1 C2
B2 C2
A2 C1 C3
B1 null null
A2 C1 C3
B3 null C3
(a)
Candidate {A,B,C}
Searching for neighbor with feature C
Space reserved for next iteretion (for neighbors for elements with feature B)
Groups
Sorting in groups by consecutive features
Max. available memory usage
A1 C2
B2
A2 C1 C3
B2
A2 C1 C3
B3
(b)

Fig. 4. Alternative *materialized iCPI-tree* search procedure

can be added to the result set. For example, using the instance $\{A2, B3\}$ we can find two neighbors with feature C: $C1$ and $C3$, however only $C3$ is a common neighbor. Therefore instance $\{A2, B3, C3\}$ is valid, while $\{A2, B3, C1\}$ does not hold a clique property.

Assume that k is the size of the candidate *cand* and *col* is the collocation equal to *cand* with the last feature removed. The simplified pseudo code for our new search procedure is presented in Alg. 1. The procedure works as follows.

Algorithm 1 Search method for the limited memory environment

```
1:  procedure SEARCH($T$,$can$,$MEMSIZE$)
2:      $res = \emptyset$
3:      $col$=subset($can$, 0, length($can$)-2)
4:      $lff$=getFeature($can$[length($can$)-1])
5:      while (nextInstanceExists($col$)) do
6:          $G = \emptyset$; $uMem := 0$;
7:          while ($MEMSIZE < uMem$) do
8:              $colInst$ = readNextInst($col$)
9:              $n1st$ =searchT($T$, $colInst$[0],$lff$)
10:             $uMem$=$uMem$+size($n1st$)
11:             $G = G \cup \{colInst, ne1st\}$
12:         end while
13:         $res = res \cup$ prGroup($can$, $T$,$G$, $lff$)
14:     end while
15:     return $res$
16: end procedure

17: procedure PRGROUP($can$,$T$,$G$,$lff$)
18:     $x = 1$ // start from 2nd feature
19:     while $i$ <length(can)-1) do
20:         sort $G$ wrt $x - th$
21:         feature of $colInst$;
22:         for each $< colInst, cne >$ in $G$ do
23:             $ne = searchT(T, colInst[x]), lff)$
24:             $neigh = intersect(cne, ne)$
25:         end for
26:         $x = x + 1$
27:     end while
28:     return not null elements from G
29: end procedure
```

Step 1. Read instances (lines 6-11) - discovered in the previous iteration - of the *col* as long as it is possible to store them in the system memory. The upper limit of the number of reads can be computed at runtime. It depends on the number of neighbors of the first element with the desired feature (lines 9 and 10). During this procedure simultaneously a group G is created by storing pairs: instance of *col*, neighbors of the first element of *col*.

Step 2. When memory is full, process the group (line 13): start with ordering the set by *x-th* feature of the candidate (line 20).

Step 3. Search for neighbors of the *x-th* element of the instances preserved in the memory. Keep only common neighbors of previous elements (lines 23-24).

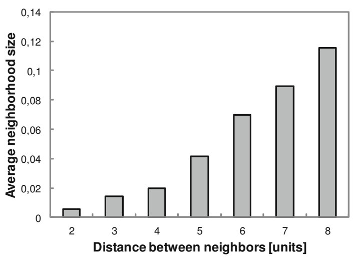
Step 4. $x = x + 1$ (line 26), perform steps 2 and 3 while $x < k$ (line 19).

Step 5. Perform steps 1 to 4 while there is no more collocation instances (line 5).

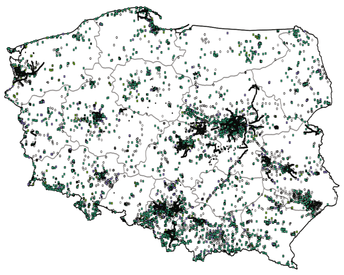
Using the aforementioned procedure, we are expecting to better utilize the memory cache that contains *iCPI-tree* elements. The desired improvement is based on the observation that, in many cases, elements in collocation/candidate instances can recur and therefore the cache hit ratio can increase. For example, in Fig. 4(b), neighbors for objects $A2$ and $B3$ are retrieved twice from the *iCPI-tree*.

4 Experimental Results

In order to validate our improvements we performed a series of experiments using both synthetic and real world datasets. Our experiments have been performed on a Linux workstation equipped with 1800MHz Athlon CPU, 2GB RAM and 80GB 7200RPM HDD. Synthetic datasets were generated using a procedure described in [15]. Parameters for experiments: the number of spatial features [15-50], the minimum prevalence threshold: [25%-40%], and the neighbor distance [10-20].



(a)



(b)

Fig. 5. The characteristic of real world dataset

Real world datasets are based on the spatial data acquired from the OpenStreetMap Project [1]. In the end of 2011 number of spatial data points, provided voluntarily by individu-als, exceeded 1.3 billion. We have processed this data and filtered out potentially interesting locations based on the user-provided tags. Due to the lack of the imposed set of possible spatial features we had to categorize them. Eventually, we have obtained 80 top-level spatial features such as leisure, sport, education etc. We have collected over 120 million of points with aforementioned tags. On account of the limited time for experiments, the subset of data used for algorithm evaluation consists of approximately 100K points located in Poland. Parameters for experiments: number of spatial features: 20; distance threshold: [2-8] units; prevalence threshold [0.25-0.55].

In Fig. 5(a), a density of the neighbors is shown for the distances used in the experiments. For clarity, each distance on the chart is expressed in units and 1 unit is equivalent to approx. 111 meters. The density should be understood as an average number of neighbors in the area of size d on the whole map. Fig. 5(b) presents visualization of several features on the map of Poland.

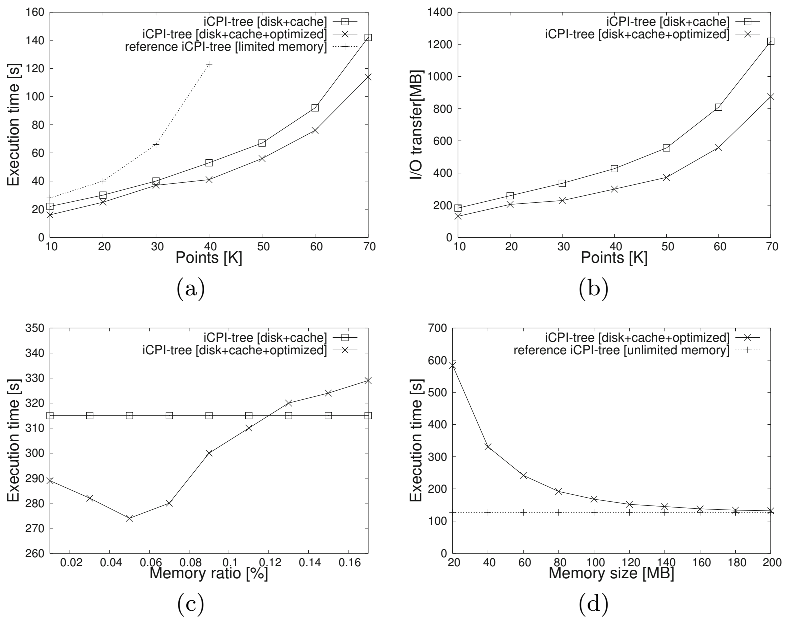
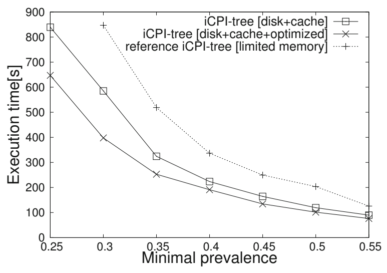


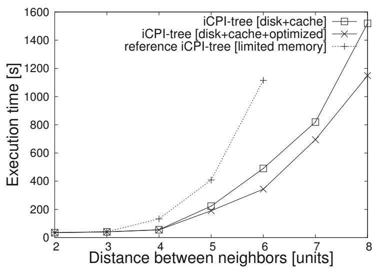
Fig. 6. The performance of the algorithms over synthetic datasets

In the first series of the experiments, we examined the processing time of three different versions of the algorithm based on the *iCPI-tree*: the original (memory based) and two implementations of our materialized *iCPI-tree* - using and not using the modified *gen_candidate_instances* method, respectively. The results presented in Fig. 6(a) are averages from 25 tests executed on the synthetic datasets. The size of the available memory was ranging from 10% to 100% of the required size. When no special structures to handle limited resources were used, the performance of the algorithm decreased substantially. The execution times, as well as I/O disk transfer (Fig. 6(b)), were better while using our improved version of the algorithm. The performance gap was increasing with the growing size of the input datasets.

In the second series of the conducted experiments, we examined how the memory limit threshold will affect the performance of the algorithm. Figure 6(c) presents the influence of the memory ratio parameter, which is the percentage of the available memory used by algorithm to store groups of candidate instances in the optimized version of the search method. The results indicate that this



(a)



(b)

Fig. 7. The performance of the algorithms over real world datasets

value should be quite small and that too much memory for storing the groups will decrease efficiency. It arises from smaller *iCPI-tree* cache size. Figure 6(d) presents how the execution time of the algorithm changes with increasing memory. We can observe that the performance is rapidly improving for lower memory sizes and finally it asymptotically approaches the optimal result.

Our final experiments were conducted on the real world dataset. We have compared solutions proposed in this work against the original algorithm. Similarly to experiments with synthetic datasets, the results presented in Fig. 7(a) and Fig. 7(b) are total averages from 25 tests, in which the available memory size was ranging from 20% to 100% of the required size. In Fig. 7(a) the influence of the prevalence threshold has been investigated, while Fig. 7(b) shows how the maximum distance between neighbors affects performance. In the first case, increasing prevalence threshold results in lower number of possible candidates, which translates into lower execution times. In spite of limited resources our method performs better and the performance gain is quite stable across whole experiment. In case of changing the distance parameter, the performance gap increases dramatically in favor of new, proposed methods. It arises from the fact that with higher distance threshold, the algorithm will generate larger neighborhoods and this directly leads up to substantially larger size of the *iCPI-tree*.

5 Summary

In this paper, we have identified the problem of collocation mining in the limited memory environment. We have shown that increasing the size of spatial databases imposes new constraints on algorithm design. The size of the internal data structures utilized by the algorithms can be a potential bottleneck, especially when the size of the input data is significantly bigger than the size of the available memory. We have presented that materialization and a dedicated method for collocation pattern instance discovery can result in better efficiency of the *iCPI-tree* algorithm in terms of overall execution time. The results of the experiments, both on synthetic and real world datasets, show that our method performs much better than the original solution. In the future, we are going to focus on the physical memory restrictions in the context of multiple, concurent spatial data mining tasks.

References

1. Openstreetmap, http://www.openstreetmap.org
2. Agrawal, R., Imieliński, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. SIGMOD Rec. 22(2), 207–216 (1993)
3. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (1994)
4. Boinski, P., Zakrzewicz, M.: Hash Join Based Spatial Collocation Pattern Mining. Foundations of Computing and Decision Sciences 36(1), 3–15 (2011)
5. Boinski, P., Zakrzewicz, M.: Memory Constraints in the Collocation Pattern Mining. Tech. rep., Poznan University of Technology (2012)
6. Graf, T., Hinrichs, K.: A Plane-Sweep Algorithm for the All-Nearest-Neighbors Problem for a Set of Convex Planar Objects. In: Dehne, F., Sack, J.-R., Santoro, N. (eds.) WADS 1993. LNCS, vol. 709, pp. 349–360. Springer, Heidelberg (1993)
7. Huang, Y., Shekhar, S., Xiong, H.: Discovering Co-location Patterns from Spatial Datasets: A General Approach. IEEE Transactions on Knowledge and Data Engineering 16, 472–485 (2004)
8. Koperski, K., Han, J.: Discovery of Spatial Association Rules in Geographic Information Databases. In: Egenhofer, M., Herring, J.R. (eds.) SSD 1995. LNCS, vol. 951, pp. 47–66. Springer, Heidelberg (1995)
9. Nanopoulos, A., Manolopoulos, Y.: Memory-Adative Association Rules Mining. Inf. Systems 29(5), 365–384 (2004)
10. Roddick, J.F., Spiliopoulou, M.: A Bibliography of Temporal, Spatial and Spatio-Temporal data Mining Research. SIGKDD Explor. Newsl. 1(1), 34–38 (1999)
11. Shekhar, S., Huang, Y.: Discovering Spatial Co-location Patterns: A Summary of Results. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 236–256. Springer, Heidelberg (2001)
12. Wang, L., Bao, Y., Lu, J.: Efficient Discovery of Spatial Co-Location Patterns Using the iCPI-tree. The Open Information Systems Journal 3(2), 69–80 (2009)
13. Wang, L., Zhou, L., Lu, J., Yip, J.: An Order-clique-based Approach for Mining Maximal Co-locations. Inf. Sci. 179(19), 3370–3382 (2009)
14. Yoo, J.S., Shekhar, S.: A Partial Join Approach for Mining Co-location Patterns. In: Pfoser, D., Cruz, I.F., Ronthaler, M. (eds.) GIS, pp. 241–249. ACM (2004)
15. Yoo, J.S., Shekhar, S., Celik, M.: A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results. In: Proceedings of the IEEE International Conference on Data Mining, pp. 813–816. IEEE Computer Society, Washington (2005)

Mining Popular Patterns from Transactional Databases

Carson Kai-Sang Leung* and Syed K. Tanbeer

University of Manitoba, Winnipeg, MB, Canada
{kleung,tanbeer}@cs.umanitoba.ca

Abstract. Since the introduction of the frequent pattern mining problem, researchers have extended frequent patterns to different useful patterns such as cyclic, emerging, periodic and regular patterns. In this paper, we introduce *popular patterns*, which captures the popularity of individuals, items, or events among their peers or groups. Moreover, we also propose (i) the *Pop-tree* structure to capture the essential information for the mining of popular patterns and (ii) the *Pop-growth* algorithm for mining popular patterns. Experimental results showed that our proposed tree structure is compact and space efficient and our proposed algorithm is time efficient.

Keywords: Data mining, knowledge discovery, interesting patterns, popular patterns, useful patterns, tree-based mining.

1 Introduction and Related Work

Since the introduction of the research problem of frequent pattern mining, numerous works have been proposed. These works can mostly be classified into two broad "categories". Works in the first "category" mainly focused on algorithmic efficiency [7,8,10]. For example, to avoid the candidate generation-and-test approach of the Apriori algorithm [1], a tree-based algorithm called FP-growth [4] was proposed to build an FP-tree to capture the contents of transactional database (TDB) so that frequent patterns can be mined recursively from the FP-tree with a restricted test-only approach.

Works in the second "category" mainly focused on extending the notion of frequent patterns to other interesting or useful patterns such as sequences, episodes, maximal and closed sets. However, the mining of these patterns are based on the support/frequency measure. While support/frequency is a useful metric, support-based frequent pattern mining may *not* be sufficient to discover many interesting knowledge (e.g., correlation, regularity, periodicity, popularity) among patterns in a TDB. This leads to the introduction of some interestingness measures [14] and their corresponding patterns such as emerging patterns [2], constrained patterns [5,9], correlated patterns [6], periodic patterns [12,17], regular patterns [13], hyperclique patterns [15], and high utility patterns [16].

* Corresponding author.

In many real-life situations, users want to find popular patterns. For example, a social analyst may want to find persons with large "groups" of friends in social networks as these persons can be the most influential one in their groups or the social networks [3,11]. Similarly, a new member may want to know individuals with high connectivity so that he can get to know more members quickly. A recommender may want to know researchers with large numbers of collaborators. As the fourth example, an event promoter may want to find events with large numbers of participants. With the increase in usage of social network media, it has become more important to be able to find popular individuals (or items, objects, events). Our **key contributions** of this paper include the following:

1. our introduction of notion of *popular patterns*;
2. our proposal of the *Pop-tree*, which is a tree structure to capture essential information about the popularity of individuals, items, objects, or events; and
3. our design and development of the *Pop-growth algorithm*, which mines popular patterns from the Pop-tree.

The remainder of this paper is organized as follows. We introduce popular patterns in the next section. In Section 3, we propose (i) the Pop-tree structure that captures important contents of the TDB and (ii) the Pop-growth algorithm that constructs the Pop-tree, from which popular patterns can be mined recursively. Experimental results are presented in Section 4. Finally, conclusions are provided in Section 5.

2 Notion of Popular Patterns

Let Item=$\{x_1, x_2, \ldots, x_m\}$ be a set of m domain items. A transactional database (TDB) is the set of n transactions: $\{t_1, t_2, \ldots, t_n\}$, where each transaction t_j in the TDB is a subset of Item. We use $|t_j|$ to represent the transaction length of t_j. Let $X = \{x_1, x_2, \ldots, x_k\} \subseteq$ Item be a pattern consisting of k items (i.e., a k-itemset), where $|X| = k \leq m$. Then, The projected database of X (denoted as DB_X) is a set of TDB transactions (in the TDB) that contain X. We use $maxTL(X)$ and $sumTL(X)$ to respectively represent the maximum length and the total length of all transactions in DB_X.

Table 1. A transaction database

Transaction ID	Transaction
t_1	$\{b, d\}$
t_2	$\{b, c, f, g, h\}$
t_3	$\{b, c, d, e, f, h\}$
t_4	$\{c, e, g, h\}$
t_5	$\{a, d\}$
t_6	$\{a, b, i\}$
t_7	$\{a, d, e\}$

Example 1. Consider the TDB shown in Table 1, which consists of n=7 transactions and m=9 domain items $a, b, \ldots, i$. For pattern $X = \{b, c\}$, its projected database $DB_{\{b,c\}}$=$\{t_2, t_3\}$. Hence, $|DB_{\{b,c\}}| = 2$, $|t_2| = |\{b, c, f, g, h\}| = 5$, $|t_3| = |\{b, c, d, e, f, h\}| = 6$, $maxTL(\{b, c\}) = \max\{|t_2|, |t_3|\} = \max\{5, 6\} = 6$, and $sumTL(\{b, c\}) = |t_2| + |t_3| = 5 + 6 = 11$. □

Definition 1. *The* **transaction popularity** $\boldsymbol{Pop(X, t_j)}$ *of a pattern X in transaction t_j measures the membership degree of X in t_j. For simplicity, we compute the membership degree based on the difference between the transaction length $|t_j|$ and the pattern size $|X|$:*

$$Pop(X, t_j) = |t_j| - |X|. \tag{1}$$

Note that, depending on real-life applications, the above equation can be adapted to incorporate some other functional operations on t_j and X.

Definition 2. *The* **long transaction popularity** $\boldsymbol{Pop(X, t_{maxTL(X)})}$ *of a pattern X in transaction $t_{maxTL(X)}$ measures the membership degree of X in t_{maxTL}, where $t_{maxTL(X)}$ is the transaction having the maximum length in DB_X:*

$$Pop(X, t_{maxTL(X)}) = \left(\max_{t_j \in DB_X} |t_j| \right) - |X|. \tag{2}$$

Definition 3. *The* **popularity** $\boldsymbol{Pop(X)}$ *of a pattern X in the TDB measures an aggregated membership degree of X in all transactions in the TDB. It is defined as an* average *of all transaction popularities of X:*

$$Pop(X) = \frac{1}{|DB_X|} \sum_{t_j \in DB_X} Pop(X, t_j). \tag{3}$$

Example 2. Reconsider the TDB shown in Table 1. The transaction popularity of pattern $\{b, c\}$ in t_2 can be computed as $Pop(\{b, c\}, t_2) = |t_2| - |\{b, c\}| = 5 - 2 = 3$. Similarly, $Pop(\{b, c\}, t_3) = |t_3| - |\{b, c\}| = 6 - 2 = 4$. Recall from Example 1 that $DB_{\{b,c\}}$=$\{t_2, t_3\}$ (i.e., $\{b, c\}$ appears only in t_2 and t_3). As t_3 is the longest transaction in $DB_{\{b,c\}}$ (because $maxTL(\{b, c\})$=6), the long transaction popularity of pattern $\{b, c\}$ in $t_{maxTL(\{b,c\})}$ can be computed as $Pop(\{b, c\}, t_{maxTL(\{b,c\})}) = \max\{|t_2|, |t_3|\} - |\{b, c\}| = 6 - 2 = 4$. Hence, the popularity of pattern $\{b, c\}$ is $\frac{1}{|DB_{\{b,c\}}|}(Pop(\{b, c\}, t_2) + Pop(\{b, c\}, t_3)) = \frac{1}{2}(3{+}4) = 3.5$ □

Definition 4. *Given a user specified minimum popularity threshold minpop, a pattern X is considered* **popular** *if its popularity is at least minpop (i.e., $Pop(X) \geq minpop$).*

Example 3. If the user specified *minpop* is 3.3, then pattern $\{b, c\}$ is popular in the TDB shown in Table 1 because $Pop(\{b, c\})$=3.5 $\geq$ 3.3=*minpop*. However, pattern $\{b\}$ is *not* popular because $Pop(\{b\}) = \frac{1}{|DB_{\{b\}}|}(Pop(\{b\}, t_1) + Pop(\{b\}, t_2) + Pop(\{b\}, t_3) + Pop(\{b\}, t_6)) = \frac{1}{4}(1 + 4 + 5 + 2) = 3 < 3.3 = minpop$. □

3 Pop-Growth: Mining Popular Patterns with a Pop-Tree

When mining frequent patterns, the frequency measure satisfies the downward closure property (i.e., if a pattern is infrequent, its superset is guaranteed to be infrequent). This helps reduce the search/solution space by pruning infrequent pattern, and thus speeds up the mining process. However, when mining popular patterns, observant readers may notice from Example 3 that popularity does *not* satisfy the downward closure property. For example, a pattern (e.g., $\{b\}$) is unpopular, but its superset (e.g., $\{b, c\}$) may be popular. Hence, the mining of popular patterns can be challenging.

To handle the challenge, let us revisit Equation (3) and redefine the popularity $Pop(X)$ of a pattern X (cf. Definition 3).

Definition 5. *The* **popularity** $\boldsymbol{Pop(X)}$ *of a pattern X in the TDB measures an aggregated membership degree of X in all transactions in the TDB. It is defined in terms of $sumTL(X) = \sum_{t_j \in DB_X} |t_j|$ as follows:*

$$\begin{aligned} Pop(X) &= \frac{1}{|DB_X|} \sum_{t_j \in DB_X} Pop(X, t_j) \\ &= \frac{1}{|DB_X|} \sum_{t_j \in DB_X} (|t_j| - |X|) \\ &= \frac{sumTL(X)}{|DB_X|} - |X|. \end{aligned} \tag{4}$$

Example 4. Reconsider the TDB shown in Table 1. Recall from Example 1 that $sumTL(\{b, c\})$=11. Then, the popularity of pattern $\{b, c\}$ is $\frac{sumTL(\{b,c\})}{|\{t_2,t_3\}|} - |\{b, c\}| = \frac{11}{2} - 2 = 3.5$ Similarly, the popularity of pattern $\{b\}$ is $\frac{sumTL(\{b\})}{|\{t_1,t_2,t_3,t_6\}|} - |\{b\}| = \frac{|t_1|+|t_2|+|t_3|+|t_6|}{|\{t_1,t_2,t_3,t_6\}|} - |\{b\}| = \frac{16}{4} - 1 = 3.$ □

Observant readers may notice from Example 4 that $sumTL(\{b, c\})$=11 $\leq$ 16= $sumTL(\{b\})$. The definition of $sumTL(X)$ further confirms that the total transaction length $sumTL(X)$ of X satisfies the downward closure property (i.e., $sumTL(X) \geq sumTL(X')$ if $X \subseteq X'$).

3.1 Construction of a Pop-Tree

To mine popular patterns, we propose the Pop-growth algorithm, which consists of two key procedures: (i) construction of a Pop-tree and (ii) mining of popular patterns from the Pop-tree.

We first build a tree structure—called **Popular pattern tree (Pop-tree)**—to capture the necessary information from the TDB with only two scans of the TDB. Recall from Section 2 that $Pop(X)$ does not satisfy the downward closure property. So, unpopular items need to be kept in the Pop-tree as some of their supersets may be popular. Fortunately, recall from Section 2 that $sumTL(X)$ satisfies the downward closure property. So, not all unpopular items need to be kept. Some of them can be pruned. See the following two lemmas.

Lemma 1. *The popularity of a pattern X is always less than or equal to its long transaction popularity, i.e., $Pop(X) \leq Pop(X, t_{maxTL(X)})$.*

Lemma 2. *For $X \subseteq X'$, $Pop(X')$ cannot exceed $maxTL(X) - |X'|$.*

Based on the above two lemmas, the following equation provides us with an upper bound of the popularity $Pop(X')$ of a pattern X' (in terms of $maxTL(X)$), where $X \subseteq X'$:

$$Pop^{UB}(X') = maxTL(X) - |X'|. \tag{5}$$

Based on Equation (5), we can calculate the popularity upper bound of a pattern X' from $maxTL(X)$ (where $X \subseteq X'$, and $|X'| = |X|+1 = k+1$), and prune unpopular patterns. We call this *super-pattern popularity check*.

Similar to FP-tree [4], each node of a Pop-tree contains the parent and child pointers as well as horizontal node traversal pointers. To facilitate popular pattern mining, we keep (i) an item x, (ii) support of $Y \cup \{x\}$, (iii) $sumTL(Y \cup \{x\})$, and (iv) $maxTL(Y \cup \{x\})$, where Y represents the set of items above x (i.e., ancestor nodes of x).

To construct a Pop-tree, we scan the TDB to find the $support(x)$, maximum transaction length $maxTL(x)$ and the popularity $Pop(x)$ for each singleton x in the TDB. Then, we perform the super-pattern popularity check and safely delete a pattern x if $Pop^{UB}(x') < minpop$ (where x' is an extension of x). We then scan the TDB the second time to insert each transaction into the Pop-tree in a similar fashion as the insertion process of FP-tree.

Example 5. Let us show how to construct a Pop-tree for the TDB shown in Table 1 with *minpop*=2.4. With the first database scan, we obtain the following information in the form of $\langle x$: $support(x)$, $maxTL(x)$, $Pop(x)\rangle$ for each of the m=9 domain items, i.e., $\langle a$:3,3,1.66$\rangle$, $\langle b$:4,6,3.0$\rangle$, $\langle c$:3,6,4.0$\rangle$, $\langle d$:4,6,2.25$\rangle$, $\langle e$:3,6,3.33$\rangle$, $\langle f$:2,6,4.5$\rangle$, $\langle g$:2,5,3.5$\rangle$, $\langle h$:3,6,4.0$\rangle$, $\langle i$:1,3,2.0$\rangle$. Note that all items—except a, d & i—are popular (i.e., with popularity at least 2.4). Although a, d & i are unpopular, their super-patterns may be popular. Hence, we cannot delete them without performing the super-pattern popularity check. The popularity upper bounds of the extensions of a, d & i are 3−2=1, 6−2=4 & 3−2=1, respectively. As the value for d is greater than *minpop*, we keep d but safely delete a & i. We sort and insert items b, c, d, e, f, g & h into a header table (H-table) in descending order of support: $\langle b, d, c, e, h, f, g\rangle$.

We then scan the TDB the second time. We compute the length of each transaction, remove all items that are not in the H-table, and sort the remaining items in each transaction according to the H-table order. Fig. 1(a) shows the contents of the H-table ($\langle x$: $support(x)$, $sumTL(x)$, $maxTL(x)\rangle$) and the Pop-tree structure after inserting t_1 of TDB. Because t_1 and t_2 share a common prefix (i.e., $\{b\}$), we increase the occurrence count of the common node $\{b : 1, 2, 2\}$ by one, its total transaction length ($sumTL$) by the transaction length of t_2 (i.e., $|t_2|$=5), and update its $maxTL$. For the remaining (uncommon) nodes of t_2, we set *support*=1, $sumTL$=$|t_2|$ and $maxTL$=$|t_2|$. The contents of the Pop-tree after insertion of t_2 are shown in Fig. 1(b). The final Pop-tree after capturing all the transactions in the TDB is shown in Fig. 1(c). □

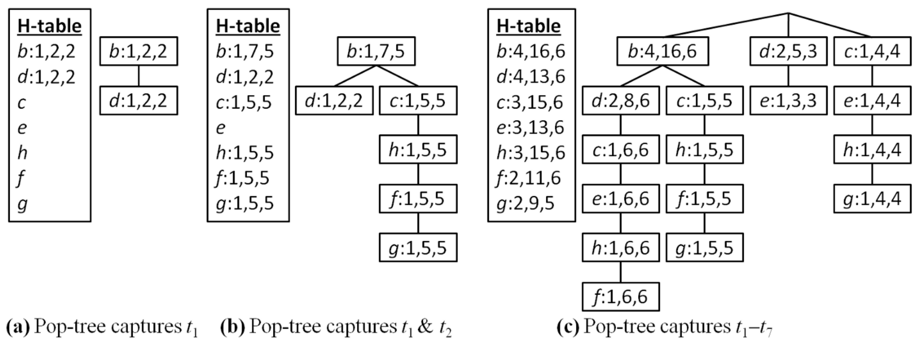


Fig. 1. The Pop-tree construction

Let $I(t_j)$ be the set of items in transaction t_j that pass through the first database scan. Based on the above Pop-tree construction procedure, we observed several important properties of Pop-trees listed as follows.

Property 1. A Pop-tree registers the projection of $I(t_j)$ for t_j in the TDB only once.

Property 2. The total transaction length *sumTL* in a node x in a Pop-tree captures the sum of lengths of all transactions that pass through, or end at, the node for all the nodes in the path from x up to the root.

Property 3. The total transaction length *sumTL* of any node in a Pop-tree is greater than or equal to the sum of transaction lengths of its children.

Properties 2 and 3 are the result of sharing common prefixes by different transactions, which allow our Pop-tree to be compact. Based on following lemma, one can observe that a Pop-tree is a highly compact tree structure.

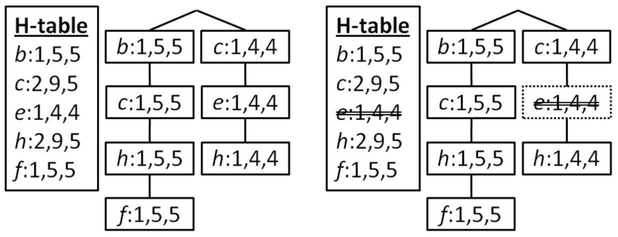
Lemma 3. *The size of a Pop-tree on a TDB for minpop is bounded above by* $\sum_{t_j \in \text{TDB}} |I(t_j)|$.

Lemma 4. *Given a TDB and minpop, the complete set of all popular patterns can be obtained from a Pop-tree for the minpop on the TDB.*

We can justify the completeness of a Pop-tree for mining popular patterns by Lemma 4. Based on this lemma, popular patterns can be found by mining our Pop-tree.

3.2 Mining of Popular Patterns from the Pop-Tree

Recall that, to mine popular patterns, the Pop-growth algorithm applies two key procedures: (i) construction of a Pop-tree and (ii) mining of popular patterns from the Pop-tree. The Pop-growth finds popular patterns from the Pop-tree,



(a) $\{g\}$-projected database **(b)** Cond'l pattern base of $\{g\}$

Pattern	Popularity
{*b*,*g*}	(5/1)–2 = 3
{*f*,*g*}	(5/1)–2 = 3
{*c*,*g*}	(9/2)–2 = 2.5
{*g*,*h*}	(9/2)–2 = 2.5

(c) Popular patterns with *g*

Fig. 2. The Pop-tree mining

in which each tree node captures its occurrence count, total transaction length, and maximum transaction length. The algorithm finds popular patterns by constructing the projected database for potential popular itemsets and recursively mining their extensions.

While constructing the conditional database from a projected database, we perform a super-pattern popularity check for extensions of any unpopular item, and delete the item only when it fails the check. We call such pruning technique the *lazy pruning*.

The lazy pruning technique ensures that no popular patterns (having unpopular subsets) will be missed by Pop-growth. The following example illustrates how Pop-growth mines popular patterns from the Pop-tree.

Example 6. Let us continue Example 5. In other words, let us mine popular patterns from the Pop-tree shown in Fig. 1(c) constructed for the TDB shown in Table 1 with *minpop*=2.4.

Recall that the Pop-growth recursively mines the projected databases of all items in H-table. Before constructing the projected database for an item x in H-table, we output the item as a popular pattern if its popularity is at least *minpop*. The conditional pattern base for the $\{g\}$-projected database (i.e., $DB_{\{g\}}$), as shown in Fig. 2(a), is constructed by accumulating the contents in the tree path $\langle b{:}1{,}5{,}5 \quad c{:}1{,}5{,}5 \quad h{:}1{,}5{,}5 \quad f{:}1{,}5{,}5\rangle$ and $\langle c{:}1{,}4{,}4 \quad e{:}1{,}4{,}4 \quad h{:}1{,}4{,}4\rangle$. The header table for $DB_{\{g\}}$, as shown in Fig. 2(a), contains all items that co-occur with g in the Pop-tree. It also contains the corresponding *support*, $sumTL$ and $maxTL$ of each item in $DB_{\{g\}}$. We then compute the exact popularity of each item in $DB_{\{g\}}$ by using Equation (4).

The conditional tree for any conditional pattern base of an itemset X may contain two types of items: (i) items that are popular in DB_X and (ii) items that are unpopular in DB_X but having potentially popular super-patterns. Other items are deleted from the projected database. To find unpopular items that having potentially popular super-patterns, we apply the lazy pruning technique and Equation (5).

Based on Equation (4), the popularity of items in the H-table of $DB_{\{g\}}$ can be computed: $Pop(\{b,g\}) = \frac{5}{1} - 2 = 3$, $Pop(\{f,g\}) = \frac{5}{1} - 2 = 3$, $Pop(\{c,g\}) = \frac{9}{2} - 2 = 2.5$, $Pop(\{g,h\}) = \frac{9}{2} - 2 = 2.5$ and $Pop(\{e,g\}) = \frac{4}{1} - 2 = 2$. All items except e are popular together with g. By applying the lazy pruning technique, the popularity upper bound $Pop^{UB}(\{e, g\})$ for e with g can be calculated as 4−2 = 2, which is less than *minpop*. Hence, we can safely delete e from the projected database of g. The conditional tree for the projected database of g is presented in Fig 2(b).

The mining for each extension (i.e., for f, h, c & b) of g is performed recursively. The set of patterns generated from the projected database of g is shown in Fig. 2(c). The mining process terminates when we reach the top of H-table of the Pop-tree. □

The Pop-growth mining technique is efficient because it applies a pattern-growth based mining technique on a Pop-tree. Moreover, the lazy pruning technique further reduces the mining cost for unpopular items whose super-patterns cannot be popular.

4 Experimental Results

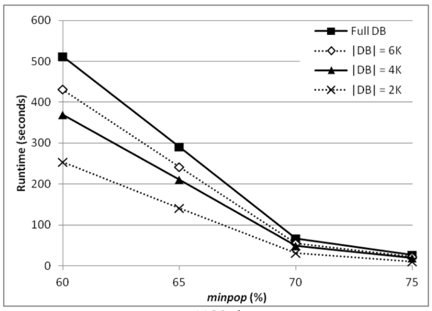
For experiments, we mostly use those datasets commonly used in frequent pattern mining experiments because characteristics of those transactional datasets are well known (see Table 2). More specially, we used (i) IBM synthetic datasets (e.g., T10I4D100K, T20I4D100K) from `www.almaden.ibm.com/cs/quest` and (ii) real datasets (e.g., chess, mushroom, connect-4) from the Frequent Itemset Mining Dataset Repository `fimi.cs.helsinki.fi/data`. We obtained consistent results for all of these datasets. Hence, due to space constraint, we report here the experimental results on only a subset of these datasets in the remainder of this section.

Table 2. Dataset characteristics

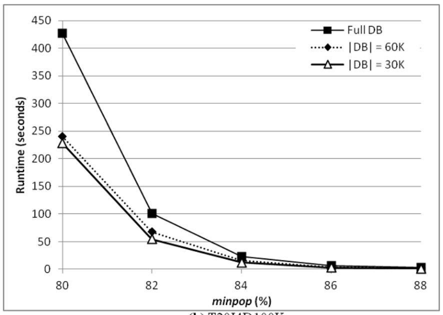
Dataset	#transactions	#items	maxTL	avgTL	Data density
T10I4D100K	100,000	870	29	10.10	Sparse
T20I4D100K	99,996	871	42	19.81	Sparse
mushroom	8,124	119	23	23.00	Dense

All programs were written in C and run on UNIX with a quad-core processor with 1.3 GHz. The runtime specified indicates the total execution time (i.e., CPU and I/Os). The reported results are based on the average of multiple runs for each case. In all of the below experiments, Pop-trees were constructed using descending order of occurrence counts of items.

To the best of our knowledge, our Pop-tree is the first approach to mine popular patterns from transactional databases. Here, we present the performance of our Pop-tree structure and Pop-growth algorithm when varying the mining parameters such as popularity threshold and dataset characteristics.



(a) Mushroom



(b) T20I4D100K

Fig. 3. Runtime

4.1 Runtime

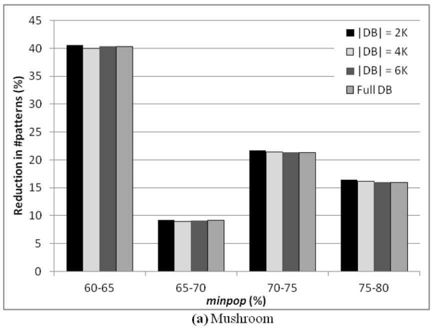
In this section, we report the execution time that the Pop-growth requires for mining popular patterns over datasets of different types and changes in *minpop*. The execution time includes all the steps of H-table construction, the Pop-tree building and the corresponding mining. The results on one sparse dataset (e.g., T20I4D100K) and one dense dataset (e.g., mushroom) are presented in Fig. 3.

To observe the effect of mining on the variation in size of such datasets, we performed popular pattern mining while increasing the size of both of the datasets: (i) From 2K to full for the mushroom dataset and (ii) from 30K to full for T20I4D100K. Thus, the series for "Full DB" represent the results for the full size of datasets. Both datasets required more execution time when mining larger datasets. As the database size increased and *minpop* decreased, the tree structure size and number of popular patterns increased. Hence, a comparatively longer time was required to generate large number of popular patterns from large trees. Although the mushroom dataset is smaller in size, the transaction lengths of all transactions are the same (i.e., 23). Hence, the Pop-tree mining took a longer time when compared to a dataset with variable length such as T20I4D100K. The experimental results show that the mining corresponding Pop-tree for popular patterns is time efficient for both sparse and dense datasets.

4.2 Reduction on the Number of Patterns When Changing *minpop*

Similar to the previous experiment, we also examined the number of patterns generated by our Pop-growth when we varied the dataset size and *minpop*. Fig. 4 shows the reduction in the number of patterns in percentage when increasing the *minpop* values in both the mushroom and T20I4D100K datasets with different dataset size. Each data point in the x-axes of the graphs reports the change of *minpop* from a low to a high value, while the y-axes indicate the percentage change in the number of patterns generated from a low to a high *minpop* value.

Note that, depending on dataset characteristics, the reduction rate varied. For example, for the mushroom dataset, the reduction rate dropped sharply when *minpop* was changed from 60%–65% to 65%–70%, but the reduction rate rose when *minpop* was changed to 70%–75%. In contrast, T20I4D100K showed a



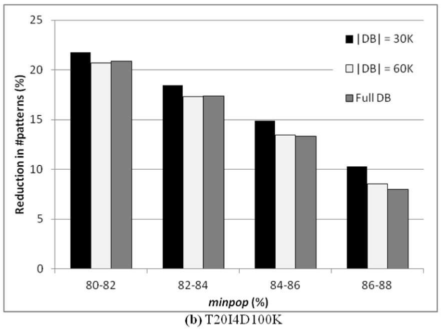


Fig. 4. Reduction on the number of patterns when changing *minpop*

consistent reduction rate when lowering the *minpop* value. However, as observed from the graphs for both datasets, the number of patterns reduced when increasing the *minpop* values. For example, for the mushroom dataset, the reduction rate was around 40% when increasing the threshold from 60% to 65%. For 30K of T20I4D100K, the reduction rate was around 21% when increasing the threshold from 80% to 82%. It is also interesting to note that the pattern count reduction rate was very similar irrespective of its different size.

We observed that the pattern generation characteristics of the proposed popular pattern mining algorithm were consistent with the variation of *minpop* and database size.

4.3 Compactness of the Pop-Tree

Here, we report the compactness of a Pop-tree in terms of number of Pop-tree nodes. Note that, as the mushroom dataset has a fixed transaction length, the maximum transaction length for every possible pattern in the dataset is always the same. Consequently, every item in the dataset passes the lazy pruning phase and contributes to the tree. Hence, for a particular portion of the mushroom dataset, the tree size (i.e., number of nodes) is the same with the variation of *minpop*. However, the number of nodes varied from 34523 (when |TDB| = 2K) to 91338 (when

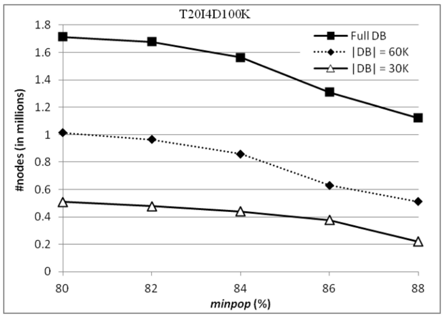


Fig. 5. Compactness of the Pop-tree: node count on T20I4D100K

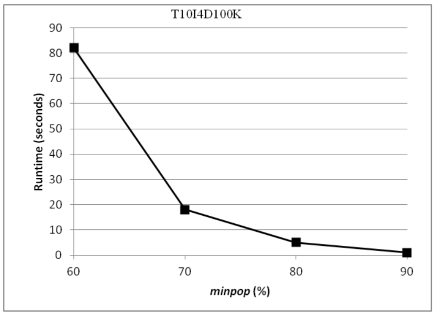
|TDB| = 6K). For the full dataset, it is around 100K. The compactness of Pop-tree on different portion of T20I4D100K is presented in Fig. 5. The size of the tree structure gradually reduced in T20I4D100K with the increase of *minpop*.

As expected, in both datasets, the number of nodes increased with the increase in size of database. However, as far as the total number of nodes is concerned, one can observe that, irrespective of fixed or variable transaction length, a Pop-tree structure is compact enough to fit into a reasonable amount of memory.

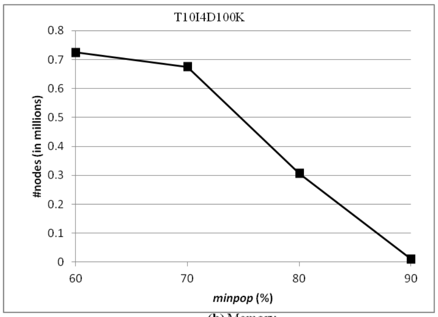
4.4 Scalability of Pop-Growth

To study the scalability of Pop-growth mining technique, we further ran our algorithm on T10I4D100K, which is sparser than T20I4D100K. Fig. 6 presents the results on scalability tests on the variation of *minpop* and required number of nodes on the dataset. Clearly, as the *minpop* decreases, the overall tree construction and mining time (Fig. 6(a)), and required memory (Fig. 6(b)) increase. However, the Pop-tree shows a stable performance with a linear increase in runtime and memory consumption as the *minpop* decreased for the dataset. Moreover, the results demonstrate that, the Pop-tree can mine the set of popular patterns on this dataset for a reasonably small value of popularity threshold with a considerable amount of execution time and memory.

To recap, the above experimental results show that the proposed Pop-tree can mine the set of popular patterns in both time and memory efficient manner over different types of dataset. Furthermore, the Pop-tree structure and the Pop-growth algorithm are scalable for popularity threshold values and memory.



(a) Runtime



(b) Memory

Fig. 6. Scalability on Pop-growth

5 Conclusions

In this paper, we introduced a new type of patterns—namely, *popular patterns*. We also proposed the *Pop-tree* (which captures important contents of transaction databases for mining popular patterns) and the *Pop-growth* algorithm (which finds popular patterns by mining the Pop-tree). Although the notion of popularity does not satisfy the downward closure property, we managed to address this issue by using the total transaction length ($sumTL$) together with projected databases, which allows lazy pruning. Experimental results showed that Pop-tree is compact, scalable, and space efficient for both sparse and dense datasets (e.g., IBM synthetic data and real data from FIMI). Moreover, results also showed that construction of Pop-tree and mining of popular patterns are time efficient.

Acknowledgement. This project is partially supported by NSERC (Canada) and University of Manitoba.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB 1994, pp. 487–499 (1994)
2. Bailey, J., Manoukian, T., Ramamohanarao, K.: Fast Algorithms for Mining Emerging Patterns. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 39–50. Springer, Heidelberg (2002)
3. Cameron, J.J., Leung, C.K.-S., Tanbeer, S.K.: Finding strong groups of friends among friends in social networks. In: IEEE DASC 2011, pp. 824–831 (2011)
4. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp. 1–12 (2000)
5. Lakshmanan, L.V.S., Leung, C.K.-S., Ng, R.T.: Efficient dynamic mining of constrained frequent sets. ACM TODS 28(4), 337–389 (2003)
6. Lee, Y.-K., Kim, W.-Y., Cai, Y.D., Han, J.: CoMine: efficient mining of correlated patterns. In: IEEE ICDM 2003, pp. 581–584 (2003)
7. Leung, C.K.-S., Hao, B.: Mining of frequent itemsets from streams of uncertain data. In: IEEE ICDE 2009, pp. 1663–1670 (2009)
8. Leung, C.K.-S., Jiang, F.: Frequent Pattern Mining from Time-Fading Streams of Uncertain Data. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 252–264. Springer, Heidelberg (2011)
9. Leung, C.K.-S., Sun, L.: A new class of constraints for constrained frequent pattern mining. In: ACM SAC 2012, pp. 199–204 (2012)
10. Leung, C.K.-S., Tanbeer, S.K.: Fast Tree-Based Mining of Frequent Itemsets from Uncertain Data. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 272–287. Springer, Heidelberg (2012)
11. Leung, C.K.-S., Tanbeer, S.K.: Mining Social Networks for Significant Friend Groups. In: Yu, H., Yu, G., Hsu, W., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA Workshops 2012. LNCS, vol. 7240, pp. 180–192. Springer, Heidelberg (2012)
12. Rasheed, F., Alshalalfa, M., Alhajj, R.: Efficient periodicity mining in time series databases using suffix trees. IEEE TKDE 23(1), 79–94 (2011)
13. Rashid, M. M., Karim, M. R., Jeong, B.-S., Choi, H.-J.: Efficient Mining Regularly Frequent Patterns in Transactional Databases. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 258–271. Springer, Heidelberg (2012)
14. Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework. Data Mining and Knowledge Discovery 21(3), 371–397 (2010)
15. Xiong, H., Tan, P.-N., Kumar, V.: Hyperclique pattern discovery. Data Mining and Knowledge Discovery 13(2), 219–242 (2006)
16. Yao, H., Hamilton, H.J.: Mining itemset utilities from transaction databases. DKE 59(3), 603–626 (2006)
17. Zhang, M., Kao, B., Cheung, D.W., Yip, K.Y.: Mining periodic patterns with gap requirement from sequences. ACM TKDD, 1(2), art. 7 (2007)

Rare Pattern Mining on Data Streams

David Huang, Yun Sing Koh, and Gillian Dobbie

Department of Computer Science, University of Auckland
dhua035@aucklanduni.ac.nz,
{ykoh,gill}@cs.auckland.ac.nz

Abstract. There has been some research in the area of rare pattern mining where the researchers try to capture patterns involving events that are unusual in a dataset. These patterns are considered more useful than frequent patterns in some domain, including detection of computer attacks, or fraudulent credit transactions. Until now, most of the research in this area concentrates only on finding rare rules in a static dataset. There is a proliferation of applications which generate data streams, such as network logs and banking transactions. Applying techniques for static datasets is not practical for data streams. In this paper we propose a novel approach called Streaming Rare Pattern Tree (SRP-Tree), which finds rare rules in a data stream environment using a sliding window, and show that it is faster than current approaches.

Keywords: Rare Pattern Mining, FP-Growth, Data Streams, Sliding Window.

1 Introduction

Traditionally pattern mining techniques focus on finding frequent patterns within a dataset. However in some scenarios rare patterns may be more interesting as they represent unexpected phenomenons. Rare patterns are patterns that do not occur frequently within the dataset and can be considered as exceptions. An example of a useful rare pattern in real life could be the association of certain occurences of symptoms to diseases. For instance, Meningitis is the inflammation of the protective membranes covering the brain and spinal cord. Symptoms of Meningitis include headache, fever, vomiting, neck stiffness, and altered consciousness. Most of the symptoms are commonly occuring for the influenza except for neck stiffness. By discovering the rare occurrence of neck stiffness, we are capable of flagging a patient possibly suffering from Meningitis out of a pool of patients suffering from the common influenza. In the recent years, the problem of extracting rare patterns from static datasets has been addressed. However as the capability to generate data streams increases, the ability to capture useful information from these data is necessary.

Ever since its inception, data stream mining has remained one of the more challenging problems within the data mining discipline. Data from a wide variety of application areas ranging from online retail applications such as online auctions and online bookstores, telecommunications call data, credit card transactions, sensor data and climate data are a few examples of applications that generate vast quantities of data on a continuous basis. Data produced by such applications are highly volatile with new patterns and trends emerging on a continuous basis. The unbounded size of data streams is considered the main obstacle to processing data streams. As it is unbounded, it makes it infeasible to store the entire data on disk. Furthermore, we would want to process data streams near real time. This raises two issues. Firstly, a multi-pass algorithm, entire dataset needs to be stored before mining can commence. Secondly, obtaining the exact set of rules that includes both frequent and rare rules from the data streams is too expensive.

To capture these types of rules, we propose a novel technique called Streaming Rare Pattern Tree (SRP-Tree), which requires a single pass through the dataset using a sliding window approach. We also propose a novel data structure called the Connection Table which allows us to efficiently constrain the search in our tree, and keep track of items within the window. The paper is organized as follows. In Section 2 we look at related work in the area of rare association rule mining. In Section 3 we present preliminary concepts and definitions for rare pattern mining in data streams. In Section 4 we describe our SRP-Tree approach, and in Section 5 we describe and discuss our experimental results. Finally, Section 6 concludes the paper.

2 Related Work

There has been a lot of work in the area of rare pattern mining. However all current research in this area is designed for static datasets and not able to handle a data stream environment. Currently there are two different types of rare pattern mining approaches: level-wise and tree based. Current rare itemset mining approaches which are based on level-wise exploration of the search space are similar to the Apriori algorithm [2]. In Apriori, k-itemsets (itemsets of cardinality k) are used to generate $k + 1$-itemsets. These new $k+1$-itemsets are pruned using the downward closure property, which states that the superset of a non-frequent itemset cannot be frequent. Apriori terminates when there are no new $k + 1$-itemsets remaining after pruning. MS-Apriori [12], Rarity [16], ARIMA [14], AfRIM [1] and Apriori-Inverse [8] are five algorithms that detect rare itemsets. They all use level-wise exploration similar to Apriori, which have candidate generation and pruning steps.

MS-Apriori [12] uses a bottom-up approach similar to Apriori. In MS-Apriori, each item can be assigned a different minimum item support value (MIS). Rare items can be assigned a low MIS, so that during candidate pruning, itemsets that include rare items are more likely to be retained and participate in rule generation. Apriori-Inverse [8] is used to mine perfectly rare itemsets, which are itemsets that only consist of items below a maximum support threshold (maxSup).

Szathmary et al. [14] proposed two algorithms that can be used together to mine rare itemsets: MRG-Exp and ARIMA. They defined three types of itemsets: minimal generators (MG), which are itemsets with a lower support than its subsets; minimal rare generators (MRG), which are itemsets with non-zero support and whose subsets are all frequent; and minimal zero generators (MZG), which are itemsets with zero support and whose subsets all have non-zero support. The first algorithm, MRG-Exp, finds all MRG by using MGs for candidate generation in each layer in a bottom up fashion. The MRGs represent a border that separates the frequent and rare itemsets in the search space.

All itemsets above this border must be rare according to the antimonotonic property. The second algorithm, ARIMA, uses these MRGs to generate the complete set of rare itemsets. ARIMA stops the search for non-zero rare itemsets when the MZG border is reached, which represents the border above which there are only zero rare itemsets.

Adda et al. [1] proposed AfRIM which begins with the itemset that contains all items found in the database. Candidate generation occurs by finding common k-itemset subsets between all combinations of rare $k+1$-itemset pairs in the previous level. Troiano et al. proposed the Rarity algorithm that begins by identifying the longest transaction within the database and uses it to perform a top-down search for rare itemsets, thereby avoiding the lower layers that contain only frequent itemsets.

All of the above algorithms use the fundamental generate-and-test approach used in Apriori, which has potentially expensive candidate generation and pruning steps. In addition, these algorithms attempt to identify all possible rare itemsets, and as a result require a significant amount of execution time. RP-Tree algorithm was proposed by Tsang et al. [17] as a solution to these issues. RP-Tree avoids the expensive itemset generation and pruning steps by using a tree data structure, based on FP-Tree [7], to find rare patterns. RP-Tree finds rare itemsets using a tree structure. However it uses a multi-pass approach, which is not suitable in a data stream environment.

There has been much work in the area of data stream mining, but most work focuses on capturing frequent patterns [4,5,9,3,10,11,13,15]. Up until now there has been no research into rare pattern mining in data streams.

3 Preliminaries

In this section, we provide definitions of key terms that explain the concepts of frequent pattern mining in a data stream. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a set of literals, called items, that represent a unit of information in an application domain. An set $X = \{i_l, \ldots, i_m\} \subseteq I$ and $l, m \in [1, n]$, is called a itemset, or a k-itemset if it contains k items. A transaction $t = (tid, Y)$ is a tuple where tid is a transaction-id and Y is a pattern. If $X \subseteq Y$, it is said that t contains X or X occurs in t. Let size(t) be the size of t, i.e., the number of items in Y.

A data stream $\mathcal{DS}$ can formally be defined as an infinite sequence of transactions, $\mathcal{DS} = [t_1, t_2, \ldots, t_m)$, where $t_i, i \in [1, m]$ is the ith transaction in data stream. A window $\mathcal{W}$ is a set of all transactions between the ith and jth (where $j > i$) transactions and the size of $\mathcal{W}$ is $|W| = j - i$. The count of a itemset X in a $\mathcal{W}$, denoted as $count_{\mathcal{W}}(X)$, is the number of transactions in $\mathcal{W}$ that contain X, and the support of an itemset X, denoted as $sup_{\mathcal{W}}(X) = \frac{count_{\mathcal{W}}(X)}{|\mathcal{W}|}$.

An association rule is an implication $X \rightarrow Y$ such that $X \cup Y \subseteq \mathcal{I}$ and $X \cap Y = \emptyset$. X is the antecedent and Y is the consequent of the rule. The *support* of $X \rightarrow Y$ in $\mathcal{W}$ is the proportion of transactions in $\mathcal{W}$ that contains $X \cup Y$. The *confidence* of $X \rightarrow Y$ is the proportion of transactions in $\mathcal{W}$ containing X that also contains Y.

3.1 Rare Itemsets

We adopted the rare itemsets concept from Tsang et al. We consider an itemset to be rare when its support falls below a threshold, called the minimum frequent support (minFreqSup) threshold. One difficulty when generating rare itemsets is differentiating noisy itemsets from the actual rare itemsets. As the support of the itemset is low, the potential of pushing unrelated items together increases as well, thus producing noisy itemsets. To overcome this problem, we define a noise filter threshold to prune out the noise called the minimum rare support (minRareSup) threshold. Typically minRareSup is set to a very low level, e.g., 0.01%.

Definition 1. *An itemset x is a* rare itemset *in a window $\mathcal{W}$ iff*

$$sup_{\mathcal{W}}(x) \leq \textit{minFreqSup and } sup_{\mathcal{W}}(x) > \textit{minRareSup}$$

However not all rare itemsets that fulfill these properties are interesting. Furthermore, rare itemsets can be divided into two types: *rare item itemsets* and *non-rare item itemsets*.

Rare item itemsets refer to itemsets which are a combination of only rare items and itemsets that consist of both rare and frequent items. Given 4 items $\{a, b, c, x\}$ with supports a = 0.70, b = 0.45, c = 0.50, and x = 0.10, with minFreqSup = 0.15 and minRareSup = 0.01, the itemset $\{a, x\}$ would be a rare item itemset assuming that the support of $\{a, x\} >$ 0.01, since the itemset includes the rare item x.

Definition 2. *An itemset x is a* rare item itemset *iff*

$$\exists x \in X, sup_{\mathcal{W}}(x) \leq \textit{minFreqSup}, sup_{\mathcal{W}}(x) \leq \textit{minFreqSup}$$

Non-rare item itemsets only has frequent items which fall below the minimum frequent support threshold. Given 4 items $\{a, b, c, x\}$ with supports a = 0.70, b = 0.45, c = 0.50, and x = 0.10, with minFreqSup = 0.15 and minRareSup = 0.01, and the itemset $\{a, b, c\}$ had a support of 0.09, then this itemset would be a non-rare item itemset as all items within the itemset are frequent, and its support lies between minFreqSup and minRareSup.

Definition 3. *An itemset X is a* non-rare item itemset *iff*

$$\forall x \in X, sup_{\mathcal{W}}(x) > \textit{minFreqSup}, sup_{\mathcal{W}}(x) \leq \textit{minFreqSup}$$

4 SRP-Tree: Rare Pattern Tree Mining for Data Streams

In this section we will be discussing the details of our proposed technique SRP-Tree that mines rare patterns in a data stream using a sliding window approach.

4.1 SRP-Tree

Current tree based rare pattern mining approaches follow the traditional FP-Tree [7] approach. It is a two-pass approach and is affordable when mining a static dataset. However in a data stream environment, a two-pass approach is not suitable. To process a static environment, traditional non-streaming rare pattern techniques, such as RP-Tree order the items within a transaction according to its frequencies before inserting it into the tree. Using these algorithms the frequency of the items are obtained during the first pass through the dataset.

In data streams we can only look at the transactions within the stream once, thus, a one-pass approach is necessary. This rules out the possibility of building a tree based on the frequency of items within the data stream. Furthermore, frequency of an item may change as the stream progresses. There are four scenarios which we need to consider in a stream:

Scenario 1. A frequent item x at particular time T_1 may become rare at time T_2.
Scenario 2. A rare item x at particular time T_1 may become frequent at time T_2.
Scenario 3. A frequent item x at particular time T_1 may remain frequent at time T_2.
Scenario 4. A rare item x at particular time T_1 may remain rare at time T_2.

T_1 represents a point in time, and T_2 represents a future point in time after T_1.

To find rare patterns within data streams, we propose a new algorithm called SRP-Tree. Our algorithm uses a tree based approach. In our approach, items from incoming transactions in the stream are inserted into a tree based on a canonical ordering. A canonical ordering allows us to capture the content of the transactions from the data stream and orders tree nodes according to a particular order. In our approach we use appearance order of the items as the canonical ordering. When we use a canonical order to build the tree, the ordering of items is unaffected by the changes in frequency caused by incremental updates. There has been work carried out using a canonical ordering to build trees for a data stream mining environment [6,10]. But all of these research have been tailored to find frequent patterns.

In our approach, the frequency of a node in the tree is at least as high as the sum of frequencies of its children. However, this does not actually guarantee the overall downward closure property which exists in a frequency ordered tree. The downward closure property in a traditional rare pattern tree mining algorithm, whereby, *rare-items will never be the ancestor of a non-rare item in the initial tree due to the tree construction process* is violated. Hence, we propose a novel item list called the Connection Table which keeps track of each unique item in the window and the items they co-occur with along with their respective frequencies.

4.2 Connection Table

The Connection Table used in our SRP-Tree captures in the transactions only items that have a lower canonical ordering. For example, given a transaction in a canonical order of $\{a, b\}$ we store in the table that a is connected to b with a frequency of 1 but it does not store b is connected to a. This is because of the properties of the canonical ordering in the constructed tree: item a will always be the ancestor of item b. Since mining is carried out using a bottom-up approach, by mining item b, item a is also mined. The opposite does not hold since mining item a does not guarantee that item b is mined. Therefore, by using the Connection Table to keep track of connected items and adding them as arguments to FP-Growth in the mining phase, the complete set of rare-item itemsets can then be captured by SRP-Tree. The Connection table is designed using a hash map which allows for $O(1)$ access. In worst case scenarios, the table could reach a max size of $\frac{x(x+1)}{2}$ with x being the total amount of items; however, in reality this is highly unlikely.

At any point of time should we decide to mine the current window, SRP-Tree uses the initial tree of the current window to construct conditional pattern bases and conditional trees for each rare-item and their connected items in the Connection Table. Note that only connection items with an occurring frequency greater than or equal to the minRareSup is included. Each conditional tree and corresponding item are then used as arguments for FP-Growth. The threshold used to prune items from the conditional trees is minRareSup. The union of the results from each of these calls to FP-Growth is a set of itemsets that contains a rare-item, or rare item itemsets.

Example. Given the dataset in Table 1, we show how the Connection Table is built in Table 2. The left column in Table 2 list the unique items in the window, whereas the right column list the set of co-occurring items along with the co-occurrence frequency of that particular item to the item in the right column. For example, item c co-occurs twice with items d, f, and h.

Table 1. Set of transactions in a given window $\mathcal{W}$ of size 12

Tid	Transaction	Tid	Transaction	Tid	Transaction
1	{a, g, h}	5	{c, f, h}	9	{c, d, h}
2	{a, g, h, i}	6	{a, g, h, e}	10	{b, f}
3	{b, c, d, f}	7	{g}	11	{a, h}
4	{b, d, j}	8	{h}	12	{a}

Table 2. Connection Table using the window of transactions listed in Table 1

Item	Items Co-occurred	Item	Items Co-occurred
a	{(e:1), (g:1), (h:1), (i:1)}	f	{(h:1)}
b	{(c:1), (d:2), (f:2), (j:1)}	g	{(h:1), (i:1)}
c	{(d:2), (f:2), (h:2)}	h	{(i:1)}
d	{(f:1), (h:1), (j:1)}	i	{∅}
e	{(g:1), (h:1)}	j	{∅}

4.3 SRP-Tree Algorithm

Our SRP-Tree algorithm is shown in Algorithm 1. SRP-Tree essentially performs in one pass the counting of item frequencies and the building of the initial tree. Therefore, in a given window $\mathcal{W}$, for each incoming transaction t, SRP-Tree first updates the list of item frequencies according to the transactions contained in the current window. We refer to this as updateItemFreqList(t) method, where we increment the counts of items contained in the new transaction and decrement the counts of items contained in the oldest transcations to be discarded from the window. SRP-Tree then updates the tree structure according to the transaction contained in the current window in a similar fashion, which is referred to as updateTree(t) method in Algorithm 1.

We mine the tree after a particular number of transactions also known as a block. We refer to a preset block size as $\mathcal{B}$. In this algorithm, $\mathcal{R}$ refers to the set of rare items and $\mathcal{C}$ refers to the set of items that co-occur with a particular rare item.

We would also like to point out, that another difference between SRP-Tree and a static rare pattern mining approach is that in SRP-Tree, the tree is built using all the transactions in the window, whereas in a static rare pattern mining approach, only transactions with rare items are used to build the tree. A static approach has the luxury of looking at the dataset twice and discarding items which it is not interested in before the tree is even built. In our case, we simply cannot know which transactions contain rare items until we decide to mine.

Algorithm 1. SRP-Tree

```
1:  **Input:** $DS$, $\mathcal{W}$, $\mathcal{B}$, $minRareSup, minFreqSup$;
2:  **Output:** $results$ (Set of rare item itemsets);

3:  while exist($DS$) do
4:      $t$ ←new incoming transaction from $DS$;
5:      currentBlockSize ← currentBlockSize + 1;
6:      updateItemFreqList($t$);
7:      updateConnectionTable($t$);
8:      $tree$ ← updateTree($t$);
9:      Mining at the end of block
10:     if $\mathcal{B}$ == currentBlockSize then
11:         currentBlockSize ← 0;
12:         $results = \emptyset$;
13:         $\mathcal{I} \leftarrow$ {all unique items in $\mathcal{W}$};
14:         $\mathcal{R} \leftarrow \{i \in \mathcal{I} \mid sup_{\mathcal{W}}(i) \geq minRareSup \wedge sup_{\mathcal{W}}(i) < minFreqSup\}$;
15:         $\mathcal{C} \leftarrow \{k \in \mathcal{R}, j \in connectionTable(k) \mid sup_{\mathcal{W}}(k) \geq minRareSup\}$;
16:         for item $a$ in $tree$ do
17:             if $a \in \mathcal{R}$ or $a \in \mathcal{C}$ then
18:                 construct $a$'s conditional pattern-base and then $a$'s conditional FP-Tree $Tree_a$;
19:                 $results \leftarrow results \cup$ FP-Growth($Tree_a, a$);
20:             end if
21:         end for
22:     end if
23: end while
24: return $results$;
```

SRP-Tree Example. Applying SRP-Tree to the window $\mathcal{W}$ of the 12 transactions in Table 1, the support ordered list of all items is $\langle$(h:6), (a:5), (g:4), (c:3), (d:3), (b:2), (f:2), (e:1), (i:1), (j:1)$\rangle$. Using minFreqSup = 4 and minRareSup = 2, only the items $\{b, c, d, f\}$ are rare, and included in the set of rare items, $\mathcal{R}$.

The initial SRP-Tree constructed for window W of size 12 is shown in Figure 1. To find the rare-item itemsets, the initial SRP-Tree is used to build conditional pattern bases and conditional SRP-Trees for each rare item $\{b, c, d, f\}$ and any additional items in the Connection Table that is connected with a rare item that has a frequency greater than the minRareSup, in this example, item h. The conditional tree for item h is shown in Figure 2. Each of the conditional SRP-Trees and the conditional item are then used as parameters for the FP-Growth algorithm.

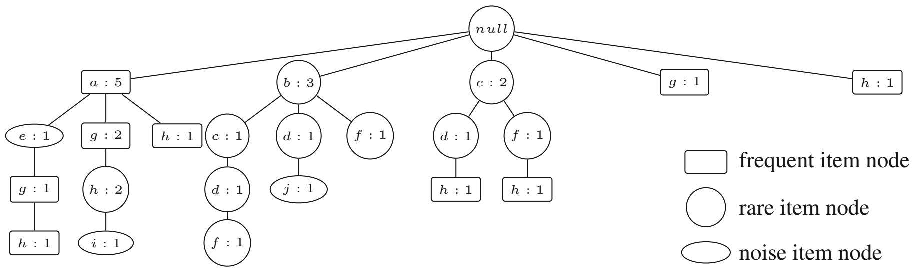


Fig. 1. Pattern tree constructed from window $\mathcal{W}$ using SRP-Tree

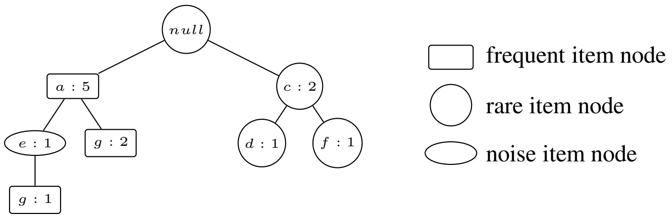


Fig. 2. Conditional tree, *Tree*$_h$

5 Experimental Results

In our experiments we compared the performance of our SRP-Tree to the Streaming Canonical Tree (SC-Tree), which is a modified version of DSTree [10] with pruning of frequent itemsets. SRP-Tree is the very first attempt at mining rare patterns in a data stream environment and there are no other techniques that mine rare patterns in a data stream to compare to, so we have compared our SRP-Tree to the technique that we call the SC-Tree with pruning which finds rare item itemsets in an unoptimized brute-force manner. The SC-Tree is a one pass technique which stores the transactions from the stream in a tree using the canonical ordering and stores/updates item frequencies like that of our SRP-Tree. One of the major differences of SC-Tree is that SC-Tree does not store or keep track of items in the Connection Table. To find all rare item itemsets, SC-Tree finds and generates all itemsets that meet the minRareSup threshold, then removes all other itemsets except rare item itemsets with an extra pruning step. SC-Tree produces the same itemsets and generates the same rules as SRP-Tree. All algorithms were implemented in Java and executed on a machine equipped with an Intel Core i5-2400 CPU @ 3.10 GHz with 4GB of RAM running Windows 7 x 64.

5.1 Real-World Dataset

In this section we present the time and relative time taken for itemset generation of SRP-Tree and SC-Tree. The time is reported in seconds and the relative time is calculated by setting RP-Tree to 1.00 and SCTree relative to that of RP-Tree.

$$\text{relative time} = \frac{\text{Time taken by SC-Tree}}{\text{Time taken by SRP-Tree}}$$

We used a window size and block size of 25K for all datasets except for the Mushroom dataset where we used 2K due to its smaller size. The minFreqSup and minRareSup thresholds are shown in Table 3 for each dataset. The thresholds are user-defined through examining the distribution of item frequencies in each of the datasets. We acknowledge that given a data stream environment, this is not the most suitable way of defining thresholds and in the future we will be looking at a way to adapt the thresholds to the change in distribution and drift of the transactions in the stream.

We have tested the algorithms on 6 datasets from the FIMI (Frequent Itemset Mining Implementations) repository: Mushroom, Retail, BMS-POS, T10I4D100K, T40I10D100K, and Kosarak (250K).

Table 3. Comparision between SRP-Tree and SC-Tree

Dataset	$\mathcal{B}$	MinRareSup	MinFreqSup	No. of Itemset	SRP-Tree		SC-Tree	
					Time (s)	Rel. Time	Time (s)	Rel. Time
Mushroom	2K	0.01	0.05	14443674	1131	1.00	1321	1.17
Retail	25K	0.0001	0.0005	572673	239	1.00	339	1.42
BMS-POS	25K	0.0002	0.0005	1426	58	1.00	3783	65.22
T10I4D100K	25K	0.0001	0.0005	1161	12	1.00	19	1.58
T40I10D100K	25K	0.003	0.05	4734806	301	1.00	380	1.26
Kosarak (250K)	25K	0.001	0.15	35623519	703	1.00	7213	10.26

Table 3 shows, for each dataset, the block size, minRareSup, and minFreqSup used to run the algorithms then the comparison between SRP-Tree and SC-Tree of the itemsets generated, time it took to run the algorithm, and the relative time. Both SRP-Tree and SC-Tree generate the same amount of itemsets because SRP-Tree only generates rare-item itemsets and SC-Tree generates all itemsets that meet the minRareSup threshold then the extra pruning step removes all other itemsets that are not rare-item itemsets. Therefore, the final number of itemsets generated by both algorithms is the same.

In all datasets of varying item frequency distribution, SRP-Tree runs faster than SC-Tree. The time taken to run the various datasets are highly dependent on the tree structure built from the transactions in the datasets and generally have a positive correlation with the number of itemsets generated. The number of itemsets generated also has a great variation and is highly dependent on the composition/nature of the respective dataset. For datasets that are more sparse in nature like Retail, BMS-POS, and T10I4D100K, the number of itemsets generated are usually than a dense dataset like Mushroom and the run-time is usually faster.

5.2 Case Study: T40I10D100K

The T40I10D100K dataset is generated using the generator from the IBM Almaden Quest research group. Figure 3 and Figure 4 shows the item frequency distribution of the T40I10D100K dataset. When we compared the distribution of the T40I10D100K dataset, to other datasets in the FIMI repository, we observed that the T40I10D100K dataset is more sparse in nature compared to the Mushroom dataset, but more dense than datasets like Retail and BMS-POS. It is also important to note that T40I10D100K contains a high proportion of items with a frequency of less than 0.1 (approximately 90% of the total items). This particular distribution contains a greater number of prospective rare items and rules to be mined from the dataset.

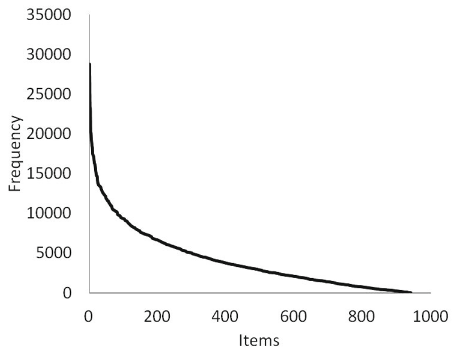


Fig. 3. Item frequency distribution

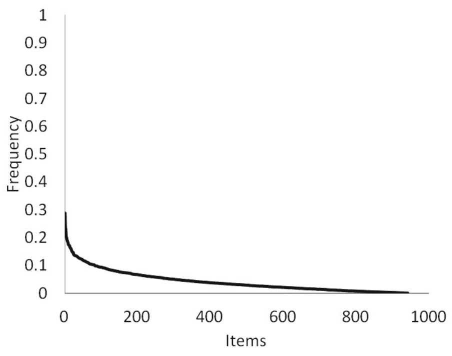


Fig. 4. Normalized item frequency distribution

In Table 4 we show the difference in itemsets generated and the time it took to generate items of varying minFreqSup on the T40I10D100K dataset. As we increase the minFreqSup, the number of itemsets generated increases and the relative time also increases. It is important to note that the real time taken for SRP-Tree to generate itemsets decreases as we increase the minFreqSup. This is because at a lower minFreqSup for this particular distribution of the dataset, there is a high co-occurrence of rare items to other items with similar item frequency (indicating that these other items are also likely to have rare properties). The mining of these additional highly connected items with rare association patterns incurred a larger overhead in maintaining the Connection Table. As the minFreqSup increases and most of the highly connected potential rare items are accounted for, the overhead decreases and the results in a faster runtime.

Table 5 shows the difference in execution time when the block size varies. The execution time is increased as the block size increases due to the increased size of the tree being built.

Table 4. Varying MinFreqSup for T40I10D100K

MinFreqSup	Itemset	SRP-Tree		SC-Tree	
		Time(s)	Rel. Time	Time(s)	Rel. Time
0.04	4397517	333	1.00	357	1.07
0.05	4734806	301	1.00	380	1.26
0.06	5028947	278	1.00	421	1.51
0.1	5105892	246	1.00	446	1.81
0.15	5136904	238	1.00	454	1.91

Table 5. Execution Time based on Varying Block Sizes for T40I10D100K

Block size	SRP-Tree		SC-Tree	
	Avg Time / Window (s)	Relative Time	Avg. Time / Window (s)	Relative Time
10K	41	1.00	70	1.70
25K	75	1.00	95	1.26
50K	142	1.00	201	1.42

6 Conclusions and Future Work

We present a new method for mining rare patterns using a tree structure in a data stream environment. To the extent of our knowledge, this is the first algorithm that looks at mining rare patterns in a data stream. Our technique is a one-pass only strategy which is capable of mining rare patterns in a static database or in a dynamic data stream. In the case of mining data streams, our technique is also capable of mining at any given point of time in the stream and with different window and block sizes. One of the contributions of this algorithm is a novel approach using a Connection Table which keeps track of related items in a sliding window and reduces the mining space during itemset generation. In our evaluations on six different datasets, the SRP-Tree is capable of generating itemsets in a more efficient manner compared to the SC-Tree.

In the future we intend to investigate on dynamically adapting the minRareSup and minFreqSup thresholds on-the-fly because data streams are volatile and neither setting fixed thresholds for all data nor defining the thresholds prior based on distribution is deemed suitable. We will also look at the possibility of dynamically adjusting the window size to reflect the density of incoming data in the stream. For example, if the new transactions in the window contained uninteresting or duplicate itemsets and rules, we could (through varying the window size) decide not to mine until more interesting itemsets and rules are captured. It will also be interesting to investigate the limitations of the tree with respect to the different characteristics and intensity of the incoming data stream.

References

1. Adda, M., Wu, L., Feng, Y.: Rare itemset mining. In: Proceedings of the Sixth International Conference on Machine Learning and Applications, ICMLA 2007, pp. 73–80. IEEE Computer Society, Washington, DC (2007)

2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, pp. 487–499 (1994)
3. Cheng, J., Ke, Y., Ng, W.: Maintaining frequent closed itemsets over a sliding window. J. Intell. Inf. Syst. 31, 191–215 (2008)
4. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining closed frequent itemsets over a stream sliding window. In: Proceedings of the Fourth IEEE International Conference on Data Mining, ICDM 2004, pp. 59–66. IEEE Computer Society, Washington, DC (2004)
5. Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. Knowl. Inf. Syst. 10, 265–294 (2006)
6. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows (extended abstract). In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2002, pp. 635–644. Society for Industrial and Applied Mathematics, Philadelphia (2002)
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD 2000, pp. 1–12. ACM, New York (2000)
8. Koh, Y.S., Rountree, N.: Finding Sporadic Rules Using Apriori-Inverse. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 97–106. Springer, Heidelberg (2005)
9. Lee, C.H., Lin, C.R., Chen, M.S.: Sliding window filtering: an efficient method for incremental mining on a time-variant database. Information Systems 30(3), 227–244 (2005)
10. Leung, C.K.S., Khan, Q.I.: Dstree: A tree structure for the mining of frequent sets from data streams. In: Proceedings of the Sixth International Conference on Data Mining, ICDM 2006, pp. 928–932. IEEE Computer Society, Washington, DC (2006)
11. Li, H.F., Lee, S.Y.: Mining frequent itemsets over data streams using efficient window sliding techniques. Expert Syst. Appl. 36, 1466–1477 (2009)
12. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 337–341 (1999)
13. Mozafari, B., Thakkar, H., Zaniolo, C.: Verifying and mining frequent patterns from large windows over data streams. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 179–188. IEEE Computer Society, Washington, DC (2008), http://dl.acm.org/citation.cfm?id=1546682.1547157
14. Szathmary, L., Napoli, A., Valtchev, P.: Towards rare itemset mining. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007, vol. 1, pp. 305–312. IEEE Computer Society, Washington, DC (2007)
15. Tanbeer, S.K., Ahmed, C.F., Jeong, B.S., Lee, Y.K.: Sliding window-based frequent pattern mining over data streams. Information Sciences 179(22), 3843–3865 (2009)
16. Troiano, L., Scibelli, G., Birtolo, C.: A fast algorithm for mining rare itemsets. In: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, pp. 1149–1155. IEEE Computer Society, Washington, DC (2009)
17. Tsang, S., Koh, Y.S., Dobbie, G.: RP-Tree: Rare Pattern Tree Mining. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 277–288. Springer, Heidelberg (2011)

A Single Pass Trellis-Based Algorithm for Clustering Evolving Data Streams

Simon Malinowski and Ricardo Morla

INESC-TEC, Faculty of Engineering, University of Porto

Abstract. The main paradigm for clustering evolving data streams in the last 10 years has been to divide the clustering process into an online phase that computes and stores detailed statistics about the data in micro-clusters and an offline phase that queries micro-cluster statistics and returns desired clustering structures. The argument for two-phase algorithms is that they support evolving data streams and temporal multi-scale analysis, which single pass algorithms do not. In this paper, we describe a single pass fully online trellis-based algorithm, named ClusTrel, designed for centroid-based clustering that supports evolving data streams and generates clustering structures right after a new point is processed. The performance of ClusTrel is assessed and compared to state of the art algorithms for clustering of data streams showing similar performance with smaller memory footprint.

1 Introduction

The increasing number of sensors and monitoring devices in today's intelligent systems makes data stream mining a topic of current interest. Mining data streams is of use in various application domains such as network management, health monitoring, sports, finance, etc. Amongst the different topics related to data stream mining, clustering the data points from a data stream is the subject of many research works over the last few years [1–9].

Algorithms designed to cluster data streams need to deal with specific additional requirements compared to those designed for offline clustering. These additional requirements make this topic extremely challenging. Systems designed for clustering of data streams should comply with the following requirements:

– The points of the stream have to be processed as they arrive and cannot be stored in memory.
– The stream cannot be interrupted to process the points: there is limited time before the next item arrives.
– An up-to-date clustering of the stream should be maintained to give more importance to recent points.
– Stream clustering algorithms should be capable of detecting noise in the streams.
– No *a priori* information on the number of clusters in the stream. The clustering algorithm must self-adapt to detect the number of clusters.

– The system should be able to deliver a clustering structure frequently. This enables the detection of concept drifts promptly after they occur.

Most of the algorithms proposed in the literature for data stream clustering fulfill the above requirements to different extents. Current algorithms for data stream clustering rely on two different phases: a first phase that maintains sufficient statistics about the stream and a second phase that generates the clustering structure of the stream given the gathered statistics. The first phase is done online and the second one is left for offline whenever a clustering snapshot is required. This offline phase represents an increase in processing time and complexity as it needs to be done every time a clustering snapshot is desired. Because of this increase in time and complexity, the frequency of the offline process is typically much smaller than that of the incoming data points, which introduces delay in the detection of concept changes. Slow reaction to changes in the clustering structure can prove to be detrimental to service quality and security in application domains such as network anomaly and intrusion detection [10].

In this paper, we present ClusTrel, a single pass fully online trellis-based algorithm designed for centroid-based clustering of data streams. Thanks to its trellis structure ClusTrel updates the clustering snapshot after each point and can hence detect concept changes for every incoming data point. This algorithm is also capable of selecting the number of clusters based on a clustering evaluation index. We use the MOA software [11] to generate synthetic streams and compare the performance of ClusTrel with state of the art algorithms for clustering data streams. Simulation results show that ClusTrel is able to reach similar performance while reducing the number of micro-clusters stored in memory.

The remainder of this paper is structured as follows. Related work is surveyed in Section 2, the ClusTrel algorithm is described in Section 3. Simulation results on synthetic and real streams are given in Section 4 and conclusions are drawn in Section 5.

2 Related Work

The initial approaches for clustering data streams focused on supporting the single pass requirements [1, 3]. A common pitfall of these approaches is that new points have the same weight as old points, making it difficult to adapt to changes in the stream. These approaches also require that the target number of clusters is provided as input to the clustering algorithm, which is an obvious limitation in the case of evolving data streams.

Most recent two-phase algorithms for clustering data streams rely on micro-clusters [2, 4–7]. Micro-clusters are a compact representation of clusters that maintain sufficient statistics that can be updated online. Creating and indexing micro-clusters from the data stream is established as the online part of most of these algorithms, while the offline process generates the final clustering structure from the current set of micro-clusters.

The CluStream [2] algorithm keeps a fixed maximum number of micro-clusters at each instant. New points that fall within the boundary of already existing micro-clusters are appended to them, while others form new micro-clusters. This avoids having to merge clusters at every step and can be implemented using micro-cluster feature vectors. CluStream also allows for the deletion of old micro-clusters and can store snapshots of the current clustering structure at different time horizons. The offline process of CluStream takes as input the desired number of clusters and uses a k-means algorithm to generate the final clustering from the current set of micro-clusters. ClusTree algorithm [7] uses an R^*-tree structure in order to index the micro-clusters that are updated online. The different levels of the tree build a hierarchical representation of the clustering. The authors propose different strategies to insert the micro-clusters into the tree. ClusTree can hence handle slow to very fast stream by adapting the way micro-clusters are inserted in the tree. The LiarTree algorithm [9] extends the concepts of ClusTree to cater for noise detection and the improvement of novelty detection in the streams.

Alternative approaches use density-based clustering such as those in [5, 6, 8]. The algorithms presented in [5] and [6] are also based on an online and an offline phase. In [5], the online phase consists of creating and updating micro-clusters, and separating outlier micro-clusters from core micro-clusters. A density-based clustering strategy (e.g. DBSCAN [12]) is used as the offline phase to generate the final clustering. The online part of D-Stream [6] is based on density grids. In [8], the authors propose a single pass density based clustering approach named *FlockStream* that makes use of so-called agents.

The ClusTrel algorithm proposed here is a single pass algorithm similar to *FlockStream* but is a centroid-based approach. The state of the art algorithms for centroid-based clustering of data streams are CluStream [2] and ClusTree [7].

The approach proposed in this paper differs from CluStream and ClusTree as it is a single pass algorithm that gathers statistics from the stream and generates a macro-clustering after every incoming point.

3 The ClusTrel Algorithm

3.1 Preliminary Notations

Let $S = S_1, \ldots, S_n, \ldots$ be a stream taking its values in $\mathbb{R}^d$. We assume that our processing system does not have enough memory to store all the points of the stream.

Definition 1. *A micro-cluster is a compact representation of a cluster. It is defined by a cluster feature vector (CFV). In this paper, a CFV of a micro-cluster is a* $(d+2)$ *tuple* $(n, c, ssqd)$*, where* n *is the number of points associated to the cluster,* c *is its centroid and* $ssqd$ *is the sum of the squared distances of all points in the cluster to* c*.*

The CFV of a micro-cluster enables the gathering of sufficient statistics about the points of the stream assigned to that micro-cluster without the need for storing all points in memory. We will see later how the CFVs are updated when new points are inserted in micro-clusters.

Definition 2. *We denote by $\mathcal{C}_k^{(t)}$ a clustering structure that represents the clustering of the stream S up to sample S_t into k micro-clusters. The k micro-clusters are represented by their CFVs.*

ClusTrel uses a clustering evaluation measure named the MDB index. It is based on the Davies-Bouldin index [13] and is defined as follows:

Definition 3. *Let $\mathcal{C}_k^{(t)}$ be a clustering structure. The MDB index of $\mathcal{C}_k^{(t)}$ is defined as :*

$$MDB = \frac{1}{k}\sum_{i=1}^{k} \max_{j\neq i} \frac{SSQ(C_i) + SSQ(C_j)}{dist(C_i, C_j)^2}, \tag{1}$$

where $SSQ(C_i)$ is the average squared distance of all points in cluster C_i to its centroid and $dist(C_i, C_j)$ is the distance between the centroids of C_i and C_j.

The advantage of the MDB index over the classical DB index is that it is computable online from the CFVs of a clustering structure. Low values of the MDB index are associated with clustering structures composed of compact and well-separated clusters. The MDB is used in ClusTrel to determine the number of clusters that is most adapted to the input stream.

ClusTrel also uses the average SSQ index as a measure of quality of a clustering structure. The average SSQ index is simply the average square distance of the points of the stream to the centroid of the cluster they are assigned to.

3.2 The ClusTrel Algorithm

ClusTrel is a dynamic programming algorithm, based on the Viterbi algorithm [14]. Given the data stream, ClusTrel minimizes one of the two cluster evaluation indices described above. It dynamically builds a trellis whose states are clustering structures, as shown in Figure 1 where the horizontal axis represents time. The two parameters n_m and n_M represent the minimum and maximum numbers of clusters to be explored by ClusTrel to find a clustering structure for the stream S.

ClusTrel considers three kinds of transition in the trellis from a given clustering structure, as shown in Figure 1. We denote these by *Inc*, *Dec*, and *Keep* transitions. Each transition generates a new clustering structure given a current structure and an incoming point. An *Inc* transition generates a structure with an additional cluster, a *Dec* transition generates a structure with one less cluster, while the structure generated by a *Keep* transition has the same number of clusters as the input one. These three transitions incorporate the incoming point in a structure and modify the CFVs that need to be updated. With a clustering structure of k clusters and an incoming point p as input, the three transitions of ClusTrel are detailed in the following.

Inc **transition.** An *Inc* transition produces a clustering structure of $k+1$ clusters updated with p. It generates a new cluster on the point p. The CFVs of the k already existing clusters are unchanged and the CFV of the newly created cluster is simply $(1, p, 0)$.

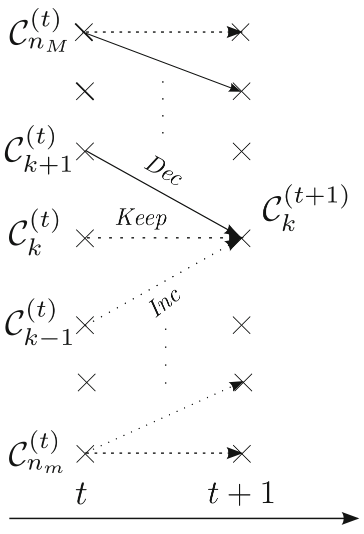

$\mathcal{C}_{n_M}^{(t)}$
$\mathcal{C}_{k+1}^{(t)}$
Dec
$\mathcal{C}_k^{(t)}$
$Keep$
$\mathcal{C}_k^{(t+1)}$
Inc
$\mathcal{C}_{k-1}^{(t)}$
$\mathcal{C}_{n_m}^{(t)}$
t
$t+1$

Fig. 1. The trellis structure of ClusTrel. The states of the trellis are clustering structures with a number of clusters varying from n_m to n_M. The three kinds of arrows represent the possible transitions between structures at two consecutive time stamps.

Keep **transition.** A *Keep* transition generates a clustering structure with k clusters. It first finds the cluster in the structure whose centroid is the closest to p, denoted $\overline{C}$. The point p is then appended into $\overline{C}$. Only the CFV of $\overline{C}$ needs to be updated. If this CFV is equal to (n, c, s), it is updated into (n', c', s'), where

$$\begin{cases} n' = n + 1, \\ c' = (p + n \times c)/n', \\ s' = s + dist(p, c')^2 + n \times dist(c, c')^2, \end{cases} \tag{2}$$

where $dist$ represents the Euclidean distance.

Dec **transition.** A *Dec* transition generates a clustering structure of $k-1$ clusters. It first merges two clusters C_i and C_j of the current clustering structure. We denote the CFVs of these clusters (n_i, c_i, s_i) and (n_j, c_j, s_j) respectively. C_i and C_j are merged into C'_i whose CFV (n'_i, c'_i, s'_i) is equal to

$$\begin{cases} n'_i = n_i + n_j, \\ c'_i = (n_i \times c_i + n_j \times c_j)/n'_i, \\ s'_i = s_i + s_j + n_i \times dist(c_i, c'_i)^2 + n_j \times dist(c_j, c'_i)^2. \end{cases} \tag{3}$$

It then incorporates p into the closest cluster of the structure and updates the appropriate CFV according to Eqn(2). Performing a *Dec* transition on a clustering structure $\mathcal{C}_k^{(t)}$ is equivalent to selecting a pair of clusters $(C_i, C_j), i \neq j, 1 \leq i, j \leq k$ to merge. We propose here three different ways to choose the two clusters to be merged in a *Dec* transition. The corresponding transitions are denoted *Dec1*, *Dec2* and *Dec3* in the following.

Dec1 **transition.** The *Dec1* transition selects the clusters (C_i, C_j) that are the closest in $\mathcal{C}_k^{(t)}$.

Dec2 **transition.** For every cluster C_i in $\mathcal{C}_k^{(t)}$, the closest cluster to C_i is searched for. This cluster is denoted $\overline{C}_i$. Merging C_i and $\overline{C}_i$ leads to a new clustering structure whose average SSQ sq_i can be computed according to Eqn(3). The pair of clusters to merge is chosen as $(C_j, \overline{C_j})$ such that $j = \arg\min_{1 \leq i \leq k} sq_i$.

Dec3 **transition.** For the *Dec3* transition, the pair (C_i, C_j) is chosen as the one that minimizes the SSQ index over all possible pairs of clusters in the given clustering structure.

The processing time associated with the three *Dec* transitions described above increases from *Dec1* to *Dec3*. We will see in the experimental results section that *Dec2* and *Dec3* transitions can lead to better clustering performance, but also that the gain brought by *Dec3* over *Dec2* is small, so that *Dec2* seems to be a good trade-off between performance and complexity. In the following, ClusTrel-1, ClusTrel-2 and ClusTrel-3 will refer to ClusTrel used respectively with the *Dec1*, *Dec2* and *Dec3* transitions.

Initialization Step of ClusTrel. The initialization step of ClusTrel consists of generating the first clustering structure of the trellis: $\mathcal{C}_{n_m}^{(n_m)}$ which is composed of n_m clusters of 1 point, hence taking into account the first n_m points of the stream S. All the others structures in the same vertical slice of the trellis are empty.

Updating Step of ClusTrel. Let us now assume that the t first points of the stream are already processed and that the trellis is filled with the clustering structures $\mathcal{C}_{n_m}^{(t)}$, $\mathcal{C}_{n_m+1}^{(t)}$, $\ldots$, $\mathcal{C}_{n_M}^{(t)}$. Given the incoming point S_{t+1}, ClusTrel updates these $n_M - n_m + 1$ structures. As can be seen in Figure 1, a structure at time $(t+1)$ can be updated from at most three different structures at time (t) (only two if the state is at the top or bottom of the trellis). ClusTrel algorithm computes $\mathcal{C}_k^{(t+1)}$ by choosing the best clustering structure (in terms of a cluster evaluation index) that ends up in state $\mathcal{C}_k^{(t+1)}$ from the previous slice of the trellis. In other words,

$$\forall k \in [n_m, n_M], \mathcal{C}_k^{(t)} = \arg\min(Inc(\mathcal{C}_{k-1}^{(t-1)}), Dec(\mathcal{C}_{k+1}^{(t-1)}), Keep(\mathcal{C}_k^{(t-1)})), \qquad (4)$$

where the arg min is taken in terms of the desired cluster evaluation index (MDB or SSQ). This step performs a local minimization of the index of the clustering structures in the trellis given the data stream. In the following, we use ClusTrel together with the minimization of the SSQ index.

For memory purposes, only the latest structures are kept in memory. Updating the structures at time $t+1$ only needs the incoming point and the structures at time t. Hence, the maximum number of CFVs stored in a slice of the trellis is equal to $1/2 \times (n_M + n_m)(n_M - n_m + 1)$. However, some micro-clusters can appear in different clustering structures. In order to reduce the amount of memory needed, a list of unique micro-clusters is kept by the ClusTrel algorithm. These micro-clusters are indexed, and the clustering structures $\mathcal{C}_{n_m}^{(t)}, \ldots, \mathcal{C}_{n_M}^{(t)}$ are described by the indexes of the micro-clusters that compose each structure.

Selection of the Final Clustering Structure. Whenever a snapshot of the current clustering is desired, ClusTrel can output the clustering structure in the trellis with the most adapted number of clusters, w.r.t. to the MDB index. This index is used for the selection of the best clustering structure as it gives a compromise between compact and well-separated clusters, which the SSQ index does not. The MDB index of the structures in the trellis is calculated online without the need for further information. The different values of the MDB index in the trellis can also give soft information about the different clustering structures that might be interesting to consider for the given stream.

3.3 Summary of the ClusTrel Features

ClusTrel is a single pass centroid-based clustering algorithm that does not require any offline process to deliver the final clustering result. ClusTrel does not need the number of clusters to search for as an input parameter as it searches for the best clustering structure with a number of clusters in $[n_m, n_M]$. In addition, it is able to output the best clustering structure (in terms of one of two cluster evaluation indices) at any time, and without the need for further processing. The selection and output of the best clustering structure is inherent to ClusTrel.

As far as memory is concerned, at most $1/2 \times (n_M + n_m)(n_M - n_m + 1)$ micro-clusters are kept in memory after the processing of an incoming point. For each of these micro-clusters, a CF vector is stored. An important point is that the maximum number of clusters n_M does not need to be much higher than the order of magnitude of the real number of clusters in the data, as will be shown in the simulation results section.

ClusTrel is able to detect changes in concept in the stream when the selected number of clusters evolves in a certain period a time. The fact that ClusTrel can generate clustering structures at every time instant make this detection faster. For two-phase algorithms, the offline process has to be executed frequently in order to quickly detect changes in concept in the data stream, which leads to a larger complexity of the overall system. The ClusTrel algorithm can also deal with up-to-date clustering by incorporating the concept of weighing down old points with a decay function (as in [4–8]).

As far as processing time is concerned, ClusTrel is more demanding than CluStream, DenStream or ClusTree, as it is a single pass algorithm that performs the statistics gathering phase and the macro-clustering phase in parallel. This algorithm may not be designed for very fast streams but is more adapted to applications where the focus is put on the accuracy of clustering and on fast detection of concept changes.

4 Experimental Results

We present some experimental results to assess the performance of ClusTrel with synthetic streams and streams from a real data set [15]. We focus here on the comparison between the clustering performance of ClusTrel and the ones of the two state of the art algorithms for centroid-based data stream clustering: CluStream [2] and ClusTree [7].

4.1 Using Synthetic Streams

The synthetic data streams used in this section were generated using the MOA software [11]. Three different streams with the following features are used:

1. 5 kernels in 2-dimensional space, 10,000 points
2. 8 kernels in 4-dimensional space, 12,000 points
3. 12 kernels in 6-dimensional space, 18,000 points.

The kernels that compose the streams are all Gaussian kernels. The radii of these Gaussian kernels is a parameter of the MOA stream generator. We have chosen values of the radii so that some of the kernels overlap in the space. The results presented for ClusTree and CluStream in this section are obtained with the MOA software that is freely available[1]. For all of these experiments, the correct number of clusters in the streams is detected by ClusTrel by looking at the MDB index of the clustering structures in the trellis. Hence, the SSQ indices given in this section always refer to clustering structures with as many clusters as the number of kernels in the stream.

Figure 2-(a) shows the clustering performance for the first stream in terms of the average SSQ index of ClusTree for different number of levels, and of CluStream for different numbers of micro-clusters kept at each time instant. The performance of ClusTrel-1 is also depicted as the horizontal black curve and is obtained for $n_M = 6$. For this stream, the clustering performance obtained with ClusTrel-1, ClusTrel-2 and ClusTrel-3 are the same. The best SSQ indices reached by ClusTree and CluStream are given by the last value of their respective curves. It can be seen that CluStream reaches the same performance as ClusTrel when 85 clusters are kept at each instant. This result highlights the benefit of the trellis structure of ClusTrel. The optimization of the clustering made at each time instant allows a decrease in the number of clusters (21 micro-clusters) that need to be kept in memory in order to obtain good performance.

Similar conclusions can be drawn when the second stream is used. The corresponding results are shown in Figure 2-(b). The performance of ClusTrel is again the same for the 3 variants of ClusTrel. The horizontal line represents the SSQ obtained by ClusTrel for $n_M = 10$. The best average SSQ obtained with ClusTree is always higher than the one obtained with ClusTrel. CluStream with 75 micro-clusters reaches the same performance as ClusTrel with 55 micro-clusters maximum.

[1] `http://moa.cs.waikato.ac.nz`

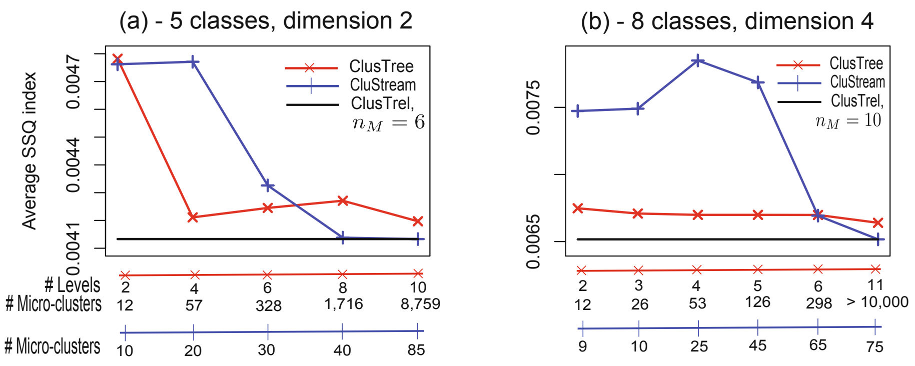


Fig. 2. Average SSQ index obtained by ClusTrel, ClusTree and CluStream for the first two synthetic streams

Table 1. Average SSQ index obtained by ClusTree and CluStream for the third synthetic stream. The performance of ClusTrel for $n_m = 15$ is the same as the one of CluStream with 120 micro-clusters.

ClusTree [7]					
# Levels	4	5	6	7	12
# Micro-clusters	56	133	331	677	> 10,000
SSQ index	0.018456	0.016093	0.016068	0.016035	0.015817
CluStream [2]					
# Micro-clusters	20	30	40	50	120
SSQ index	0.017162	0.016059	0.015349	0.015348	0.015347

The clustering performance of ClusTree and CluStream on the third stream are given in Table 1. For this stream, the average SSQ index obtained by ClusTrel with $n_M = 15$ (120 micro-clusters maximum) is equal to 0.015347, the same value obtained using CluStream with 120 micro-clusters.

These experiments made on synthetic streams highlight the benefit of the trellis structure of ClusTrel w.r.t. the maximum number of clusters that need to be kept in memory. Even when the height of the trellis is close to the real number of classes in the stream, ClusTrel is able to generate a clustering with a good SSQ index compared to the ones obtained by ClusTree and CluStream. In the following, we assess the performance of ClusTrel with a stream coming from a real data set and highlight the impact of ClusTrel-2 and ClusTrel-3 on the clustering performance.

4.2 Using Real Data

We use the Forest Covertype data set available from [15] to assess the performance of ClusTrel with real data streams. This data set is composed of approximately 500,000 10-dimensional points (only the numerical attributes are considered). The points are labeled with an integer from 1 to 7 corresponding to

7 different classes. Experimental results are given here in terms of the average SSQ index together with the widely-used purity index, defined as follows. For a set of classes $C = \{c_1, \ldots, c_L\}$ and a set of clusters $\Gamma = \{\gamma_1, \ldots, \gamma_K\}$, the purity index of the clustering is equal to

$$purity(C, \Gamma) = \frac{1}{N} \sum_k \max_l \left(\mathrm{card}(\gamma_k \cap c_l) \right), \tag{5}$$

where N is the total number of points. This index looks at how the different classes are distributed amongst the clusters. The closer the purity is to 1, the better the distribution of classes in the clusters.

We built a first stream composed of 100,000 points from the Forest Covertype data set. The proportion of the classes in this stream is similar to those in the whole data set.

The average SSQ and purity indices obtained by ClusTrel, ClusTree and CluStream for different parameters are given in Figure 3-(a) and (b) respectively. The performance obtained by the three variants of ClusTrel is given in this figure. The SSQ and purity index are calculated on a clustering structure of 7 clusters output by these algorithms (not at the micro-clustering level). The results on this figure demonstrate that the use of ClusTrel-2 and ClusTrel-3 brings an improvement in terms of clustering performance. The SSQ index obtained with ClusTrel-2 and ClusTrel-3 is almost always less or equal than the one of ClusTrel-1. The best performance is reached for this stream using ClusTrel-3 and $n_M = 11$.

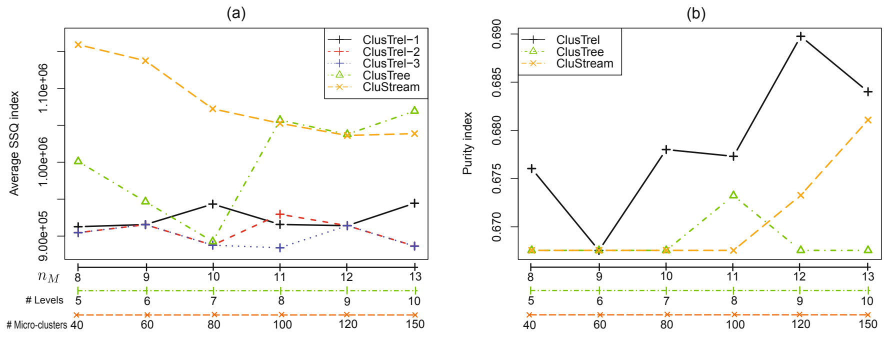


Fig. 3. Average SSQ index (a) and purity index (b) obtained by ClusTrel, ClusTree and CluStream for different parameters on a stream obtained from the Forest CoverType data set

Table 2. Average SSQ index obtained with the three variants of ClusTrel on a second stream based on the Forest CoverType data set

n_M	12	24	36
ClusTrel-1	$8.640e+5$	$8.720e+5$	$8.860e+5$
ClusTrel-2	$8.630e+5$	$8.564e+5$	$8.4884e+5$
ClusTrel-3	$8.630e+5$	$8.564e+5$	$8.4882e+5$

Fig 3-(b) shows that the purity index obtained by ClusTrel is higher than the ones obtained with ClusTree and CluStream for the different set of parameters.

Regarding the difference between the performance obtained by ClusTrel-2 and ClusTrel-3, it is important to note that they are equal or very close almost every time on this experiment. They are exactly equal for $n_M = 8, 9, 10$ and 12. The difference between the two is very close for $n_M = 13$ (it cannot actually be seen on the figure, but there is a less than a 0.1% difference). The benefit of ClusTrel-3 is only significant for $n_M = 11$ here. ClusTrel-1 and ClusTrel-2 hence seems to provide a good trade-off between processing complexity and clustering performance. The results given in Table 2 also support this observation. These average SSQ indices are obtained on another stream of 75,000 points from the Forest CoverType data set. It can be seen that the increase of the n_M parameter does not improve the clustering performance of ClusTrel-1 while it does for ClusTrel-2 and ClusTrel-3. The gain obtained with ClusTrel-3 over ClusTrel-2 is again small.

5 Conclusion

We propose in this paper ClusTrel, a single pass algorithm designed for the clustering of data streams. Clustering structures with different numbers of clusters and their statistics are maintained while processing the stream. These structures are chosen so that they minimize a cluster evaluation index at every step. ClusTrel is able to deliver a clustering of the stream as soon as a new point is processed. Thanks to the structure of ClusTrel, changes in concept in the streams can be detected quickly without the need for further processing. The performance of ClusTrel has been compared to two state of the art algorithms for clustering data streams. Simulation results show that ClusTrel is competitive with these algorithms in terms of clustering performance while enabling a reduction in the number of clusters that need to be stored in memory.

Acknowledgments. This work is financed by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Fundao para a Cincia e a Tecnologia (Portuguese Foundation for Science and Technology) within project CMU-PT/RNQ/0029/2009.

References

1. O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: Streaming-data algorithms for high-quality clustering. In: Proc. of Intl. Conf. on Data Engineering, pp. 685–694 (2002)
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92 (2003)
3. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: Theory and practice. IEEE Transactions on Knowledge and Data Engineering 15(3), 515–528 (2003)

4. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for projected clustering of high dimensional data streams. In: Proc. of the Intl. Conf. on Very Large Data Bases, pp. 852–863 (2004)
5. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: 2006 SIAM Conference on Data Mining, pp. 328–339 (2006)
6. Chen, Y., Tu, L.: Density-based clustering for real-time stream data. In: Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 133–142 (2007)
7. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: The ClusTree: indexing micro-clusters for anytime stream mining. In: Knowledge and Information Systems, pp. 1–24 (2010)
8. Forestiero, A., Pizzuti, C., Spezzano, G.: A single pass algorithm for clustering evolving data streams based on swarm intelligence. In: Data Mining and Knowledge Discovery, pp. 1–26 (2011)
9. Hassani, M., Kranen, P., Seidl, T.: Precise anytime clustering of noisy sensor data with logarithmic complexity. In: Proc. of International Workshop on Knowledge Discovery from Sensor Data, pp. 52–60 (2011)
10. Gu, G., Perdisci, R., Zhang, J., Lee, W.: Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th Conference on Security Symposium, pp. 139–154 (2008)
11. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. Journal of Machine Learning Research, 3–16 (2011)
12. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
13. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-1(2), 224–227 (1979)
14. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. on Information Theory 13(2), 260–269 (1967)
15. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml

New Management Operations on Classifiers Pool to Track Recurring Concepts

Mohammad Javad Hosseini, Zahra Ahmadi, and Hamid Beigy

Computer Engineering Department, Sharif University of Technology, Tehran, Iran
{mjhosseini,z_ahmadi}@ce.sharif.edu, beigy@sharif.edu

Abstract. Handling recurring concepts has become of interest as a challenging problem in the field of data stream classification in recent years. One main feature of data streams is that they appear in nonstationary environments. This means that the concept which the data are drawn from, changes over the time. If after a long enough time, the concept reverts to one of the previous concepts, it is said that recurring concepts has occurred. One solution to this challenge is to maintain a pool of classifiers, each representing a concept in the stream. This paper follows this approach and holds an ensemble of classifiers for each concept. As for each received batch of data, a new classifier is created; there will be a huge amount of classifiers which could not be maintained in the pool. To handle the memory limitations, a maximum number of concepts and classifiers are assumed. So the necessity of managing the concepts and classifiers is obvious. This paper presents a novel algorithm to manage the pool. Some pool management operations including merging and splitting the concepts are introduced. Experimental results show the performance dominance of using our method to the most promising stream classification algorithms.

Keywords: Recurring concepts, pool management, ensemble learning, data stream, concept drift.

1 Introduction

Classical classification algorithms assume that all training data are available at first. The classifier learns from the training data and after training, it is used for the test data. However in many real problems, we could not assume that all data are in hand. They arrive during the time; for example in the user interest problem, the interests of a user are collected by time and they also could change during the time. This form of problem violates the use of classical classification algorithms and new solutions are needed. Therefore, the field of data streams has been presented. Data stream is an unlimited stream of data which arrives in time and have some properties. The main property is the occurrence of concept drift, which means that the distribution of data changes over the time. Concept drift is divided into three main categories: abrupt, gradual and recurring [1]. Abrupt drift occurs when the distribution of data changes suddenly at a time slice, so that before that time the data is extracted from a distribution and after that, it is drawn from another distribution. Though in gradual drift, this change will not take place at a specific time, but in a period of time it happens. The third drift, called recurring concepts, occurs when an old concept which was learnt by the classifier some long enough time age, reappears. As the data evolves in time and gradual or abrupt drifts may occur, the classifier may forget the concepts belonging to a long time age. This makes recurring concepts to be a very challenging problem in the data stream field.

To support recurring concepts, this paper uses a pool of concepts. Each concept is modeled by an ensemble learner with a weighted structure, and it is represented by a modified version of Conceptual Vector [2, 3]. A new classifier is built on every batch of data and if its representing concept is detected to present one of the available concepts of the pool, it will be added to the corresponding ensemble; otherwise it will be added as a new concept in the pool. As data streams have huge amount of data and a classifier is built on every batch, the number of classifiers in an ensemble learner and the number of concepts in the pool should be limited. This leads us to present an effective management of concepts and classifiers in the pool. The management is done by two operations: merge and split. Two ensembles can be merged if they describe the same concept and an ensemble which contains classifiers of more than one concept can be split into two disjoint concepts. The experimental evaluations show the effectiveness of pool management operations especially when the data stream is huge.

The rest of paper is organized as follows: section 2 briefly reviews previous relevant researches. Section 3 explains the proposed algorithm accurately. Section 4 discusses the experimental evaluations and section 5 concludes the paper.

2 Related Works

Data stream classification has received much attention in recent years. The classification algorithms could be categorized into: single or ensemble. Examples of single classification algorithms for data streams are CVFDT [4] and SVM [5] and there are lots of ensemble approaches such as SEA [6], online bagging and boosting [7], DWM [8] and learn++.NSE [9]. The ensemble approaches differ in the way of forgetting classifiers and therefore data. However, none of these algorithms consider the problem of recurring concepts.

Recurring concepts algorithms have been of interest in recent years [2, 3,10-12]. These approaches try to extract the concepts from data and maintain all the concepts in a pool of concepts and classifiers. The most relevant approaches to the present paper are CCP framework [3] and PASC [2] which use an ensemble of classifiers, each of them representing a concept of the environment. The two algorithms are the same in extracting concepts from the batch of data, however, differ in the batch assignment and classification methods. To extract a concept from a batch $B_t = (x_{t,1},x_{t,2},\dots,x_{t,k})$ of instances with n features and labels $L_t=(l_{t,1},l_{t,2},\dots,l_{t,k})$, a Conceptual Vector is defined as $Z=(z_1,\dots,z_n)$ where z_i is calculated from:

$$z_i = \begin{cases} \{P(f_i = v|c_j): i = 1..n, j = 1..m, v \in V_i\} & if\ f_i\ is\ nominal \\ \{\mu_{i,j}, \sigma_{i,j} : j = 1..m\} & if\ f_i\ is\ numeric \end{cases} \tag{1}$$

where f_i is the i^{th} feature of the instances, c_j is a class label between m possible values and v is a nominal value for f_i in the set V_i. $\mu_{i,j}$ and $\sigma_{i,j}$ are the mean and standard deviation of the i^{th} feature of the instances in the j^{th} class. To compare the conceptual vectors and cluster them to detect recurring concepts, Euclidean distance measure is used:

$$ConDis(Z_A, Z_B) = \{dis(z_{1(A)}, z_{1(B)}) + \cdots + dis(z_{n(A)}, z_{n(B)})\}^{1/2}, \tag{2}$$

and distance function is calculated as:

$$dis(z_{i(A)}, z_{i(B)}) = \sum_{k=1}^{size(z_i)} (z_{ik(A)} - z_{ik(B)})^2, \tag{3}$$

where $z_{ik(X)}$ is the k^{th} element of $z_{i(X)}$. If the distance between the conceptual vector on the recent batch and a concept in the pool is less than a threshold, it means that the batch represents that concept and so the corresponding classifier in the pool will be updated by the batch instances. Otherwise a new classifier and concept is added to the pool. PASC introduces new distance measures between a labeled batch and a concept with thresholds which can be set more easily. Both algorithms have the shortcoming of not supporting the pool management operations. As a result, their performance will decrease significantly because of wrongly setting the threshold parameters or while facing large data sets.

3 Proposed Learning Algorithm

Our goal is to propose an ensemble method called Pool Management base Recurring Concepts Detection (PMRCD) to classify concept drifting data streams in the presence of recurring concepts. This method exploits the ideas presented in CCP framework method [3] and PASC method [2]. These methods maintain a pool of classifiers and update them according to consecutively received batches of data. Our method enriches these methods by providing a strong pool management and automatic parameter tuning strategies operating on the classifiers of the pool. In addition, some other changes have been made to achieve more accurate results.

In the proposed method (*Procedure 1*), we maintain a pool of ensembles. Each ensemble contains a number of classifiers and describes a concept of the nonstationary environment. Using an ensemble of classifiers for each concept enables us to manage the pool more effectively. The maximum number of concepts of the environment is assumed to be bounded by a parameter, *maxE*. In addition, the maximum number of classifiers of an ensemble is determined by a parameter, *maxC*. After receiving a batch of unlabeled data, $B_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,k})$, the ensemble pool is used to classify instances (line 3). A pivot classifier is maintained for each ensemble and the classification procedure uses the pivots of the ensembles in a weighted scheme to classify the instances. Assume the true labels of instances in a batch are presented as $L_t = (l_{t,1}, l_{t,2}, \ldots, l_{t,k})$. The labeled batch is given to a learner to construct a new classifier h_t (line 4). In line 5, a conceptual vector is made for B_t. This vector which was first introduced in [3], is a summarized vector for B_t and determines the concept which describes B_t. This vector is assigned to h_t as well. A similar vector is also assigned to an ensemble. The summarized vector of an ensemble is the average of the vectors of its constituting classifiers. These vectors are used while ensembles are being updated by a new classifier. Using these vectors, each ensemble can be interpreted as a cluster of points associated to its classifiers and the pool is then a combination of clusters. Line 6 computes the nearest concept of the pool to B_t (e_{best}) and the distance between them (d). A modified version of ConDis [3] named Magnitude based ConDis (MConDis) is presented to determine the distance between two concepts (a concept in the ensemble and the concept describing B_t). If d is less than a parameter, θ, h_t should be added to e_{best}. Adding a new classifier to an ensemble is dependent on whether the pool is full or not. If d is more than θ and the pool is not full, a new ensemble with only one member (h_t) will be added to the pool (line 10). However, if the ensemble pool is full, no new ensemble can be added. In this case, it is checked by a statistical hypothesis test, whether two ensembles of the pool should be merged or not (line 11). The test checks if the concepts describing B_t and e_{best} are similar with high probability. If the test becomes successful, the two nearest ensembles of the pool will be merged (line 12), the parameter θ will be increased according to the distance between the two nearest neighbors and a new ensemble will be added to the pool (line 13). If the test is not successful, h_t will be added to e_{best} (line 15). Line 16 tests if the classifiers of e_{best} can be grouped into two unlike clusters. In this case, a split operation will be done on e_{best} and the parameter θ is decreased according to the distance between the two clusters of e_{best}.

A combining procedure is needed in the merging and splitting operations while updating the pivot classifier of an ensemble and adding a classifier to a full ensemble. We present a simple procedure to combine two classifiers into one. Our algorithm uses naïve Bayes classifier, although other classifiers which could be combined in a similar way can be used. In this procedure, it is assumed that the instances which have made the classifiers may not be available. The other important part of algorithm is to make the pivot classifier for an ensemble and to use the pivots for classifying an instance. Then, we should discuss how to update the ensemble pool after receiving the correct labels of the current batch and to make the conceptual vector (CV) for a given batch of data and also how to measure the distance between two conceptual vectors. These will be discussed in the following subsections. In addition, the merging and splitting operations and automatic parameter tuning in these operations are discussed.

3.1 Combining Two Classifiers

Ensemble algorithms have already been discussed in the literature. But to best of our knowledge, all these algorithms classify instances by evaluating the results of one or some of the base classifiers. As the number of classifiers in an ensemble is assumed to be limited, the merging operation may not maintain all the classifiers of the merging clusters. Meanwhile, it is needed to retain the information of both clusters in a single cluster. In addition, a pivot classifier of the same type of the base classifiers is maintained for each ensemble. This classifier needs to be updated in the merging operation. So we present a procedure which combines two classifiers into one classifier of the same type. This procedure acts only on naïve Bayes classifiers and should support the case when the instances which have made the classifiers are not available. A similar situation occurs in the splitting operation which will be discussed later.

In the combining algorithm (*Procedure 2*), two classifiers C_1 and C_2 are combined and the resulted classifier is stored in C_{out}. If instances which have made one of the classifiers are available, they will be given as input to the other classifier to be updated (lines 1-4). Otherwise, the distributions of C_1 are updated according to the distributions of C_2 and then C_1 will become a classifier which has the information of both C_1 and C_2 (lines 6-20). In line 9, the class distribution is updated for each of the class attributes. In a naïve Bayes classifier for each pair of class attribute, c, and instance attributes, a_i, a distribution is maintained. According to the type of a_i, these distributions are updated. If a_i is nominal, for each of the possible values, v_i, in the domain of a_i, the maintained distribution will be updated according the number of instances in class c which their a_i attribute is v_i. If a_i is numeric, the maintained distribution of the pair a_i and c will be updated according to the information (e.g. mean and standard deviation) stored in C_2 and its number of training instances.

```
Input: $B_t$: Batches of the stream.
$L_{t,i}$: Labels of the instances of $B_t$
$maxE$: the max number of ensembles in the pool.
$maxC$: the max number of ensemble classifiers
Output: Predicted labels of instances $B_{t,i}$.
1  for $t$ = 1 to infinity
2    if $t \neq 1$
3      $PL_t \leftarrow Pool$.classify($B_t$);
4      $h_t \leftarrow$ make_classifier($B_t$, $L_t$);
5      $CV_t \leftarrow$ make_CV($B_t$, $L_t$);
6      $(d, e_{best}) \leftarrow (min, argmin_{e \in Pool}) MConDis\ (CV_t, m_e)$;
7    if $d < \theta$
8      add_to_ensemble($e_{best}$, $h_t$);
9    else if $size(Pool) < maxE$
10     $Pool \leftarrow Pool \cup$ {new ensemble($h_t$)};
11    else if size($e_{best}$) > 1 and $not$(similar($e_{best}$, $h_t$))
12     merge_NN($Pool$);
13     $Pool \leftarrow Pool \cup$ {new ensemble($h_t$)};
14   else
15     add_to_ensemble($e_{best}$, $h_t$);
16   if size($e_{best}$)>maxC/2 and can_split($e_{best}$)
17     split ($e_{best}$, $Pool$);
```

Procedure 1. PMRCD method

```
Input: $C_1$, $C_2$: two naïve Bayes classifiers.
Output: $C_{out}$: Combination of $C_1$ and $C_2$.
1  if data_available($C_1$)
2    update_classifier($C_2$, dataof($C_1$));
3  else if data_available($C_2$)
4    update_classifier($C_1$, dataof($C_2$));
5  else
6    $n_1 \leftarrow C_1$.numInstances();
7    $n_2 \leftarrow C_2$.numInstances();
8    for each $c \in$ classes
9      $C_1$.classDistr().addvalue($c$, $n_2$);
10     for each $a_i \in$ attributes($C_2$)
11       $distr1 \leftarrow C_1$.distribution($a_i$, $c$);
12       $distr2 \leftarrow C_2$.distribution($a_i$, $c$);
13       if isNominal($a_i$)
14         $count \leftarrow distr2$.numInstances($v_j$);
15         for each $v_j \in$ values($a_i$)
16           $distr1$.addvalue($v_j$, $count$);
17       else
18         $distr1$.addvalue($distr2$.info(), $n_2$);
19 $C_1$.setNumInstances($n_1$+$n_2$);
20 $C_{out} \leftarrow C_1$;
```

Procedure 2. Combining two naïve Bayes classifiers into one classifier

3.2 Classification of Instances

The first task after receiving an unlabeled batch is to classify its instances. A pivot classifier is maintained for each ensemble in the pool. These classifiers will be updated during the execution of the procedure and will be used in a weighted scheme to classify batch of instances. The more relevant the pivot classifiers are to the concept of the current batch, the greater weights they are assigned.

PASC weighting scheme [2] is used here. In PASC, one classifier is maintained for each concept. A classifier in PASC is equivalent to an ensemble in our method (both describe a concept). PASC weights the votes of classifiers of different concepts and update them during classification of the batch and after the classification of a specified number of instances. The weights of the classifiers which do not classify each of the selected instances correctly will be multiplied by a factor β in the interval (0,1). The updating rule of ensemble weights for the selected subsample instance, $B_{t,j}$, will be as:

$$w'_{e_j} \leftarrow w_{e_j} * \beta^{M(j,i)}, \tag{4}$$

where w'_{e_j} and w_{e_j} are the weights of the j^{th} ensemble after and before the update. $M(j,i)$ is 0 if the j^{th} ensemble correctly classifies $B_{t,i}$, and 1 otherwise.

In [2], it is discussed that the accuracy of the algorithm using the majority vote of classifiers with the aforementioned weights will converge to the accuracy of the best classifier which is the ensemble with the least error on the last batch in our method, if the size of the batch is large enough.

The weights of ensembles are set in the beginning of the process of a batch so that the ensembles with higher accuracies on the previous batch of data will have higher weights. The weights are set as:

$$w^0_{e_j} \leftarrow (\frac{1}{\beta})^{2^{A(j)}}, \tag{5}$$

where $A(j)$ is the accuracy of the j^{th} ensemble on the selected subsample of the last batch. Finally, for the sake of efficiency, the classifier with the highest weight is used for the classification and the instances which are not used for the updating task are classified only by one classifier.

3.3 Updating the Ensemble Pool

After receiving true labels, the pool will be updated. First, a new classifier, h_t, is made. Then, a summarized vector for the current batch is constructed (line 5 of *Procedure 1*). Finally, h_t will be added to the pool and the pool will be updated accordingly (lines 6-17). It is discussed in the next subsections how to construct the summarized vector and measure the distance between two vectors and also how to update the pool.

Summarized Vector of a Batch. The summarized vector of a batch is used to determine the concept that it has been drawn from. So batches of data with similar concepts will have summarized vectors similar to each other. Also a distance measure should be defined between two summarized vectors. Conceptual vector and ConDis, defined in (2), were first developed to satisfy these properties. A modified version of the distance (Magnitude based Conceptual vector) between conceptual vectors is used here. The Magnitude based Conceptual Distance (MConDis) between two batches A and B modifies $dis(z_{ik(A)}, z_{ik(B)})$ in (3). If f_i is nominal, $dis(z_{ik(A)}, z_{ik(B)})$ will be the difference of these values. But if f_i is numeric, this distance will be defined as:

$$dis(z_{ik(A)}, z_{ik(B)}) = \begin{cases} 0, & if\ z_{ik(A)} * z_{ik(B)} \le 0 \\ \dfrac{z_{ik(A)} - z_{ik(B)}}{z_{ik(A)} + z_{ik(B)}} & otherwise' \end{cases} \tag{6}$$

In ConDis there is no difference between nominal and numeric attributes, but it may cause some problems when the range of these attributes is very large or small and so a modified definition is presented here.

Adding the New Classifier to an Ensemble. The nearest ensemble to the current batch and its distance is computed as e_{best} and d, respectively. MConDis between the CV of the current batch and the mean of the ensembles is used as a measure of distance comparison. If d is less than θ, h_t will be added to e_{best}(line 8 of *Procedure 1*). *Procedure 3* shows how to add a new classifier to an ensemble. If the number of ensemble classifiers is less than *maxC*, h_t will be added to the ensemble. The mean of the ensemble will be updated as well. Otherwise, the nearest neighbor to h_t(i.e. h_N) will be found (line 4). h_N will be combined with h_t and the new classifier will be replaced in the ensemble. Note that the CV of the combined classifier will be the weighted average of the combining classifiers with respect to the number of instances used to train them. The mean and pivot classifier of the ensemble will be updated in this case, too.

Adding a New Ensemble to the Pool. If the distance (d) between the new classifier and the nearest ensemble (e_{best}) is more than θ, it could not be added to e_{best}. If the pool is not full, a new ensemble with the new classifier as its only member will be added to the pool (line 10 of procedure 1).

```
     Input:   $e$: an ensemble of the pool
              $h_t$: a new classifier
     Output:  $e$: updated ensemble
1    if size($e$) < $maxC$
2        $e \leftarrow e \cup \{h\}$
3    else
4        $h_N \leftarrow argmin_{h \in e} MConDis(h_N, h_t)$;
5        $h \leftarrow$ combine($h_N$, $h_t$);
6        $CV_h \leftarrow$ wavg($CV_{h_N}, CV_{h_t}$);
7        $e \leftarrow e$.replace($h_N$, $h$);
8        $p_e \leftarrow$ combine($p_e$, $h$);
```

Procedure 3. Adding a classifier to an ensemble

Merging Operation. If the distance d is more than θ and the ensemble is full, the two conditions checked in lines 7 and 9 (*Procedure 1*) will not be satisfied. So the concept describing the new classifier is not similar enough to any other concepts of the pool and the merging condition is checked in line 11. As it is assumed that the number of environment concepts is at most *maxE* and the new concept is different from all existing ones, it may be the case that two ensembles of the pool describe a single concept. If so, these two concepts should be merged with each other and a new ensemble should be added to the pool with the new classifier as its only member. In this case, a hypothesis test can be used to decide whether the concept of the new classifier and e_{best} are different from each other with high probability or not. If the test becomes successful, the new classifier should not be added to e_{best} with high confidence. So two of the concepts should be merged and a new ensemble with this classifier as its only member should be added to the pool (lines 12-13 of *Procedure 1*).This can be rewritten in *Procedure 4* with more details. First it is checked whether the number of the classifiers of e_{best} is more than 1 (line 1). The hypothesis test is done as follows: An interval i_1 with high confidence *M%* is found for s_1, the same prediction probability of h_t and e_{best}on B_t. In addition, an interval i_2 with high confidence *N%* is found for s_2, the same prediction probability of the concepts describing e_{best} on B_t. For this reason, e_{best} is split into two parts of almost equal size and the same prediction probability of these two parts is computed. If the intervals i_1 and i_2 have no intersection with each other, then the two nearest ensembles of the pool will be merged together. s_1 can be estimated as a normal variable with mean equal to $\hat{s}_1$ and standard deviation σ_1. Therefore, s_1 will be in the interval:

$$i_1 = \left(\hat{s}_1 - Z_{\frac{1+M}{2}}\sigma_1, \hat{s}_1 + Z_{\frac{1+M}{2}}\sigma_1\right), \tag{7}$$

with probability $M\%$. The same discussion holds for the variable i_2. If the intervals i_1 and i_2 have no intersection, the two nearest ensembles of the pool are merged together (lines 10-13). To merge the two ensembles, e_{m1} and e_{m2}, e_{m2} is removed from the pool and all of its classifiers are added to e_{m1}. Finally, θ is increased to the distance between the two merged ensembles. Because these two ensembles describe the same concept and so their distance should be less than θ.

Let us assume that if the concept of a hypothesis is the same as e_{best}, the same prediction probability of this concept and e_{best} will fall into the interval i_2. Then we can state:

Lemma 1. The probability that the merging operation (*Procedure 4*) is done incorrectly is at most (200-*M*-*N*)%.

Proof. It is assumed that the number of environment concepts is at most *maxE* and the pool is full. If the nearest concept to h_t differs from it, a new concept must be added for h_t in the pool. Therefore, there are two ensembles describing one concept and a merging operation is needed. Hence, the probability that the merging operation is incorrect is at most equal to the probability that s_1 or s_2 falls outside the intervals i_1 and i_2. This probability is the union of the two probabilities: the probability that s_1 falls outside i_1 and the probability that s_2 falls outside i_2. Therefore, the probability that the merging action is incorrect is at most:

$$(100 - M)\% + (100 - N)\% = (200 - M - N)\%, \tag{8}$$

This probability is very low, because *N* and M almost equal to 100. ■

Splitting Operation. After adding the new classifier h_t to e_{best}, it is checked whether e_{best} can be split into two groups or not (lines 16-17 of *Procedure 1*). The splitting procedure can be rewritten as *Procedure 5*. To split an ensemble, it should have at least some number of classifiers (line 1 of *Procedure 5*). Then, two parts of e_{best} which are almost far from each other are constructed (lines 2-4). The farthest classifier to the first classifier of the ensemble is found and named h_{far} (line 2) and the farthest classifier to h_{far} is found as h'_{far} (line 3). Then e_{best} is split into two groups of classifiers e_1 and e_2 based on their distances to h_{far} and h'_{far} (lines 4-5). The same prediction probability of e_1 and e_2 on B_t, named as s_3, can be estimated by a normal variable with mean $\hat{s}_3$ and σ_3. It can be stated that s_3 is less than:

$$MI \leftarrow \hat{s}_3 + Z_L\sigma_3, \tag{9}$$

with probability $L\%$. If *MI* (Maximum Interval) is less than a parameter M_s, the splitting operation is done as follows: First, e_2 is removed from e_{best} and if the pool is not full, e_2 is added to the pool as a new ensemble. Else, all the classifiers in e_2 are added to the nearest ensemble of the pool. Finally θ is decreased to the distance between two parts of e_{best}. Because these two parts cannot be in an ensemble and so their distance should be more than θ.

4 Experimental Results

In this section, we first introduce the data sets we used in the experiments. Second, we compare our method with the CCP framework, one of the best methods in the domain of data streams and recurring concepts. Finally, the effect of merging and splitting operations is discussed.

4.1 Data Sets

In order to evaluate the proposed method we used two data sets. The first one is the Sensor data set which is a very large real data set. The second is the Moving Hyper Planes data set which we created using the MOA stream generator [13].

Sensor Data Set. This data set contains the ordinal information collected by 54 sensors deployed in Intel Berkeley Research laboratory in a period of two months. The attributes of the data set contain the temperature, humidity, light and voltage measured by the sensors and also the time the measurement has been done. The class label is the sensor ID. This data set contains 54 classes, 5 attributes and 2,219,803 instances. The concept drift in this data set is because of the changes that occur in the attributes such as lightening a bulb or the temperature changes during a day[14].

Moving Hyper Plane Data Set. A Hyper Plane determines a decision surface characterized by $g(\vec{x}) = \vec{w}.\vec{x} = 0$, where $\vec{w}$ determines the orientation of the surface and $\vec{x}$ is an instance of the space. Changing the orientation of the hyper plane gradually or suddenly will simulate gradual or sudden concept drifts. The data set contains 800,000 instances and 10 numeric attributes. There is a sudden concept drift after each 200,000 instances. The concepts reappear after the first half of the instances.

```
if size($e_{best}$) > 1
  $\hat{s}_1 \leftarrow$ same_prob($e_{best}, h_t, B_t$);
  $\sigma_1 \leftarrow \sqrt{\hat{s}_1 * (1 - \hat{s}_1)/K}$;
  $I_1 \leftarrow (\hat{s}_1 - Z_{\frac{1+M}{2}}\sigma_1, \hat{s}_1 + Z_{\frac{1+M}{2}}\sigma_1)$;
  ($e_1$, $e_2$)= split($e_{best}$);
  $\hat{s}_2 \leftarrow$ same_prob($e_1, e_2, B_t$);
  $\sigma_2 \leftarrow \sqrt{\hat{s}_2 * (1 - \hat{s}_2)/K}$;
  $I_2 \leftarrow (\hat{s}_2 - Z_{\frac{1+N}{2}}\sigma_2, \hat{s}_2 + Z_{\frac{1+N}{2}}\sigma_2)$;
  if (not has_intersect($I_1$, $I_2$))
    $(e_{m1}, e_{m2}) \leftarrow argmin_{e_1,e_2}(MConDis(e_1, e_2))$;
    $Pool \leftarrow Pool \setminus \{e_{m2}\}$;
    for each $h \in e_{m2}$
      add_to_ensemble ($e_{m1}$, $h$);
    $Pool \leftarrow Pool \cup$ {new ensemble($h_t$)};
    $\theta \leftarrow$ max ($\theta$, MConDis($m_{e_{m1}}, m_{e_{m2}}$));
```

Procedure 4. Merging operation

```
if size($e_{best}$) > $macC/2$
  $h_{far} \leftarrow argmax_{h \epsilon e_{best}}(MConDis(h_{1_e}, h))$;
  $h'_{far} \leftarrow argmax_{h \epsilon e_{best}}(MConDis(h_{far}, h))$;
  $e_1 \leftarrow \bigcup_{h \epsilon e_{best}} MConDis(h, h_{far}) < MConDis(h, h'_{far})$;
  $e_2 \leftarrow e_{best} \setminus e_1$;
  $\hat{s}_3 \leftarrow$ same_prob($e_1, e_2, B_t$);
  $\sigma_3 \leftarrow \sqrt{\hat{s}_3 * (1 - \hat{s}_3)/K}$;
  $MI \leftarrow \hat{s}_3 + Z_L\sigma_3$;
  if $MI < M_s$                                  /*split test*/
    $e_{best} \leftarrow e_{best} \setminus e_2$;
    if size($Pool$) < $maxE$
      $Pool \leftarrow Pool \cup e_2$;
    else
      for each $h \in e_2$
        $e_N \leftarrow argmin_{e \epsilon Pool} MConDis(m_e, CV_h)$;
        add_to_ensemble($e_N$, $h$);
        $\theta \leftarrow$ min ($\theta$, MConDis($m_{e_1}, m_{e_2}$));
```

Procedure 5. Splitting operation

4.2 Parameter Tuning

There are several parameters used in our method that need to be set. But setting them is straight forward. We will see in the next subsections that the primary value of θ is not important. In fact, this parameter will be updated to the right value during the execution of the algorithm. The parameter *maxE* is set to 10 in both methods as we estimate that 10 concepts are sufficient to describe the data sets. The parameter *maxC* is set to 5 for the Sensor data set and 3 for the Moving Hyper Plane data set. Another parameter is the size of the batch which is set to 50 for both data sets. *M*, *N* and *L* which are used in the merging and splitting operations and need to be near 100, are set to 95, 95 and 97.5 respectively. Another parameter which is needed in the splitting operation is the *MI* parameter which is set to 0.5 for the Sensor data set and 0.7 for the Moving Hyper Plane data set. Finally, the β parameter is set to 0.1 for both data sets.

4.3 Evaluating the PMRCD Method

In this section, the accuracy of the proposed method is compared with the CCP framework method. Parts (a) and (b) of *Figure 1* show the accuracies of these methods on the Sensor and Moving Hyper Planes data sets, respectively. The batch size and the concept number parameters of the CCP framework method are set the same as our method. There is only a parameter, θ, which needs to be set for CCP method and is set to 2 for the Sensor data set and 0.7 for the Moving Hyper Plane data set. The value of the parameter θ of our method is set to 15 for this experiment.

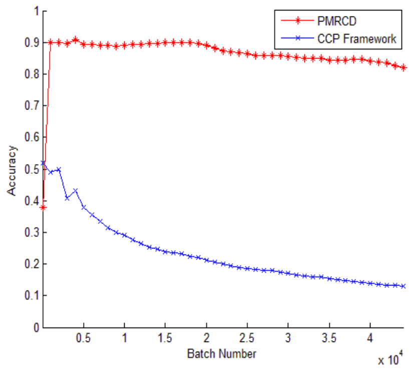
The first part of *Figure 1* shows the results for the Sensor data set. It can be seen that the accuracy of our method is about 70 percent more than the CCP framework method. In fact, the accuracy of the CCP framework method and the similar ensemble methods without pool management operations decrease after processing the first batches of instances. The reason is that the batch assignment methods to an existing concept will not work well and the pool management operations are needed. The same results can be seen for the second data set, except that the accuracy of the CCP framework method is not far below ours. This is because of the small size of the Moving Hyper Plane data set in comparison with the Sensor data set. Therefore, the decrease in the performance is less than the first data set.

4.4 Evaluating the Pool Management Operations

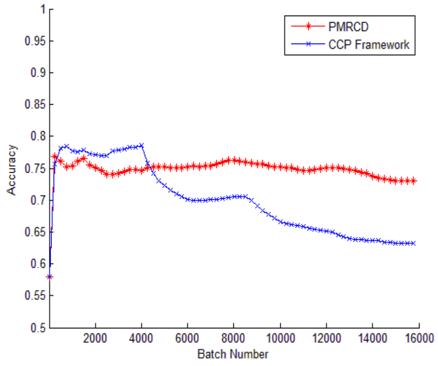
Parts (a) and (b) of *Figure 2* show the accuracies of the proposed method for different values of the parameter θ and for the different pool management operations on the two data sets. To evaluate the pool management operations, all possible cases are examined. The first case is the one which was presented in PMRCD, i.e. using the both merging and splitting operations. The second case uses only the merging operation. The third case uses only the splitting operation. The last case uses none of the merging and splitting operations. When only one of the splitting or merging operations is used, we do not change the parameter θ. Because the splitting operation will always decrease the parameter and the merging operation will always increase it. So the value will become so low or high and this will lead to a low performance. Therefore, it is preferable not to change the parameter in these cases.

The first part of *Figure 2* shows the results for the Sensor data set. The primary value of the parameter θ is changed from 0.1 to 18.1 with step size 3 which covers a wide range of values for this data set. Our experiments show that the regular value of MConDis between two concepts is in the range [14, 16] and no distance is more than 18 or less than 10. So the range [0.1, 18.1] covers all reasonable values for θ. It can be seen that using the both operations will lead to the best performance regardless of the primary value of θ. In addition, the results are almost the same for all values of θ. It is because of the quick adaptation of the parameter to the right value during the execution of the algorithm. Using only one of the operations will lead to a performance less than the first case. In addition, the merging operation is more important for lower primary values of θ and the splitting operation is more important for higher values of this parameter. Also, it can be seen that when the primary value is very high and no split is done, the performance will be very low. It is because of the fact that in this case, only one ensemble will be added to the pool and so only one concept will be detected. Finally, using none of the pool management operations will lead to a very low accuracy. This means that no primary value of θ can guaranty the right assignment of batches to the concepts during the execution of the algorithm.

The second part of *Figure 2* shows the results for the Moving Hyper Plane data set. θ is changed from 0.1 to 1.9 with step size 0.2 which covers a proper range for this data set. Similar arguments hold for this case. But for some primary values of θ, the results of the method with no pool management operation are better than using only one of them. This is because of the small size of this data set. In fact, the effectiveness of the pool management operations will be clearer for larger data sets. But still it can be seen that using the pool management operations will be the best choice in this case, too. In addition, the merging operation is necessary for lower primary values of θ and the splitting operation is necessary for upper values.

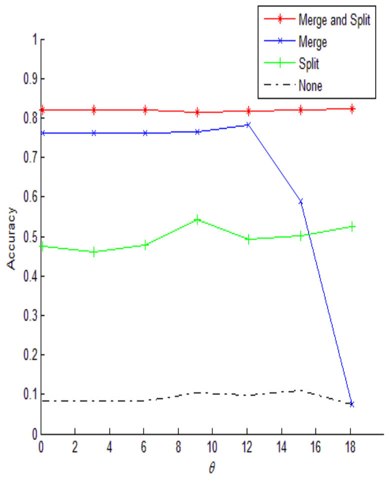


(a) Sensor data set

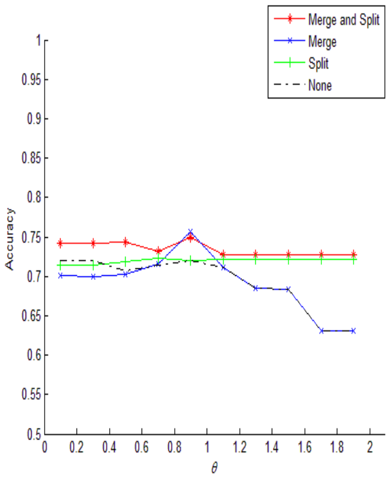


(b) HyperPlane data set

Fig. 1. Comparison of PMRCD method with CCP framework



(a) Sensor data set



(b) HyperPlane data set

Fig. 2. Comparison of θ paramater

5 Conclusion and Future Works

We proposed an ensemble method to classify streaming data in the presence of concept drift and recurring concepts. This method maintains a pool of ensembles of classifiers. Each ensemble represents a concept. After receiving a batch of instances, it is classified using the ensemble pool. Then the true labels arrive and a new classifier is made for the labeled batch of instances. This classifier is added to an existing ensemble of the pool or a new ensemble. The ensembles of the pool can be merged together or split in case of wrong assignment of the concepts to them. Our method improves the accuracy of similar methods significantly, especially in case of large data sets.

We believe that future further study on this method and similar methods is required. The pool of classifiers can be managed even more effectively noting the fact that each ensemble can be interpreted as a cluster having the classifiers as its points. So, stream clustering algorithms could be used. More complicated structures, such as hierarchical concepts could be handled using hierarchical and cascade classifiers. In addition, more experiments could also be done on wider range of real world data sets.

Acknowledgement. This work was supported by Iran Telecommunication Research Center.

References

[1] Tsymbal, A.: The Problem of Concept Drift: Definitions and Related Work (2004)
[2] Hosseini, M.J., Ahmadi, Z., Beigy, H.: Pool and Accuracy Based Stream Classification: A new ensemble algorithm on data stream classification using recurring concepts detection. In: Proceedings of International Conference on Data Mining, HaCDAIS Workshop (2011)

[3] Katakis, I., Tsoumakas, G., Vlahavas, I.: Tracking recurring contexts using ensemble classifiers: an application to email filtering. Knowledge and Information Systems 22(3), 371–391 (2009)

[4] Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, San Francisco (2001)

[5] Klinkenberg, R., Joachims, T.: Detecting Concept Drift with Support Vector Machines. In: Proceedings of the Seventeenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. (2000)

[6] Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, San Francisco (2001)

[7] Oza, N.C.: Online ensemble learning, PhD Thesis, University of California, Berkeley (2001)

[8] Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research (8), 2755–2790 (2007)

[9] Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks (99), 1517–1531

[10] Gama, J., Kosina, P.: Tracking Recurring Concepts with Meta-learners. In: Lopes, L.S., Lau, N., Mariano, P., Rocha, L.M. (eds.) EPIA 2009. LNCS, vol. 5816, pp. 423–434. Springer, Heidelberg (2009)

[11] Gomes, J.B., Menasalvas, E., Sousa, P.A.C.: Tracking Recurrent Concepts Using Context. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 168–177. Springer, Heidelberg (2010)

[12] Lazarescu, M.M.: A Multi-Resolution Learning Approach to Tracking Concept Drift and Recurrent Concepts. In: 5th IAPR Workshop on Pattern Recognition in Information Systems (PRIS), Miami, USA, pp. 52–61 (2005)

[13] Bifet, A., et al.: Moa: Massive online analysis. The Journal of Machine Learning Research (11), 1601–1604

[14] Zhu, X.: Stream Data Mining repository (2010), `http://www.cse.fau.edu/~xqzhu/stream.html`

Extrapolation Prefix Tree for Data Stream Mining Using a Landmark Model

Yun Sing Koh[1], Russel Pears[2], and Gillian Dobbie[1]

[1] Department of Computer Science, University of Auckland, New Zealand
{ykoh,gill}@cs.auckland.ac.nz,
[2] School of Computing and Mathematical Sciences, AUT University, New Zealand
rpears@aut.ac.nz

Abstract. Since the introduction of FP-growth there has been extensive research into extending its usage to data streams or incremental mining. This task is particularly challenging in the data stream environment because of the unbounded nature of a data stream and the need for avoiding multiple scans of the data. In this paper, we propose an algorithm, Extrapolation Prefix Tree that extracts frequent itemsets using a landmark windowing scheme. The algorithm uses a prefix tree structure to store arriving transactions, but unlike previous approaches estimates the structure of the tree in the next block of data based on the arrival pattern of items appearing in transactions that arrive in the current block. Our experimentation shows that Extrapolation-Tree significantly outperforms the CP-Tree, both in terms of the number of updates and the execution time required to keep the tree current while maintaining a compact tree.

Keywords: Landmark Model, Data Stream, Prefix Tree.

1 Introduction

Most traditional frequent pattern mining only cope with static datasets, which provide snapshots in time of the patterns found. With the growing importance of data streams, mining frequent patterns from data streams has become an important and challenging problem for a range of applications including real-time surveillance systems, communication networks, Internet traffic, online transactions in the financial market or retail industry, electric power grids, and remote sensors.

Unlike traditional data sets, a data stream consists of continuous, unbounded, data elements usually arriving with high speed and a data distribution that often changes with time. Due to the nature of data streams, there are some inherent challenges in mining data streams, particularly in high speed environments. Due to the sheer volume of data arriving each data element of a stream can be examined at most once. Moreover, the memory usage in the process of mining data streams has to be bounded by the amount of primary storage available despite the fact that new data elements are continuously generated from the source. A mine anytime approach should be available, in order to be able to capture the most recent patterns. Considering the characteristics above, an efficient data stream mining technique should require only one scan of the streaming data and should maintain as compact a data structure as possible.

In terms of frequent pattern mining, the key issue is the maintenance of a compact data structure that contains an up to date version of frequent patterns. Despite extensive research in the area of frequent pattern mining, a number of issues remain unresolved. Among them are: (1) how to detect when the concept has drifted to an extent that requires reorganization of the structure used to store patterns; (2) an accurate but inexpensive estimation mechanism that will predict the data structure required in the next block based on changes observed in the stream in the current block of transactions. In this paper we propose a scheme for determining when updates are required, enabling us to defer updates at blocks when stability is observed in the stream, thus avoiding unnecessary and expensive updates. In addition, we implement an accurate but efficient linear estimation mechanism that is used whenever our concept drift detection mechanism signals that changes to the tree structure are required. The estimation scheme anticipates changes to the structure of the tree in advance, rather than simply assuming that the state of the stream in the next block is the same as the state in the current block, as in the CP-Tree approach [8]. An accurate assessment of the next state has a flow-on effect on subsequent states, minimizing the changes required to the tree, improving the overall tree maintenance time.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. Section 3 contains the preliminary details. In Section 4, the Extrapolation Prefix Tree algorithm is presented. Our experimentation with Extrapolation Tree and comparison with CP-Tree is presented in Section 5. Finally we summarize our research contributions in Section 6 and outline directions for future work.

2 Related Work

In a landmark window model, mining is performed on the transactions between a specific point in time called the landmark and the current time. The initial attempt to mine frequent patterns over the entire history of streaming data was proposed by Manku and Motwani [7]. They proposed two single-pass algorithms, Sticky-Sampling and Lossy Counting, both of which are based on the anti-monotone property; these algorithms provide approximate results with an error bound. Li et al. [5,6] proposed DSM-FI and DSM-MFI to mine frequent patterns using a landmark window. Each transaction is converted into k small transactions and inserted into an extended prefix-tree-based summary data structure called the item-suffix frequent itemset forest. Yu et al. [9] proposed an efficient algorithm to mine false negative or false positive frequent itemsets from high speed transactional data streams using the Chernoff bound. This approach uses a running error parameter to prune itemsets and a reliability parameter to control memory usage.

Most of the methods discussed above allow us to find approximate frequent patterns within a guaranteed error bound or require an additional pruning threshold. Up until now only a few proposed approaches[3,1,4,8] were designed to find the exact set of recent frequent patterns from a data stream using a landmark window model. We will discuss two existing FP-tree based algorithms that mine exact sets of patterns, namely the (i) CanTree algorithm, and (ii) CP-Tree.

Leung et al. [4] proposed the Canonical-Ordered Tree for stream mining. This algorithm is designed so that it only requires one dataset scan. In CanTree, items are arranged in some canonical order, which can be determined by the user prior to the mining process or at runtime during the mining process. The items are arranged according to a prefix tree structure, thus unaffected by the item frequency. CanTree generates compact trees if and only if the majority of the transactions contain a common pattern-base in canonical order. Otherwise, it may generate skewed trees with too many branches and hence with too many nodes, thus impacting on both memory usage and tree construction time.

Tanbeer et al. [8] proposed a tree structure, called CP-Tree that constructs a compact prefix structure. CP-Tree has a frequency descending structure that captures part by part data from the dataset and dynamically restructures itself. The construction operation consists of two phases: insertion phase and restructuring phase. In the insertion phase transactions are inserted into the CP-Tree according to the current sorted order of the item list while updating the frequency of items contained within an item list. The restructuring phase rearranges the list according to frequency-descending order of items and restructures the tree nodes according to the new ordered item list.

One major disadvantage of these techniques is that they use data from previous blocks to build the prefix tree for the next block. These techniques work under the assumption that drift within the stream remains fairly constant, an assumption which is not always true in practice. The relaxation of this assumption requires both a drift detection mechanism as well as an estimation mechanism which our proposed approach implements.

3 Preliminaries

For a landmark model, a transaction data stream $S = \{W_1, W_2, \ldots, W_n\}$ is an infinite sequence of basic blocks, where n is the block identifier of the latest block, W. Online mining of frequent itemsets utilizing a landmark window model for data streams involves extracting the set of all frequent itemsets from the transactions between a specified block identifier, called a landmark and the current block identifier n. The frequency of an itemset, X, in W, denoted as $count_W(X)$, is the number of transactions in W that support X. The support of X in W, denoted as $supp_W(X)$, is defined as $\frac{count_W(X)}{N}$, where N is the total number of transactions received in W. X is a frequent itemset in W, if $supp_W(X) \geq minsup$ support threshold.

Given a transaction data stream and a minimum support threshold, *minsup*, the problem of frequent itemset mining over a window, W, in the transaction data stream is to find the set of all itemsets whose support is greater than *minsup* when measured between a given landmark L and the current block, W. To mine frequent items in a data stream, it is necessary to keep not only the frequent, but also the infrequent itemsets, as an infrequent itemset can potentially become a frequent itemset later in the stream.

3.1 Support Ordered Prefix Trees

The Support ordered tree was designed to optimize mining time. In a support ordered tree nodes are arranged in descending order of support. For a given minimum support threshold *minsup*, frequent patterns can very efficiently be extracted by scanning the tree in a top-down fashion starting from the root and terminating the search at each node where the support falls below the minsup value. However, the challenge with growing a support ordered tree is the maintenace required to preserve the tree in suppport order when the pattern of occurrence of items relative to each other changes continuously in time.

In order to deal with this issue Tanbeer et al. proposed a periodic reorganization of the tree. Figure 1 illustrates the growth of the support ordered tree with a reorganization interval of 3 transactions. A list (I-list) that indexes each item by its support value is the key data structure used to drive the process of tree growth. Fig 1 (b) shows the I-list based on item occurrence in the first block of 3 transactions. This I-list is used to build the tree in the next block comprising transactions 4 to 6 and the resulting tree is shown in Fig 1 (c) . During the arrival of the next 3 transactions the tree is grown based on the I-list from the previous block. However, as can be seen from Fig 1 (c) the prefix tree is not in support order and conversion to support order requires resorting the IS-list and restructuring as in Fig 1 (d) and (e) respectively. This simple example illustrates the drawback of the support ordered approach: tree growth based on the support order from the previous block incurs potentially heavy overheads in resorting the index, and more importantly restructuring a large portion of the tree during the reorganization phase. These overheads can be greatly reduced by anticipating the support order of the items and growing the tree using this anticipated order instead of an outdated order from the previous block. If the anticipated order is very close to the actual support order then the restructuring overheads can be minimized. This the rationale behind the extrapolation prefix tree approach that we propose in this research.

4 Extrapolation Prefix Tree (Extrapolation-Tree)

In this section we discuss the construction of our Extrapolation Prefix tree. Our approach has two phases: Tree Construction Phase and Tree Mining Phase.

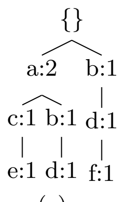
The Tree Construction phase can be broken down into two additional phases, Insertion Phase and Reorganization phase. In the Insertion phase, items in a transaction are inserted into the tree based on an estimated support order, as described in Section 3.1. The tree is then reorganized after a predefined interval,

Tid	Trans
1	b a c
2	d a b
3	e c f
4	e c
5	f c
6	c e a

(a)

Item	Support
a	2
b	2
d	1
c	1
e	1
f	1

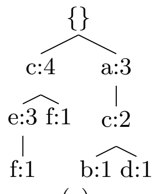
(b)



(c)

Item	Support
c	4
a	3
e	3
f	2
b	2
d	1

(d)



(e)

Fig. 1. Growth and Maintenance of a Support Ordered Tree (a) is the dataset, (b) is the IS-list after arrival of the first 3 transactions, (c) is the tree grown in first block, (d) is the IS-list built on second block and (e) is the tree grown on the updated IS-list in (d)

typically the block size. However to reduce the need for unnecessary reordering we implement a drift detection mechanism to ensure that the size of a tree has increased sufficiently to entail a reorganization. This is particularly useful as transactions have been inserted into the tree based on a forecast order, thus intuitively the need for reorganization at every block can be expected to reduce. We also introduced a delay condition whereby items are only subjected to the extrapolation process when the delay condition is not triggered. We discuss the delay condition in detail in Section 4.3. The Tree Mining phase follows the FP-Growth mining technique. Once the Extrapolation-Tree is constructed we use FP-Growth to mine patterns with support above a user defined minsup.

4.1 Extrapolation Functions

Given the information of an item in a current block, W_n, we estimate the support for a given item for new incoming data. This allows us to predefine a prefix-order for the tree. To approximate the support of an item there are two cases that need to be handled. The first case is when a given item is unaffected by drift, and the second corresponds to the situation where drift causes a change in support. In this section we propose two functions to handle these cases. To infer the support of an item in the next block, W_{n+1} we track the support of an item within the current block W_n. We divide the block W_n into two halves. We then track the change in support of an item between the first half of the block denoted by $supp_{W_n[1,\frac{w}{2})}(X)$ to the second half of the block denoted by $supp_{W_n[\frac{w}{2},w)}(X)$, where w is the block length. The change in support is then given by:

$$\Delta supp_{W_n}(X) = supp_{W_n[\frac{w}{2},w)}(X) - supp_{W_n[1,\frac{w}{2})}(X) \tag{1}$$

In order to assess which values of $\Delta supp_{W_n}(X)$ are significant we note that the function is a difference between the sample means of two random variables which are drawn from unknown distributions. In order to assess significance between the means we make use of the Hoeffding bound [2]. The Hoeffding bound is attractive as it is independent of the probability distribution generating the observations. The Hoeffding bound states that, with probability $1-\delta$, the true mean of a random variable r is at least $\overline{r}-\epsilon$ when the mean is estimated over t samples, where

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2t}}$$

R is the range of r. In our case the variable r is denoted by $|\Delta supp_{W_n}(X)|$, which has a range value R of 1, and the number of samples $t = w$, the block size. Thus for any value of $|\Delta supp_{W_n}(X)| \leq \epsilon$, our assumption of no concept drift holds, and the estimated support, $supp^{extrap}_{[W_l,W_{n+1}]}(X)$ of an item X at the $(n+1)^{th}$ block measured relative to the landmark block W_l is given as follows:

$$supp^{extrap}_{[W_l,W_{n+1}]}(X) \approx supp_{W_n}(X) = \frac{count_{[W_l,W_n)}(X) + count_{W_n}(X)}{N} \tag{2}$$

where N represents the number of transactions that arrived between W_l to W_n.

However for any value of $|\Delta supp_{W_n}(X)| > \epsilon$ our assumption of no concept drift is violated and an estimate of the support of item X in the $(n+1)^{th}$ block, $supp^{extrap}_{W_{n+1}}(X)$ is required.

We can now formulate the support of item X, $supp^{extrap}_{[W_l,W_{n+1}]}(X)$ as:

$$supp^{extrap}_{W_{n+1}}(X) = \begin{cases} \max(supp^{extrap'}_{W_{n+1}}(X), 0) \text{ if } \Delta supp_{W_n}(X) < 0 \\ \min(supp^{extrap'}_{W_{n+1}}(X), 1) \text{ if } \Delta supp_{W_n}(X) > 0 \end{cases} \tag{3}$$

where $supp^{extrap'}_{W_{n+1}}(X)$ is estimated using linear extrapolation. Under the linear assumption, the difference between the support of item X at the end of the $(n+1)^{th}$ block and its support at the end of the n^{th} block must be twice the difference in its support between the two halves of the previous block n. Thus we have:

$$\frac{supp^{extrap'}_{W_{n+1}}(X) - supp_{W_n[\frac{w}{2},w)}(X)}{|W_{n+1}|} = \frac{supp_{W_n[\frac{w}{2},w)}(X) - supp_{W_n[1,\frac{w}{2})}(X)}{|W_n|/2} \tag{4}$$

where

$$|W_{n+1}| = |W_n| = w$$

giving,

$$supp^{extrap'}_{W_{n+1}}(X) = 3supp_{W_n[i+\frac{w}{2},i+w)}(X) - 2supp_{W_n[i,i+\frac{w}{2})}(X). \tag{5}$$

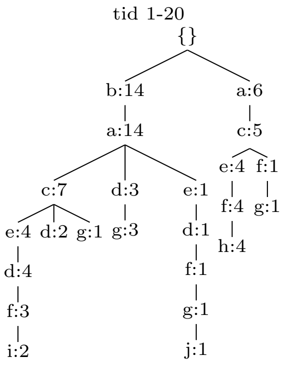
The rationale behind Eq 3 is that $\Delta supp_{W_n}(X)$ from Eq 1 represents the change in item X. If $\Delta supp_{W_n}(X) > 0$, it represents an increasing occurrence of item X within a stream, and thus any forecasted value for support will need to be bounded above by 1. Likewise, if $\Delta supp_{W_n}(X) < 0$, it represents a decreasing occurrence of item X within a stream and any forecast support value will need to be bounded below by 0.

4.2 Example: Extrapolation-Tree

Figure 2 shows the resulting Extrapolation-Tree tree after an initial block with 20 transactions is added. In this tree, items are inserted in canonical order. The figure shows the tree after insertion of the first 20 transactions. The number in bold represents the item that has $|\Delta supp_{W_1}| > \epsilon$ with a δ value of 0.01.

Transactions for an initial block W_1

tid	transactions	tid	transactions	tid	transactions	tid	transactions
1	b a c e d f	6	b a	11	a c e f h	16	a c f g
2	b a c d	7	b a c e d	12	a c e f h	17	b a c e d f i
3	b a	8	b a c g	13	a c e f h	18	b a e d f g j
4	a	9	b a c e d f i	14	a c e f h	19	b a d g
5	b a c d	10	b a d g	15	b a	20	b a d g



Items	$supp_{W_1}$	$supp_{W_1[1,10]}$	$supp_{W_1[11,20]}$	$\Delta supp_{W_1}$	$supp^{extrap}_{[W_1,W_2]}$
a	1.00	1.00	1.00	0.00	1.00
b	0.70	0.90	0.50	-0.40	0.70
c	0.60	0.60	0.60	0.00	0.60
d	0.50	0.50	0.50	0.00	0.50
e	0.45	0.30	0.60	0.30	0.45
f	0.45	0.20	0.70	**0.50**	0.73
g	0.30	0.20	0.40	0.20	0.30
h	0.20	0.00	0.40	0.40	0.20
i	0.10	0.10	0.10	0.00	0.10
j	0.05	0.00	0.10	0.10	0.05

Estimated order using linear extrapolation
a
f
b
c
d
e
g
h
i
j

Reorganization based on estimated order

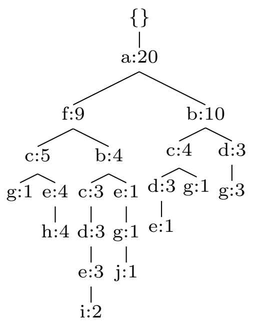


Fig. 2. Example of Extrapolation-Tree

Once the new extrapolated support has been calculated we reorder the tree following the new extrapolated sorted order. All new incoming transactions for W_2 will be inserted according to the extrapolated sorted order.

4.3 Delay Condition

Based on our linear extrapolation function, the transactions in the current block are inserted into the tree based on an estimated order that was the inferred order from the previous block. This estimation function, while reducing the need for reorganization does not eliminate the need altogether as the gap between actual and estimated order will in general widen, given a sufficiently wide time interval, no mater how good the estimation function is. We thus include a mechanism that runs at every block to determine whether reorganization is necessary.

We propose a delay condition, whereby the reorganization phase is delayed until at least the next block if the growth in the tree is less than a specified growth threshold, γ. We measure the growth of the tree by tracking the number of nodes in the tree at the last reorganization point and comparing it with the current number of nodes in the tree. We then calculate the percentage of increased nodes which we call percentage of growth. If the percentage of growth is less than γ, then the reorganization phase is delayed.

4.4 Extrapolation-Tree Algorithm

Algorithm 1 outlines the processes involved in constructing and maintaining Extrapolation Tree.

Algorithm 1. Creation and maintenance of Extrapolation-Tree

```
Input: Transaction database $DB$, block size $w$
Output: $I_{Tree}$, Extrapolation-Tree for the current block
$n \leftarrow 0$ $I_{Tree} \leftarrow \{\}$
cmp ← Appearance_Sorted_Order()
while hasNextTransaction() do
    Insertion Phase:
    trans ← getNextTransaction()
    $I_{Tree}$.Insert_Transaction(trans, cmp)
    $n \leftarrow n + 1$
    Reorganization Phase:
    if $n == w$ and TreeGrowth() $> \gamma$ then
        cmp ← Extrapolation_Support_Order() using Eq 5
        Restructure($I_{Tree}$, cmp)
        $n \leftarrow 0$
    end if
end while
```

In the first block of transactions the function Appearance_Sorted_Order () generates a comparator based on the appearance of the items in transactions as support ordering can only be determined after the arrival of at least one block of data. After the arrival of the first block the support order is computed and a support order for the next block is estimated from the arrival pattern of items in the first block, by calling the Extrapolation_Support_Order() method. The tree is then restructured if the value returned by the function TreeGrowth() is greater than γ.

5 Experimental Results

In our experimentation we tested the Extrapolation-Tree algorithm on both real-world and synthetic datasets. The algorithms were coded in Microsoft Visual

C++, and ran on Windows 7 operating system (Intel core 2 Duo processor with 4GB of main memory). In all experiments, runtime excludes I/O cost.

5.1 Experimentation on Synthetic Data

To evaluate the performance of our algorithm against concept drift we modified the IBM synthetic data generator to inject known drift patterns so that computed reorganization points could be assessed against the occurrence of actual drift points. The data generator produces two types of itemsets: large and small. Those in the large category tend to have higher support values than those in the small category. We divided our itemsets in the large category into two subgroups. We select x random points in time at which we, introduce or remove large itemsets. The main motivation of introducing large itemset is to simulate emerging patterns. Given the original set of large itemsets introduced, we hold off introducing a particular large itemset until we reach a predetermined point. The main motivation of removing large itemset is to simulate disappearing patterns. When we reach a predetermined point, we reduce the occurrence of the large itemset in the stream. All the synthetic datasets were generated using the

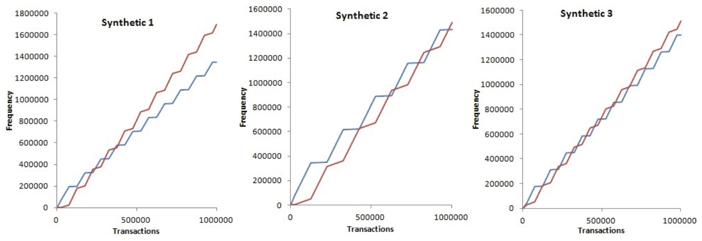


Fig. 3. Drift characteristics of the synthetic datasets

same parameters where the average transaction length was 10, the number of frequent items was 100, the average length of a large item was 4 and the number of transactions was 1 million using 1000 unique items. Figure 3 shows the drift patterns for some of the synthetic datasets that were generated. Each line in the graphs represents a group of large itemsets. The vertical axis represents the cumulative frequency of the particular group.

Evaluation. We measured performance by the effort involved in updating the tree. Table 1 shows the number of nodes updated for each of the datasets. In these experiments we varied the block size (or interval for CP-Tree) from 50K to 200K, while using a γ value of 0.20 and a δ value of 0.01. The fewer the number of updated nodes the more efficient the algorithm, as less processing needs to be carried out.

Table 1. Comparison based on Number of Updates

Dataset	50K		100K		150K		200K	
	Extrap	CP-Tree	Extrap	CP-Tree	Extrap	CP-Tree	Extrap	CP-Tree
Synthetic 1	1145	1249	6096	6283	14618	15081	23315	23402
Synthetic 2	1266	1276	4488	4612	10620	10659	15718	16101
Synthetic 3	995	1008	2382	2611	4173	5190	6983	7452
Synthetic 4	334	369	919	1117	1558	2163	1182	1411
Synthetic 5	754	952	2561	3433	7585	8474	13802	15027
Synthetic 6	298	489	1474	1487	4827	6223	14745	15184
Synthetic 7	98	192	2825	2853	8566	8715	14940	14970
Synthetic 8	155	202	572	720	1623	1646	2231	2559

It is evident from Table 1 that Extrapolation tree outperforms CP-Tree, especially at the smaller block size of 50K. Significant reductions of 20.8%, 39.1%, 49.0% and 23.3% in the number of updated nodes are obtained at this block size for datasets 5, 6, 7 and 8. We next compared the two approaches on tree size using the same range of block sizes as in the above experiment. Our results showed that the two methods produced trees with very similar sizes across the entire range of datasets that we experimented with, thus confirming the accuracy of the forecasting method used.

Table 2. Number of Nodes from Synthetic dataset 7

Block	Extrapolation-Tree		CP-Tree
	After Reorganization	Current Nodes in Tree	After Reorganization
50000	112552		112552
100000	213163		213163
150000	290036		290036
200000	359934		359934
250000	Delayed	424134	424754
300000	487230		487230
350000	Delayed	543749	543286
400000	599576		599576
450000	Delayed	652671	654830
500000	Delayed	705449	706836
550000	756880		756880
600000	Delayed	805459	806855
650000	Delayed	853979	856440
700000	Delayed	901098	903038
750000	947494		947494
800000	Delayed	992137	993236
850000	Delayed	1036327	1037551
900000	Delayed	1079946	1081401
950000	Delayed	1123709	1126087
1000000	1169503		1169503

We then investigated the impact of delay on the growth pattern of Extrapolation Tree. Table 2 contains the results generated from Synthetic dataset 7 with a block size of 50000. For the Extrapolation Tree method we present the results of the nodes after reorganization or the current nodes in the tree if the reorganization is delayed. Table 2 shows that 55% of the block reorganization can be delayed. Furthermore, after reorganization is performed after deferment, it can be seen that the number of nodes in both Extrapolation tree and CP-Tree are still comparable in value, indicating that the extrapolation function is accurately estimating future states of the tree. Thus the Extrapolation Tree method is able to achieve approximately the same degree of compactness without the need to perform expensive reorganizations of the tree after every block, as the CP-Tree method does. The results produced by the rest of the synthetic datasets follow a similar trend.

Table 3. Comparison based on Execution Time

Dataset	50K		100K		150K		200K	
	Extrap	CP-Tree	Extrap	CP-Tree	Extrap	CP-Tree	Extrap	CP-Tree
Synthetic 1	31.4	61.1	25.6	38.5	24.9	27.6	22.9	27.5
Synthetic 2	27.7	54.0	25.3	34.9	23.4	25.4	21.4	24.9
Synthetic 3	28.8	55.8	23.7	35.5	23.2	25.9	21.3	24.5
Synthetic 4	30.5	59.2	24.7	37.1	24.2	40.5	22.3	25.9
Synthetic 5	31.5	57.5	26.0	36.5	25.1	26.3	23.9	25.6
Synthetic 6	31.7	56.4	26.3	35.9	24.6	26.3	23.8	25.6
Synthetic 7	29.8	56.9	24.5	35.9	24.0	26.3	22.2	25.3
Synthetic 8	28.5	54.5	23.4	34.6	22.8	25.2	21.2	24.4

Table 3 shows that on average Extrapolation-Tree was faster than CP-Tree by 25.6% (ranging from a minimum of 4.5% to a maximum of 48.8%). This is mainly due to the possibility of delaying the restructuring phase of the tree, thus reducing the execution time.

5.2 Experimentation on Real World Data

In this section we tested Extrapolation Tree on five real-world datasets from the FIMI and UCI machine learning repositories. We divided each dataset into 10 blocks and used the first transaction as the landmark point. Table 4 shows the

Table 4. Results based on Real-World Dataset

Dataset	Extrapolation-Tree				CP-Tree			
	Nodes	Construction	Mining	Total Time (s)	Nodes	Construction	Mining	Total Time (s)
BMS-POS	1622548	17.0	0.9	17.8	1622820	21.3	0.8	22.1
Accidents	1392600	11.1	0.2	11.2	1394813	16.5	0.2	16.6
Connect 4	366700	7.6	0.0	7.7	359969	8.9	0.4	9.3
Poker-hand-testing	3807746	46.6	0.7	47.3	3809060	66.3	0.1	66.4
Pumsb	1128555	11.6	0.2	11.8	1127574	14.0	0.2	14.2

results in terms of number of nodes, tree construction time, mining time and total time taken for each of the datasets. The results produced by these datasets follow the same trend as the results produced by the synthetic datasets. Both Extrapolation-Tree and CP-Tree produce a similar number of nodes, whereas the total time taken by Extrapolation-Tree remains significantly lower than that of CP-Tree.

6 Conclusions and Future Work

This research has demonstrated that linear extrapolation performs well in predicting future states of a prefix tree as opposed to an approach that simply assumes that the next state of the prefix tree will be the same as the current state. Extrapolation combined with a drift detection and delay mechanism resulted in significant gains in maintenance effort, thus reducing the time required to extract frequent patterns from the tree. Our experimentation also revealed that the gains in runtime performance were not at the expense of tree size as the trees produced by Extrapolation-Tree were similar in size to those of CP-Tree. Our future work will concentrate on examining the performance of Extrapolation-Tree in a sliding window environment as well as testing the effectiveness of non linear extrapolation functions using kernel regression.

References

1. Cheung, W., Zaiane, O.: Incremental mining of frequent patterns without candidate generation or support constraint. In: Proceedings of Seventh International Database Engineering and Applications Symposium, pp. 111–116 (2003)
2. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
3. Koh, J.-L., Shieh, S.-F.: An Efficient Approach for Maintaining Association Rules Based on Adjusting FP-Tree Structures. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 417–424. Springer, Heidelberg (2004)
4. Leung, C.K.S., Khan, Q.I., Li, Z., Hoque, T.: Cantree: a canonical-order tree for incremental frequent-pattern mining. Knowl. Inf. Syst. 11, 287–311 (2007)
5. Li, H.F., Lee, S.Y., Shan, M.K.: Online mining (recently) maximal frequent itemsets over data streams. In: Proceedings of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications, RIDE 2005, pp. 11–18. IEEE Computer Society, Washington, DC (2005)
6. Li, H.F., Shan, M.K., Lee, S.Y.: DSM-FI: an efficient algorithm for mining frequent itemsets in data streams. Knowledge and Information Systems 17, 79–97 (2008)
7. Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB 2002, pp. 346–357. VLDB Endowment (2002)
8. Tanbeer, S.K., Ahmed, C.F., Jeong, B.-S., Lee, Y.-K.: CP-Tree: A Tree Structure for Single-Pass Frequent Pattern Mining. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 1022–1027. Springer, Heidelberg (2008)
9. Yu, J.X., Chong, Z., Lu, H., Zhang, Z., Zhou, A.: A false negative approach to mining frequent itemsets from high speed transactional data streams. Information Sciences 176(14), 1986–2015 (2006)

Dynamic Topography Information Landscapes – An Incremental Approach to Visual Knowledge Discovery

Kamran Ali Ahmad Syed[1], Mark Kröll[2], Vedran Sabol[2], Arno Scharl[3], Stefan Gindl[3], Michael Granitzer[4], and Albert Weichselbraun[5]

[1] Vienna University of Economics and Business, Institute for Information Business, Vienna, AT
`kamran.syed@wu.ac.at`

[2] Know-Center, Division for Knowledge Relationship Discovery, Graz, AT
`{mkroell,vsabol}@know-center.at`

[3] MODUL University Vienna, Department of New Media Technology, Vienna, AT
`{arno.scharl,stefan.gindl}@modul.ac.at`

[4] University of Passau, Media Informatics Department, Passau, DE
`michael.granitzer@uni-passau.de`

[5] University of Applied Sciences Chur, Faculty of Information Science, Chur, CH
`albert.weichselbraun@htwchur.ch`

Abstract. Incrementally computed information landscapes are an effective means to visualize longitudinal changes in large document repositories. Resembling tectonic processes in the natural world, dynamic rendering reflects both long-term trends and short-term fluctuations in such repositories. To visualize the rise and decay of topics, the mapping algorithm elevates and lowers related sets of concentric contour lines. Addressing the growing number of documents to be processed by state-of-the-art knowledge discovery applications, we introduce an incremental, scalable approach for generating such landscapes. The processing pipeline includes a number of sequential tasks, from crawling, filtering and pre-processing Web content to projecting, labeling and rendering the aggregated information. Incremental processing steps are localized in the projection stage consisting of document clustering, cluster force-directed placement and fast document positioning. We evaluate the proposed framework by contrasting layout qualities of incremental versus non-incremental versions. Documents for the experiments stem from the blog sample of the *Media Watch on Climate Change* (www.ecoresearch.net/climate). Experimental results indicate that our incremental computation approach is capable of accurately generating dynamic information landscapes.

Keywords: Information visualization, information landscape, incremental clustering, multi-dimensional scaling.

1 Introduction

These days we are confronted not only with constantly growing, but also with continuously and often rapidly changing "big data" repositories. Information Landscapes represent a powerful visualization technique for conveying topical relatedness in large document repositories [20]. Yet, the concept of information landscapes does only allow for visualizing static conditions. In previous research, we have introduced dynamic topography information landscapes [29] to address both (i) topical relatedness and (ii) visualization of data changes. As such, dynamic landscapes have proved valuable in enterprise scenarios involving visual knowledge discovery in large, dynamic text repositories, where they have been applied for tracking of topical relationships and trends in media and patent databases [30].

Dynamic topography information landscapes are visual representations based on a geographic map metaphor where topical relatedness is conveyed through spatial proximity in the visualization space with hills representing agglomerations (clusters) of topically similar documents. Hills are labeled with dominant terms from the underlying documents to facilitate the users' orientation. When a document repository changes over time, e.g. new documents are added or old documents are removed, the overall topical structure changes as well. Dynamic information landscapes convey these changes as tectonic processes which modify the landscape topography accordingly. In the process of generating information landscapes, high-dimensional data is projected into a lower-dimensional space. Yet, existing dimensionality reduction approaches lack several aspects including (i) support for incremental computation, (ii) scalability with respect to data set size and high-dimensionality (iii) and generation of aesthetically pleasing layouts which are necessary for visual applications.

This paper presents an incremental, scalable algorithmic approach for computing dynamic topography information landscapes capable of visualizing dynamically changing text repositories. Our incremental processing pipeline is introduced in Section 3 and includes implementation details of text preprocessing, projection (dimensionality reduction), labeling and rendering stages where the projection part combines document clustering, cluster force-directed placement and, an improved approach to fast document positioning. We conclude this section by visualizing a temporal sequence of eight incrementally computed information landscapes, which reflect weekly changes in the underlying document set. In Section 4, we experimentally verify our approach's runtime behavior which we discuss only in theory in Section 3. In addition, we evaluate our incremental computation framework by comparing stress values between incrementally and non-incrementally computed layouts. Documents for these experiments are taken from the environmental blog sample of the *Media Watch on Climate Change* [16], a Web content aggregator on climate change and related environmental issues. Our experimental results show that the incremental computation approach yields not only comparable, but even slightly better stress values and thereby indicate our framework's validity.

2 Related Work

Information landscapes are commonly used to visualize topical relatedness in large document repositories, for example in Krishnan et al. [20] and Andrews et al. [1]. Static landscape visualizations, however, cannot convey changes. *ThemeRiver* [13] is a visual representation designed to represent changes in topical clusters, but it cannot express relatedness between documents or topical clusters. Visualization of topical changes through information landscapes with dynamic topographies were proposed in Sabol et al. [28]. An approach suitable for larger data sets was demonstrated in [27]. It relies on 3D acceleration for animated morphing of landscape geometry, which makes it unsuitable for Web applications. However, the performance of the incremental algorithms remains unclear as it was not evaluated or compared with a non-incremental variant.

Visualization techniques in general have to cope with today's ever-growing data production and data consumption. Incremental algorithms provide the required functionality to process big data. Incremental algorithms do not recalculate their internal model from scratch for newly arriving data items and are thus capable of efficiently handling and seamlessly integrating continuously changing or growing data. In the context of generating dynamic information landscapes we review work on incremental dimensionality reduction and incremental clustering techniques.

Incremental Dimensionality Reduction. Dimensionality reduction techniques transform high-dimensional data into low-dimensional data seeking to lose as little information as possible. This transformation has turned out to be particularly useful in the field of visualization for projecting the high-dimension data into the low-dimensional visualization space. To face the growing amount of data, incremental variants have been developed usually on top of batch methods. Incremental unsupervised techniques include multi-dimensional scaling (cf. [4]), singular value decomposition (cf. [31]), principal component analysis (cf. [2]), random indexing (cf. [18]) or locally linear embeddings (cf. [19]). Unsupervised methods are effective in finding compact representations, but ignore valuable class label information of the training data. Incremental supervised techniques are thus better suited for pattern classification tasks. Representatives of incremental supervised dimensionality reduction techniques include linear discriminant analysis (cf. [23]) or subspace learning (cf. [34]).

Incremental Clustering. Incremental clustering algorithms can be traced back to the 1970s, cf. Hartigan's *leader* algorithm which requires only one pass through the data [12], Slagle's shortest spanning path algorithm [33] or Fisher's COBWEB system, an incremental conceptual clustering algorithm [9]. The COBWEB system, for example, has been successfully applied to support fault diagnosis or bridge design. Inspired by COBWEB, Gennari et al. proposed the CLASSIT [11] system which is capable of handling numerical data sets. In [5], the authors introduced an incremental clustering algorithm for dynamic information processing. In dynamic databases there is a constant adding or removing of data items over time. The idea is that these changes should be recognized in the generated partition without affecting current clusters. In the late nineties, several incremental clustering algorithms have been presented including BIRCH [35], incremental DBSCAN [8] to support data warehousing or Ribert et al.'s clustering algorithm to generate a hierarchy of clusters [26]. Incremental clustering of text documents has been conducted as a part of the Topic Detection and Tracking initiative [1] to detect a new event from a stream of news articles.

To compute dynamic topography information landscapes in an incremental and thus timely efficient manner, we integrate and combine incremental aspects into the generation process. (i) For clustering, we apply a simple, spherical k-means [7] and use previously computed partitions of the document set as initial state for incremental computations. (ii) We introduce an improved approach for document positioning which is essentially based on a simple spring forces-based model (cf. [10]) since we observed that landscapes generated with standard positioning method displayed geometrical edges. (iii) We use a force-directed placement (FDP) algorithm [10] for projecting these high-dimensional cluster centroids into a 2D visualization space. The parameters of FDP-based methods provide significant control over the layout, which allows them to deliver more pleasing layouts than traditional methods. The FDP algorithm is intrinsically incremental when applied on a previously computed stable layout. Re-applying FDP on a previous layout of centroids with modified similarities will produce a new layout closely resembling the previous one.

3 Algorithmic Approach

In this section we introduce and describe our approach to generating dynamic information landscapes. Fig. 1 depicts the overall workflow, which can be grouped into three main components: (i) First (shown in green), we prepare an augmented document-term matrix by combining information from keyword relevance and word frequency tables. (ii) In a second step (cyan), we cluster and position the documents. We use the k-means clustering algorithm to partition the documents into topically related clusters. We then employ force directed placement to project clusters centroid positions into 2D visualization space, and apply a fast method for positioning documents in 2D based on cluster positions. (iii) In the last step (magenta), we use the documents' layout position to model a topical landscape which is essentially an elevation matrix on a 2D grid. A coloring scheme is used to construct landscape surface images. A peak detection algorithm then finds major peaks (hills) and collects underlying documents to compute text descriptors for labeling the peaks. Note that previous computation results are used as initial state for incremental processing (in orange).

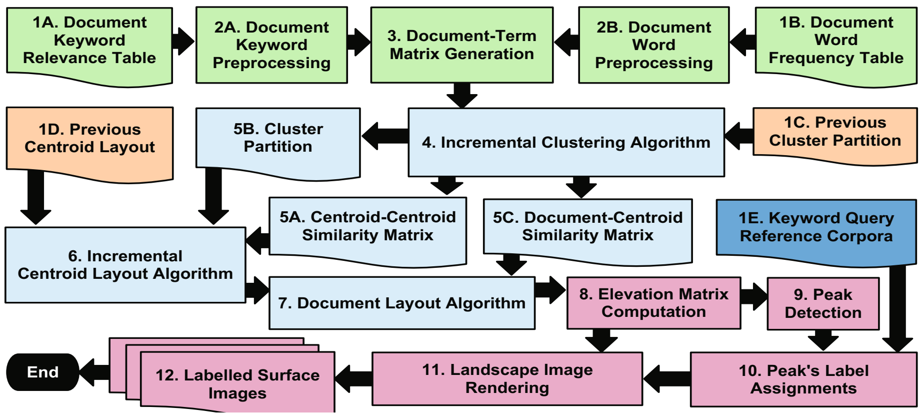

1A. Document Keyword Relevance Table
2A. Document Keyword Preprocessing
3. Document-Term Matrix Generation
2B. Document Word Preprocessing
1B. Document Word Frequency Table
1D. Previous Centroid Layout
5B. Cluster Partition
4. Incremental Clustering Algorithm
1C. Previous Cluster Partition
5A. Centroid-Centroid Similarity Matrix
5C. Document-Centroid Similarity Matrix
1E. Keyword Query Reference Corpora
6. Incremental Centroid Layout Algorithm
7. Document Layout Algorithm
8. Elevation Matrix Computation
9. Peak Detection
End
12. Labelled Surface Images
11. Landscape Image Rendering
10. Peak's Label Assignments

Fig. 1. Workflow diagram for the incremental landscape computation framework

Details on these three main components will be provided in the following subsections 3.1 to 3.3, followed by a separate description of the architecture's incremental aspects in Section 3.4.

3.1 Document-Term Matrix Generation

Prior to the beginning of the computation, the raw textual data to be analyzed is gathered via a web crawler, then converted and annotated into the content repositories based on previous research [16], [32]. We utilize our experiences with *webLyzard*, an established and scalable media monitoring and Web intelligence platform (www.weblyzard.com), to generate the document keyword relevance table (1A) and the document word frequency table (1B).

Using the information from 1A and 1B, we create the document keyword matrix (2A) as well as the document word matrix (2B). Both matrices are then linearly combined into one augmented document term matrix (3) with unique terms IDs.

3.2 Clustering and Projection

The incremental clustering algorithm (4) takes the document term matrix (3) as input and outputs (i) a centroid-to-centroid similarity matrix (5A), (ii) a document-centroid relationship graph (5B), and (iii) a documents-to-centroids similarity matrix (5C).

In incremental mode the k-means algorithm module is initialized by the previously computed clustering result (1C). The centroid positioning algorithm (6) uses results (5A) and (5B). The algorithm can be initialized with previous centroid positions (1D) for the incremental case, or by assigning random positions for the non-incremental computation. The centroid positions (6) and the document to centroid similarity matrix (5C) are then used for computing the document positions (7).

Document Clustering. We apply the spherical k-means algorithm [7] to partition the documents into topical clusters. The k-means algorithm is known to be highly sensitive to the initial guess of the cluster partitions and the number of partitions. To overcome this sensitivity, we use the k-means++ seeding method [3]. In addition, we split and merge the clusters [22] for deducting the number of clusters within the limit of specified minimum and maximum bounds. As human cognition puts certain limits to conceiving visualizations, we limited the number of clusters to account for usability. We observe that setting the minimum and maximum number of cluster bounds to be 30 and 40, respectively, result in meaningful and aesthetically pleasing information landscapes. Therefore, in subsequent iterations we perform the splitting of large clusters to obtain higher cluster cohesion as well as the merging of small, similar clusters, according to improvements using Bayesian Information Criterion [24].

The algorithm's runtime complexity is $O(mnd)$, where m is the number of clusters, n is the number of documents, and d is the dimensionality of the term space. Since $m \ll n$ in our case, and according to Heaps' law [14], d scales logarithmically with n, the clustering part of our algorithm is considered to scale with $O(n \log(n))$. For incremental clustering, we use previously computed partitions of the document set as the initial state. For a fixed number of documents to be clustered incrementally, old documents are removed from their respective clusters and new documents are added to the most similar cluster centroid. Afterwards, additional k-means iterations, including the split and merge procedure, are performed to further refine the initial partition.

Cluster Positioning. The partitioned set of documents is represented by the high-dimensional centroids of the respective clusters. We use a force-directed placement (FDP) algorithm [10] for projecting these high-dimensional cluster centroids into a 2D visualization space. The idea is that attractive forces pull together topically similar centroids while dissimilar centroids are repulsed. Spatial closeness between centroids thus relates to their topical closeness. The FDP algorithm is known to produce accurate and aesthetically pleasing layouts. Most FDP variants scale poorly, e.g. $O(m^3)$. Yet, as in our approach $m<<n$, and because there is a fixed upper limit on m, in our case 40 clusters, the runtime complexity of cluster positioning may be considered constant. As stopping criterion for the FDP algorithm, we used two parameters: (i) a fixed maximum number of iterations, and (ii) the local minima for the stress value [21]. The most attractive feature of the FDP algorithm is that it is intrinsically incremental when applied on a previously computed stable layout. The impact of incremental clustering is reflected in similarities between cluster centroids. When changes in the data set are small, the similarities between the centroids will also change by a small proportion. Re-applying FDP on previous layout of centroids with modified similarities will produce a new layout closely resembling the previous one.

Document Positioning. In an earlier version of the algorithm [29], we used an algorithm based on Delaunay triangulation of centroid positions in the 2D space. The most similar triangle was chosen based on the similarities between the document and the most similar centroids, and the document position was assigned using Barycentric coordinates in $O(m)$ time, m being the number of centroid vertices.

Unfortunately, we observed that landscapes generated with this positioning method reflected geometrical edges; i.e., documents were positioned in straight lines. To maintain the viewing experience of a realistic landscape without artifacts, and to achieve a linear running time, we introduce an improved approach for fast document positioning which is essentially based on a simple spring forces-based model (cf. [10]). In this model we assumed that the document, in two dimensions, is attached to each centroid, in two dimensions, by a spring having a spring constant proportional to the similarity between the document and the centroid in the n-dimensional space.

If $R_1, R_2, R_3, \ldots, R_m$ are the given position vectors of all m centroids and $s_{i1}, s_{i2}, s_{i3}, \ldots, s_{im}$ are the given similarities of the i^{th} document with m centroids respectively, then an analytical solution of the equilibrium conditions for Hooke's law forces between i^{th} document and all centroids eventually formulate the position of the i^{th} document as $r_i = \sum_{k=1..m} s_{ik} R_k / \sum_{k=1..m} s_{ik}$. This simple algorithm makes our computation for document positioning linear in time and, in contrast to [29], without any overhead

3.3 Landscape Creation and Peak Labeling

With document layout positions at hand, we compute an elevation matrix (8) that represents an information landscape model. We then utilize this matrix to identify peak locations, heights and a list of documents related to the peak (9). The peak detection employs a kernel window convolution over the landscape model. The peak label assignment module (10) determines the peak's labels by using the list of documents under the peak for querying and comparing with the semantically tagged reference corpora (1E), which is continuously refined by the *webLyzard* platform. Finally, the assigned labels are positioned on the information landscape surface images (12), computed based on the coloring scheme (8) and the heuristic labeling algorithms of the landscape image rendering module (11).

Landscape Modeling. Information landscapes with specific resolutions are modeled as elevation matrices of the same resolution. A document is thought of as a small Gaussian peak at the corresponding position on the underlying matrix cells. The influence of a document on a matrix cell location is reflected by the value of Gaussian density at that location. Thus the height and the asymptotic radius of the Gaussian peak reflect the document's influence in the landscape. We further assume a document has a fixed influence on its own location on the matrix cell. The densities of all documents at particular location are superimposed, adding to the elevation values of the underlying matrix cells.

Peak Detection. A kernel window-based peak detection algorithm is used to detect the significant peaks of the landscape (cf. [15], [25]). The average of the convolution of the window with the elevation matrix is compared with the center value of the matrix cell. A peak is assumed if the center value is higher than the average convolving value. After detecting the significant peaks, documents are assigned to their nearest peak by using the minimum Euclidean distance criterion in the 2D layout.

Label Computation. The term distribution in the set of documents in the vicinity of a peak is compared with a reference distribution. A chi-square test of significance with Yates' correction determines over-represented terms. The *term* co-occurrence analysis, based on pattern matching algorithm, along with *trigger phrases* based on regular expressions, is used to identify the frequently appearing text fragments within the same sentences and within the documents [16], [32]. The redundancy of nouns' singular and plural forms and synonyms in the resultant list of labels are removed by using a combination of regular expression queries and WordNet library lookup.

Map Generation and Label Placement. In the final step, colors are assigned to the image pixels depending on the density of the corresponding density matrix cells. In our scheme of colors the blue is used to express lowest density, then green and brown, and finally light gray is used for highest density. The resulting landscape surface image resembles a geographic map with peaks at areas, where document density is large, and oceans or valleys, where document density is low. Finally, a heuristic point feature label placement algorithm [6] is used with the labeling quality evaluation in the following basic rules: (i) No overlap of a label with other labels and the image boundary. (ii) No overlap of a label and another peak location. (iii) Each label is placed among the four possible labeling rectangular spaces of the peak locations. (iv) At most five labels for a peak location can be assigned.

3.4 Incremental Computation

Computing incremental landscapes at first requires an initial computation of a landscape. We apply our algorithm to an initial data set where the documents' layout positions are saved for future use. Whenever the data set changes, the incremental k-means algorithm is initialized with a previously computed stable partition, i.e. with the old locations for the new centroids. The ongoing process of removing old documents and adding new documents to the most similar clusters leads to several k-means iterations for the next stable partition. The successive iteration of FDP will stop at the first local minima for the average stress value.

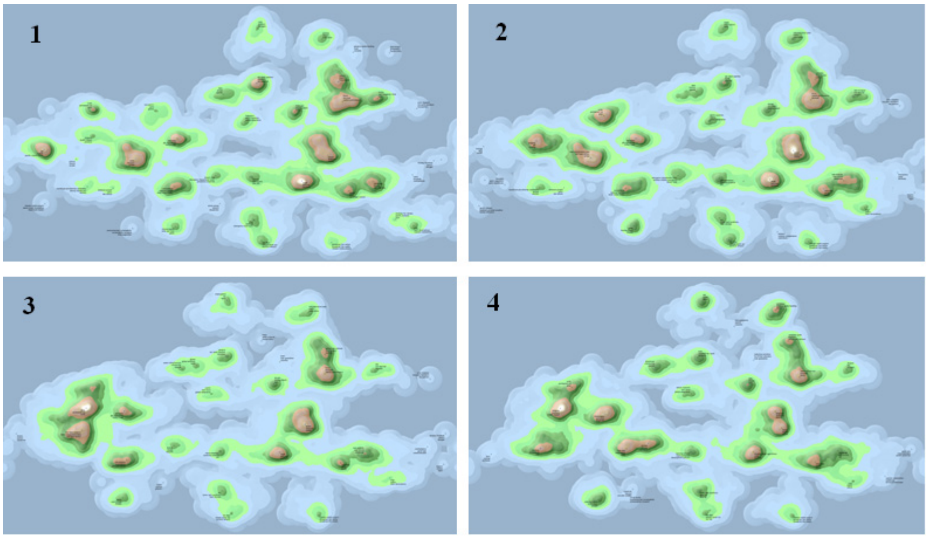


Fig. 2. A sequence of incrementally computed landscapes from environmental blogs, visualizing 2,000 documents each, reflects weekly changes from Sept 30th, 2011 to Oct 21th 2011. Approximately 10% of the data set changed between each individual step resulting in seamless transformations of topography portions, while the overall structure remains stable. Rising hills indicate the emergence of new topics (images 1, 2 and 3); shrinking hills a fading of topics (image 4). Hill movements towards or apart from each other indicates converging or diverging topical clusters. As the incremental algorithm seamlessly integrates a stream of continuous changes, the user retains orientation through recognition of unchanged parts of the topography.

To acknowledge the growing number of documents to be processed by state-of-the-art web intelligence applications, we briefly discuss scalability issues in this section. Many processing steps of our algorithmic approach scale linearly (or even better) with the number of documents *n*. Yet, the dominating factor remains with the clustering, so the time complexity of the entire landscape generation process is ***O(nlog(n))***. This matches the performance of other scalable algorithms, such as [17], which however do not provide support for incremental layout computation. While we have experienced with data set sizes up to 20.000 documents, we still need to conduct large-scale experiments to make reliable statements with respect to scalability.

4 Evaluation

For evaluating the incremental computation framework we computed ten consecutive landscapes for 2000 documents from the environmental blog data set of the *Media Watch on Climate Change* [16], a Web content aggregator about climate change and related environmental issues that serves static versions of the information landscapes presented in this paper as part of a multiple coordinated view representation. Each week new documents are gathered via the *webLyzard* crawler. For each incremental step, new documents replace an equal number of old documents from the set of 2000 documents. Each new incremental computation is based on the previous one. For comparison, we compute the landscape for the same document set in a non-incremental manner. The projection quality in both landscapes is then evaluated using stress values [21].

However, the stress value computation requires the computation of distances (dissimilarities) for all pairs of documents in high-dimensional space, which is quadratic in time. To speed up the process and to be capable of handling large data sets, we introduce a faster variant which approximates the true stress values. We used geometric mean of the similarity of one document with the centroid of the second document and the similarity of second document with the centroid of the first document as an estimated value of similarity between both documents. All measurements were performed on a 2.66GHz Intel Xeon X5355 CPU with 8GB of memory, running 64-bit versions of Linux and Java v1.6.0_29.

The resulting stress values for both computation types are summarized in Fig. 3 (left). The initial sample of 2000 documents was taken from September 30th, 2011. Every week this document selection changes, i.e. new documents arrive whereas the same number of documents, the oldest ones, are removed resulting in a set of constant size. Stress values for both computation types are decreasing while values for the incremental computation appear to be slightly lower than for the non-incremental computation. In the non-incremental case, the curve exhibits more fluctuations, e.g. the peak on November 4th. In our opinion this behavior is due to k-means' and FDP's sensitivity to initial conditions. We hypothesize that stress values for the incremental computations are lower because these weekly incremental changes have the potential to shake the FDP process out of local minima so that the performance can improve. The experimental results corroborate that our algorithmic approach is capable of accurately generating dynamic information landscapes in an incremental manner.

To examine the algorithm's execution times for different data set sizes, we experimentally verified the runtime estimates for individual processing steps given in Sections 3.1 to 3.3. Fig. 3 (right) summarizes timing results of landscape computations for eight different document set sizes ranging from 2000 to 16000. Processing steps include document-term matrix preparation (A), clustering (B), document positioning (C) (including cluster positioning with FDP which is in constant time for fixed number of clusters) and peak detection, label positioning and image construction (D). Graph (T) reflects the total runtime for generating dynamic topography information landscapes for different data set sizes. According to Fig. 3 (right), the clustering step (B) appears to be the algorithm's runtime bottleneck.

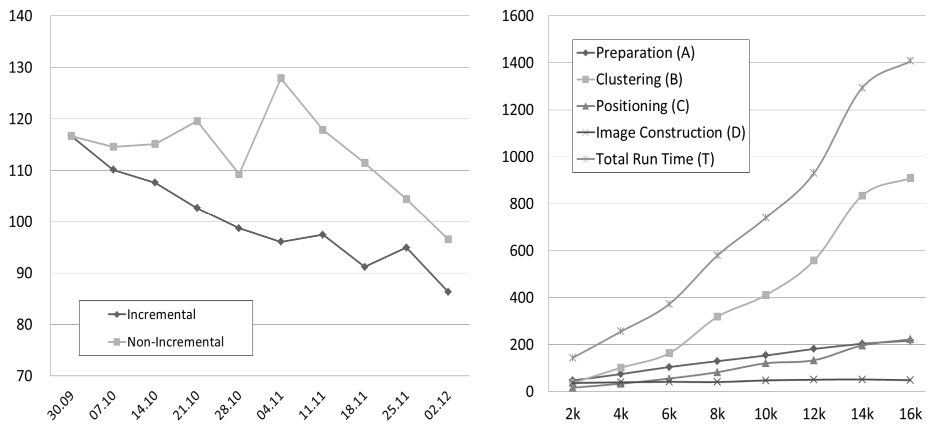


Fig. 3. Left: The stress values (y-axis) for incrementally computed documents layout and for non-incrementally computed documents layout over a period of 10 weeks; Right: Run times in seconds (y-axis) for landscape computation framework with different document sets (x-axis)

5 Conclusion

We have introduced and evaluated an incremental approach to generating dynamic topography information landscapes, and applied this approach to visualize the content dynamics of environmental blogs. Our method combines well-known algorithmic approaches, such as k-means clustering and force-directed placement, and introduces an improved method for fast document positioning which relies on previously computed cluster centroid positions. In experiments, we have compared the quality of incrementally and non-incrementally computed layouts where the incremental version achieves not only comparable, but even slightly superior stress values.

By capturing changes in textual data repositories such as news and social media archives, and by revealing the emergence and decay of major topics in such repositories, an incremental version for computing information landscapes extends the repertoire of existing Web intelligence and social media analytics applications such as the *Media Watch on Climate Change* (www.ecoresearch.net/climate).

Although some of incremental ideas are discussed in [27, 28, 29, 30], this paper contributes by presenting a novel document positioning method and evaluates document positioning improvements on subsequent incremental landscape computations.

Future work will focus on improving layout quality by utilizing semantic information in the process of calculating similarities between documents. These semantics will help us to better handle linguistic concepts such as synonymy and thus to capture more implicit, meaningful associations amongst textual resources.

Acknowledgement. The work presented in this paper was developed within the DIVINE project (www.weblyzard.com/divine), funded by the Austrian Ministry of Transport, Innovation & Technology (BMVIT) and the Austrian Research Promotion

Agency (FFG) within the strategic objective FIT-IT (www.ffg.at/fit-it). The Know-Center is funded within the Austrian COMET Program (Competence Centers for Excellent Technologies) under the auspices of BMVIT, the Austrian Ministry of Economics and Labor, and by the State of Styria.

References

1. Allan, J., Carbonell, J., Doddington, G., Yamron, J., Yang, Y.: Topic Detection and Tracking. Pilot Study Final Report (1998)
2. Artac, M., Jogan, M., Leonardis, A.: Incremental PCA for on-line visual learning and recognition. In: Proceedings of the 16th International Conference on Pattern Recognition, pp. 781–784 (2002)
3. Arthur, D., Vassilvitskii, S.: K-means++: The advantages of careful seeding. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035 (2007)
4. Basalaj, W.: Incremental multidimensional scaling method for database visualization. In: Proceedings of SPIE - The International Society for Optical Engineering, pp. 149–158 (1999)
5. Brand, M.: Incremental Singular Value Decomposition of Uncertain Data with Missing Values. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part I. LNCS, vol. 2350, pp. 707–720. Springer, Heidelberg (2002)
6. Christensen, J., Marks, J., Shieber, S.: An empirical study of algorithms for point feature label placement. ACM Trans. on Graphics 14(3), 203–232 (1995)
7. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Machine Learning 42(1/2), 143–175 (2001)
8. Ester, M., Kriegel, H.-P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: Proceedings of 24th International Conference on Very Large Data Bases (VLDB 1998), pp. 323–333 (1998)
9. Fisher, D.: Knowledge acquisition via incremental conceptual clustering. Machine Learning 2, 139–172 (1987)
10. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. Software - Practice and Experience 21, 1129–1164 (1991)
11. Gennari, J., Langley, P., Fisher, D.: Models of incremental concept formation. Artificial Intelligence 40, 11–61 (1989)
12. Hartigan, J.A.: Clustering Algorithms. John Wiley and Sons, Inc., New York (1975)
13. Havre, S., Hetzler, E., Whitney, P., Nowell, L.: ThemeRiver: Visualizing thematic changes in large document collections. IEEE Transactions on Visualization & Computer Graphics 8(1), 9–20 (2002)
14. Heaps, H.H.: Information Retrieval: Computational and Theoretical Aspects, pp. 206–208. Academic Press (1978)
15. van Herk, M.: A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. Pattern Recognition Letters 13(7), 517–521 (1992)
16. Hubmann-Haidvogel, A., Scharl, A., Weichselbraun, A.: Multiple coordinated views for searching and navigating web content repositories. Information Sciences 179(12), 1813–1821 (2009)
17. Jourdan, F., Melancon, G.: Multiscale hybrid MDS. In: Proceedings of the Eighth International Conference on Information Visualisation (IV 2004), pp. 388–393 (2004)

18. Kanerva, P., Kristofersson, J., Holst, A.: Random indexing of text samples for latent semantic analysis. In: Proceedings of the 22nd Conference of the Cognitive Science Society, pp. 103–106 (2000)
19. Kouropteva, O., Okun, O., Pietikäinen, M.: Incremental locally linear embedding. Pattern Recognition, 1764–1767 (2005)
20. Krishnan, M., Bohn, S., Cowley, W., Crow, V., Nieplocha, J.: Scalable visual analytics of massive textual datasets. In: 21st IEEE Int'l Parallel and Distributed Processing Symposium 2007. IEEE Computer Society (2007)
21. Kruskal, J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29(1), 1–27 (1964)
22. Muhr, M., Granitzer, M.: Automatic cluster number selection using a split and merge k-means approach. In: Proceedings of the 20th International Workshop on Database and Expert Systems Application, pp. 363–367 (2009)
23. Pang, S., Ozawa, S., Kasabov, N.: Incremental linear discriminant analysis for classification of data streams. IEEE Transactions on Systems Man and Cybernetics 35, 905–914 (2005)
24. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proceedings of the 17th International Conference on Machine Learning, pp. 727–734 (2000)
25. Razaz, M., Hagyard, D.M.P.: Efficient convolution based algorithms for erosion and dilation. In: Proc. of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP 1999), pp. 360–363 (1999)
26. Ribert, A., Ennaji, A., Lecourtier, Y.: An incremental hierarchical clustering. In: Proceedings of the Vision Interface Conference, pp. 586–591 (1999)
27. Sabol, V., Kienreich, W.: Visualizing Temporal Changes in Information Landscapes. Poster and Demo at the EuroVis (2009)
28. Sabol, V., Scharl, A.: Visualizing Temporal-Semantic Relations in Dynamic Information Landscapes. In: 11th International Conference on Geographic Information Science, Semantic Web Meets Geospatial Applications Workshop. AGILE, Girona (2008)
29. Sabol, V., Syed, K.A.A., Scharl, A., Muhr, M., Hubmann-Haidvogel, A.: Incremental Computation of Information Landscapes for Dynamic Web Interfaces. In: Proc. of the 10th Brazilian Symposium on Human Factors in Computer Systems, pp. 205–208 (2010)
30. Sabol, V., Kienreich, W., Muhr, M., Klieber, W., Granitzer, M.: Visual Knowledge Discovery in Dynamic Enterprise Text Repositories. In: Proceedings of the 13th International Conference on Information Visualisation (IV 2009). IEEE Computer Society (2009)
31. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Proceedings of the 5th International Conference on Computer and Information Science, pp. 399–404 (2002)
32. Scharl, A., Weichselbraun, A., Liu, W.: Tracking and modelling information diffusion across interactive online media. International Journal of Metadata, Semantics and Ontologies 2(2), 135–145 (2007)
33. Slagle, J.R., Chang, C.L., Heller, S.R.: A clustering and data-reorganizing algorithm. IEEE Trans. Syst. Man Cybern. 5, 125–128 (1975)
34. Yan, J., Cheng, Q., Yang, Q., Zhang, B.: An Incremental Subspace Learning Algorithm to Categorize Large Scale Text Data. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 52–63. Springer, Heidelberg (2005)
35. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 103–114. ACM, New York (1996)

Classifying Websites into Non-topical Categories

Chaman Thapa, Osmar Zaiane, Davood Rafiei, and Arya M. Sharma

University of Alberta
{chaman,zaiane,drafiei,amsharm}@ualberta.ca

Abstract. With the large presence of organizations from different sectors of economy on the web, the problem of detecting to which sector a given website belongs to is both important and challenging. In this paper, we study the problem of classifying websites into four non-topical categories: *public*, *private*, *non-profit* and *commercial franchise*. Our work treats each website and all pages from the site as a single entity and classifies the entire website as opposed to a single page or a set of pages. We analyze both the textual features including terms, part-of-speech bigrams and named entities and structural features including the link structure of the site and URL patterns. Our experiments on a large set of websites related to weight loss and obesity control, under a multi-label classification setting using the SVM classifier, reveal that with a careful selection and treatment of features based on keywords, one can achieve an F-measure of 70% and that adding structural, part-of-speech and named entity based features further improves the F-measure to 74%. The improvement is more significant when textual features are not accurate or sufficient.

Keywords: non-topical classification, structural features, topical features, non-topical features, web genre.

1 Introduction

The tremendous growth of World Wide Web over the past few years has made it extremely easy for end-users to reach the general mass public by having a web presence. As more people, organizations and governments publish their information on the web, it is important and increasingly difficult to find and filter the desirable information from the web. For example, one may want to know from the website of a health clinic if it is publicly funded hence the treatment expenses are paid by the public health insurance. In such a scenario, associating websites with desirable labels can be helpful in improving the search by linking labels with the search query and allowing the users to filter the websites more easily. Automatic classification of websites can also be helpful in automating the process of creating web directories which takes considerable effort if humans were to label the websites manually.

Website classification can be treated under text classification assuming that a website is a set of web pages or documents. A problem with applying a textual classifier to non-topical classes is that these classes may not be well-described in text and a richer set of features need to be maintained. The problem is similar to classifying documents based upon the sentiment [3, 4], identifying text genre [2], etc. For example, features such as part-of-speech tags, punctuations, and named entities in addition to words from text have turned out to be useful, as reported in our experiments and also in some past work [5, 9]. Beyond text and part-of-speech patterns, the link structure of the site, URL patterns [10–12] and HTML tags [8] may also provide additional useful information that can help to correctly classify the website.

In this research, we will be classifying websites into 4 non-topical categories: *public, private, non-profit*, and *commercial franchise*. The non-topical categories that we are concerned with are related to websites that fall under the domain of weight loss/obesity control in Canada. Since many service providers for obesity control have a web presence, classifying the entire website would reveal important facts about these organizations, These facts may inform the users, for example, about the cost and the reliability of a service provided by these organizations. Our domain experts have confirmed that these facts will be useful for obesity patients to efficiently filter the required resources from the web. With the categories that we are concerned about, a website can have more than one label, hence we explore how the independent feature sets based on the link structure of the site, part-of-speech and named entity distribution, and bag-of-words perform in a multi-label classification setting.

In order to classify the entire website, we consider a collection of web pages from a website as a single document. A website can contain hundreds of pages and the inclusion of every page from the website increases the number of features based on bag-of-words. It also takes a significant amount of time to extract the part-of-speech and named entities. More attributes would also mean that the classification takes more time and there is a high probability for the occurrence of noise. In order to mitigate this problem, we analyze dimensionality reduction by selecting the features based on information gain and click-depth of a page, the number of clicks required by the user to reach a page. As the click-depth of the website increases, there is an increase in the number of pages. We analyze how the classifier performs when the word-based features are extracted at click-depth of zero, one and two.

Our contributions include: (1) a study in detecting the business type of an entity from its website, (2) a non-topical website classifier with classes that relate to the business type of the entity a website presents, (3) applying the classifier to the real world domain of weight loss and obesity control in Canada, and (4) an experimental evaluation showing the performance of the classifier and the effectiveness of the features studied.

The rest of the paper is organized as follows. Section 2 reviews the related work and Section 3 presents the dataset and the way it is acquired and the labels are assigned. Section 4 presents the types of features and their selection process and Section 5 reports our results and analysis of the classification. Finally, we draw the conclusions in Section 6.

2 Related Work

Related to our investigation is the body of work on non-topical classification on the Web. Mishne [1] illustrated a supervised classification of blog posts based on the mood of the writer. Turney [3] presented an unsupervised classification of reviews and Pang et al. [4] applied supervised algorithms such as SVM, Naïve Bayes and the maximum entropy to movie reviews. These work on non-topical classification focus on finding the right features that work best for the dataset. Some of the features that have been used for sentiment analysis are unigrams, unigrams combined with bigrams and part-of-speech tags. Bekkerman [5] showed that combining POS-bigrams along with bag-of-words improved the classification accuracy in case of the genre classification.

Dai et al. [6] classified web pages into commercial and non-commercial classes in an attempt to detect the online commercial intent of a page. This work is close to ours as some of the categories overlap, however the authors only used keywords from text and html attributes as features whereas we are combining non-topical features with word-based features. Ester et al. [7] performed a topical classification of websites using a k-order markov model and also treating pages from the same site as a single page. Pierre [8] showed that words from metatags are useful in the classification of websites into industrial categories. His result showed that words from metatags alone can be more effective than words from metatags and html body combined together; however, it also showed that metatags is not widely used by many websites. In our research, the bag-of-words feature comprises of words from metatags, title and the html body. A more recent work by Eickhoff et al. [9] classified web pages based on whether it is targeted towards children or not. They combined both topical and non-topical aspects of a document by using features such as part of speech, shallow texts, html features, and language complexity. They showed that combining topical and non-topical features can work well for non-topical classification.

There is also work on analyzing the structural properties of websites. Amitay et al. [10] used the structure of websites to classify them into eight functional categories and showed that sites with similar functions shared similar link structures. Lindemann and Littig [11] did a thorough study on the relationship between the structure and functionality of the websites. Their work analyzed 1461 websites distributed among five functional categories and reported a strong accuracy using the structural properties. Later, Lindemann and Littig [12] also showed that utilizing both the content and structural properties for website classification performed better than using structural or word based features alone.

From the past work on non-topical classification of documents and websites, it is evident that words are a powerful set of features even for non-topical classification. Results have also shown that combining part-of-speech along with words improves the performance for non-topical classification of documents. Non-topical website classification also benefits from a combined feature set where words are augmented by structural properties. Our work combines and analyzes all three aspects (i.e. the structure of text in the form of part-of-speech tags and named entities, structural pattern of the website, and bag-of-words) in a real world non-topical website classification task.

3 Dataset Preparation

We used a set of keywords related to weight loss in a search engine to come up with the list of websites. As we were only concerned with organizations providing services related to obesity control/weight loss and having a physical presence in Canada, we built a collection of search queries by appending different city names like Edmonton, Toronto, etc to useful keywords, which were suggested by obesity experts. Some of the keywords used were "obesity clinic", "weight management", "fitness and exercise","diet program" etc. Using these search queries with Google, we came up with a list of websites. We then did an extensive online survey where 77 users participated in labeling the websites. The definition of the categories that were used to label the websites are as follows:

Public: A website providing service that is offered or subsidized by the government.

Private: The service provider has a private firm and is a licensed health care professional or has certification.

Non-profit: A service that has been provided on a non-profit basis.

Commercial Franchise: An organization that provides or sells services or products for profit. In many cases, the organization has many branches (> 2) in different parts of the country and is considered a chain.

We picked only those labels where two or more users agreed upon and where the websites provided service related to obesity control. This was checked through the online survey where the users had the option to tag websites that were not related to obesity control. This helped us filter out many blog sites and web directories. However, obtaining the labels this way did not give us enough labels to populate each category as most of the websites in the search results were either private or franchise. Hence, we also asked one patient and one student to extensively search the web for non-profit and public categories. All the website labels were later verified by an expert and the expert's decision on the label was considered final. The final list of labels we collected comprised of 215 websites where the majority of the labels were private (116). This set was highly imbalanced with more than 50% of the labels comprising of private websites. Since we are measuring the strength of each type of feature, we did not want any bias due to over-fitting in micro and macro measures during evaluation; hence we balanced the dataset by randomly selecting the websites from the over populated category. Table 1 shows the label distribution among various combinations of multi-label categories after the dataset was balanced.

Since the public and non-profit categories had a small number of samples, we kept all the websites in these categories. On the other hand, we under-sampled

Table 1. Website count for various label combination

Label Combination	**Number of Websites**
Public	25
Non-profit	24
Franchise	21
Private	13
Public,Private	10
Public,Private,Non-profit	2
Private,Franchise	24
Public,Non-profit	6
Total	125

the franchise and private categories such that the final label distribution for each category is: public (43), private (49), franchise (45) and non-profit (32).

In order to capture the features based on the language model, we crawled the websites at various click-depths. The landing page or the homepage of a website is considered at a click-depth of zero. All the links present in the homepage are then considered at click-depth of one and so on. We crawled the internal links of each website at click-depth of zero, one and two. We saved only those files for which the server response was valid and the header had text/html as the content-type. The maximum number of files we crawled for each website was limited to 1000 pages. Table 2 shows the number of pages crawled at each click-depth. Click depth of 2 contains all the pages at depth zero, one and two inclusively. We will use this convention throughout the paper.

Table 2. Number of pages crawled at each click-depth

Depth	**Number of Pages**	**Avg. Page Size (In KB)**
Click-depth 0	125	24.9
Click-depth 1	5322	99.05
Click-depth 2	34981	127.91

The structural properties of the websites were captured by crawling the internal links of each website with valid HTML server response up to a depth of ten. The HTML pages themselves were not saved but the URLs pertaining to external and internal links at each depth were recorded to extract the structural properties. We only expanded the internal links and marked the external links as a new external link or previously appeared external link. We crawled an internal link only once. If an internal link appears more than once, we mark them as already visited.

4 Features Used for Classification

4.1 Features Based on Words

Words provide useful cues as to which category a particular website belongs. Analyzing the websites with human eye, we can see that websites in franchise categories mostly contain words such as *order*, *pay*, *success*, and *testimonials.* Private websites mostly contain terms like *doctor*, *clinic*, *physician* and other common medical terms. Public websites mostly contain the keyword *government* and non-profit ones often have words like *donate*, *voluntary.* Some of the non-profit organization list themselves as being a not-for-profit organization in their about page. There are many variations of the keyword non-profit expressed as "not for profit", "non profit" or even "not-for-profit". We combined these variations as a single entity: *nonprofit*, using a regular expression.

We followed the bag-of-words approach and extracted word unigrams from HTML documents. We used a HTML parser[1] to extract the text from the body and title of the document. Words from meta-keywords and meta-description tags were then added to the list of unigrams. We then extracted the word-stem for each unigram and represented the word stem in a feature vector using the TF-IDF metric. Since we were dealing with a collection of web pages within a website, we followed a slightly different definition of TF-IDF to normalize the term frequency within a website.

$$\mathrm{tfidf(t,W)} = \frac{\mathrm{tf}}{\mathrm{P}} \times \log\left(\frac{\mathrm{N}}{\mathrm{DF}}\right) \qquad (1)$$

In equation 1, *tfidf(t,W)* gives the TF-IDF measure of a term t for the website W. *tf* is the term frequency of the word t i.e. the number of times t occurs in W. P is total number of web pages in W where term t occurs. DF is the document frequency of t with respect to the websites i.e. the number of websites in the dataset in which the term t occurred. N represents the total number of websites in the dataset. We discarded any term whose document frequency (DF) is less than three.

4.2 Part-of-Speech and Named Entity Based Features

Part-of-speech provides useful information about the structure of a sentence which can be helpful in capturing the notion of the categories. It has been shown to be useful for text classification where sentences are well formed. However, capturing POS tags from HTML documents can be a bit tricky as HTML documents can mostly contain words as opposed to sentences. In order to extract POS tags from HTML documents, we processed the document to extract groups of text which contain a sentence boundary, (i.e. the symbols "." , "?" and "!"). We only extracted those sentences which contained more than two words and removed

[1] `http://lxml.de/`

the anchor tags <a> along with any formatting tags <b>, <i> from the sentences. We used NLTK's default tagger to tag the sentences with the simplified tagset [2] and extracted POS-bigrams from each sentence. We used the frequency of each POS-bigram as a feature.

Besides POS tags, several text patterns can help to identify the category of a website. Franchises often indicate price of an item that is being sold in their website. We used a regular expression to extract the patterns of price that is indicated in dollar amount and used the frequency of price patterns in the feature set.

In many cases, franchises also have more than two branches. In order to capture this notion, we extracted the postal addresses from the HTML page. We looked for those URLs that are present in the home page and contain the word stem "about", "contact", "locat", and "map" in the anchor text. We then used a geo extractor [3] that uses a regular expression based method [13] to extract the postal addresses. We used the total number of unique addresses and the number of different provinces as features based on address.

Some organizations often repeat their name many times in their website which can provide useful hint about the category of the website. We extracted organization names from sentences using NLTK's named entity tagger and counted the occurrence of each organization name. We then picked the organization name with highest frequency and used its frequency and occurrence per page as a feature. We also added the total number of unique organization names and the number of organization names per page to the feature set.

4.3 Link Structure and URL-Based Features

Amitay et al. [10] and Lindemann et al [11, 12] have analyzed many structural features that are useful to classify the websites into functional categories. We used 8 URL features and 8 link structure based features from [10–12]. URL features were extracted from a collection of URLs obtained during the depth-crawl of the website. The URL features included average number of digits in the path, number of sub-domains encountered during the crawling of website depth, average path length, average number of slashes in the path, fraction of PDF/PS, HTML and script files, and number of unique file types obtained by analyzing the file extension from the URL.

Link structures are mainly based on the external (a link pointing to a page outside the website) and internal links (a link pointing to any other page within the website) and the depth at which these links were found. The structural features included from [10–12] were average external depth, average internal depth, maximum depth of the website, average depth, total number of unique URLs, fraction of links at the densest depth, average size of the crawled pages in KB, and fraction of the files having javascript in it.

[2] http://www.nltk.org/
[3] http://www.folkarts.ca/geo/

We added more features which were mostly based on click-depth level. We counted the external links, internal links, and outdegree at each depth and calculated the maximum internal links, maximum external links and maximum outdegree occurring at any click-depth. We also computed average external links per depth, average internal links per depth and average outdegree per depth. Furthermore, we created four bins for each depth d indicating the counts of external, internal, repeated internal, and repeated external links at that depth. During the crawling phase, we noticed that some of the private websites had very few internal links at a shallow depth. On the other hand, some of the public websites contained many internal links at a shallow depth and the size of the website rapidly grew along with the depth. By assigning count bins at each depth we intended to capture this property. We also created three bins for the maximum depth of the websites indicating the depth value between 0-5, 6-10 and > 10. We assigned a value of zero or one to each bin depending upon whether or not the maximum depth of the websites falls under the range.

In Canada, many government websites have the domain name of the format *.gov* and *.gc.ca*. In order to capture the essence of top-level domain (TLD) names, we created binary features based on the various patterns of top-level domain names present in the dataset. Our dataset consisted of 7 different patterns of domain names including *.org*, *.com*, *.ca*, *.net*, *.gov*, *.province.ca* (where *.province* can be any Canadian province in its short form) and *.gc.ca*. We combined *.gc.ca*, *.gov* as a single pattern representing government and used a total of six binary features based on the occurrence of the various TLD patterns.

5 Experiments

We used a SVM based one-vs-all BR method [20] to perform multi-label classification. As is the case with one-vs-all approach, we built 4 binary classifiers, one for each class label, and assigned a class label to the website if the prediction of the classifier is positive. All the experiments were performed in a stratified 10-fold cross-validation setting. We created the folds only once and used the same set of folds throughout the experiment for consistency.

We used the RBF (Radial Basis Function) as the kernel for SVM and performed a grid search to select the best values for the parameters γ and C. We searched for the best values of γ and C by maximizing the F-measure in a 5-fold cross validation setting. One of the training folds was used for grid search to obtain the value of γ and C, and the same value of γ and C was used for the rest of the folds in the 10-fold cross-validation setting. LibSVM [14] was used for grid search and SVM based classification.

5.1 Feature Selection

Feature selection is an important step in classification and helps to improve the performance by selecting the most informative features. We performed two steps of feature selection based on document frequency and information gain.

Table 3. Number of features extracted at each click-depth

Feature	click-depth 0	click-depth 1	click-depth 2
BOW	1221	7876	31923
POS-Bigram	248	307	320
Structure	90	90	90

First, we discarded any features having document frequency less than 3. We then followed the ALA (All Labeled Assignment) [15] based document transformation approach and calculated the Information Gain (IG) to select the features by ranking. For a website with multiple labels, the ALA approach requires having duplicate rows of the website each indicating one of the labels in the set of feature vectors used for classification. Throughout this research, we measured the information gain only from the training fold in the 10-fold cross validation setting. Features were ranked by information gain and the top x% of the features were selected for classification, where the value of x ranges from 10 to 100. The information gain was calculated using the implementation of Weka [19].

We also analyzed how the number of features at each click-depth of the website affects the classification performance. Table 3 shows the number of features per click-depth. We can see that the number of features based on bag-of-words (BOW) greatly increases along with the click-depth. On the other hand, numbers of POS-bigrams are not much affected. As we deal with structural features separately, the number of features based on structure remains the same.

Figure 1 shows the performance of bag-of-words at each depth and at various thresholds of information gain. We can see that bag-of-words has the least performance at depth 0. Upon analyzing the websites, we found that the index page of the website seldom contains important words which are able to classify the website into the non-topical categories. Lack of words like "non-profit" which mostly occur in the "About" page at a click depth of one supports this fact. At depth 1, the highest micro f-measure attained by bag-of-words is 0.65, at an IG threshold of 70%. At depth 2, it performs best with a micro f-measure of 0.70 at an IG threshold of 60%. Although depth 2 has the higher micro f-measure, we notice that at 60% IG threshold, bag-of-words require 19153 features at depth 2 whereas depth 1 utilizes 5513 features. This clearly suggests the addition of noise at depth 2. Due to the large number of features, the training time of the classifier is also more than that of depth 1.

5.2 Classification by Various Feature Types

We deal with three types of features: bag-of-words, POS-bigram and named entity distribution, and structural properties which provide three different views of the website. Bag-of-words represent the explicit information provided by the website, POS-bigram and named entity distribution capture the patterns in text and link structure and URL properties give important information about the structure of the website. For each of these views, we created separate feature

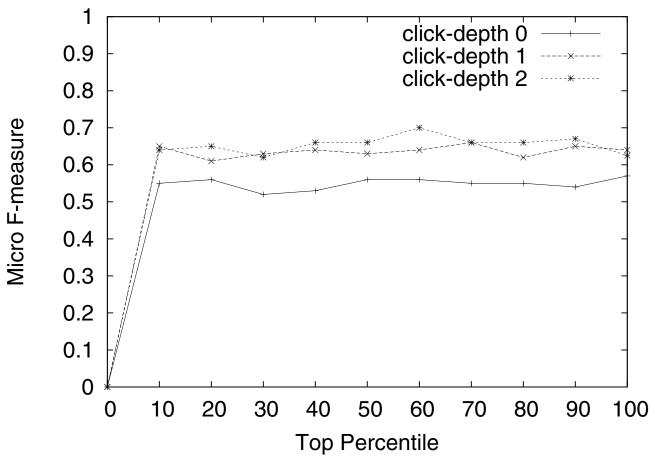


Fig. 1. Micro F-measure for bag-of-words at each click-depth. X-axis denotes top x% of features when ranked by Information Gain where x is between 10 and 100.

vector to classify the websites. Table 4 shows the performance of these features at a depth of zero, one and two. As structural properties were crawled separately, they are not related to the number of pages crawled at each depth and has been reported separately without the depth information. Table 4 also shows the information gain threshold at which each set of features perform at its best. Along with the best performance of each set of features, we also include bag-of-words with all the features (at an IG threshold of 100%) to compare the performance gained by selecting important features by IG ranking. Bag-of-words with a threshold on IG seemed to collect the most informative features and as a result performed well out of the three sets of features; however we should note that it also has the largest number of features compared to our non-bag-of-word features. The best performance of POS bigram and named entities require the use of all of their feature at a depth of zero and one, and 90% of their feature at depth two. At depth one and two, the performance of POS bigram and named entities is comparable to that of structural features. Using only the top 30% of their features, structural features attained a micro and macro f-measure of 0.55.

Table 5 shows some of the top features along with their information gain. We could not provide the full list of features due to the space constraints; nonetheless the list gives the informative strength of each set of features. The list corroborates the fact that explicit words on the website provide the most informative features. Words like "voluntary", "donate", "nonprofit" are good indicators of a website belonging to the non-profit category, whereas "testimonials", "llc", "inc" are good indicators for private and franchise. Keywords "ministry", "government" help to identify websites from the public category. Bag-of-words has the highest value of information gain which is also the reason behind its good performance in classification. Table 5 also shows that the features related to POS

Table 4. Precision, Recall along with Micro and Macro F-measures at each click-depth

			Public		Private		Non-Profit		Franchise		F-measure	
Depth	Features	IG	P	R	P	R	P	R	P	R	Micro	Macro
zero	BOW	100%	0.67	0.53	0.68	0.48	0.46	0.37	0.71	0.62	0.57	0.56
	POS_Bigram_NE	100%	0.64	0.41	0.56	0.34	0.33	0.18	0.62	0.37	0.42	0.41
one	BOW	70%	0.67	0.53	0.73	0.57	0.69	0.5	0.86	0.71	0.65	0.65
	BOW	100%	0.66	0.46	0.75	0.61	0.63	0.37	0.89	0.75	0.64	0.62
	POS_Bigram_NE	100%	0.67	0.53	0.51	0.59	0.42	0.37	0.54	0.57	0.53	0.52
two	BOW	60%	0.66	0.65	0.63	0.75	0.72	0.56	0.73	0.86	0.70	0.69
	BOW	100%	0.76	0.53	0.62	0.63	0.56	0.28	0.82	0.71	0.62	0.59
	POS_Bigram_NE	90%	0.58	0.55	0.55	0.69	0.45	0.28	0.53	0.46	0.53	0.50
-	Structrue	30%	0.56	0.51	0.54	0.67	0.57	0.59	0.55	0.42	0.55	0.55

Table 5. Features ranked by Information Gain

Bag-of-words		**POS Bigram + NE**		**Structure**	
volunt	0.318	[ADV PRO]	0.048	Fraction of PDF/PS files	0.155
committe	0.224	[DET ADV]	0.046	Max. external links per level	0.068
collabora	0.222	[TO P]	0.043	Repeated internal links at depth 5	0.061
...		...		...	
ministri	0.203	[VG ADJ]	0.032	Tot. internal links at depth 5	0.059
fund	0.193	[VG PRO]	0.027	Repeated internal links at depth 6	0.056
donor	0.186	[EX ADJ]	0.023	Max.depth between 0 to 5	0.056
donat	0.150	Price Count	0.023	Tot. external links at depth 6	0.055
govern	0.147	[N NP]	0.022	Avg. digit in domain path	0.049
...		...		...	
nonprofit	0.137	[VG WH]	0.018	Number of file types	0.043
community-bas	0.121	Province Count	0.015	Tot. external links at depth 4	0.039
llc	0.046	[ADJ NP]	0.014	Tot. internal links at depth 3	0.032
testimoni	0.036	Address count	0.010	Avg. outdegree per level	0.031
inc	0.023	Org. name count	0.010	Domain with .gov extension	0.011
...		...		...	

bigram are more informative than the patterns captured from named entities. It also confirms that structural features comprising of internal and external links and related statistics at a depth level are good measures for classification. Public websites generally have many pages at a shallow depth, while most private websites have only a few pages at shallow depths. Also, the maximum depth of a website is a good indicator of a government website (at large depth) and a small private clinic (at small depth).

5.3 Combining Classifiers

As three different sets of features provide different perspective about the website, we try to capture the notion of all three views by combining the individual classifiers built on each view. Kittler et al. [16] show several ways to combine classifiers. We measured the probabilistic output from SVM using Platt's method [17, 18] and used the sum rule to combine the classifiers based on the three feature types. There are 3 classifier predictions (one from each view) for each website sample. For every sample, we summed up the confidence measure of the classifier based on each feature type for every positive prediction. We then only picked the positive predictions where the sum of confidence measure was more than the best performing threshold of 0.65, which was obtained by trying different threshold values in the range of 0.1 to 1.0 at a step of 0.1. Algorithm 1 shows each step of the process.

Alg. 1. Algorithm used to combine the classifiers in a multi-label setting

```
1   for each label, $l$ do
2       for each website,$w$, in test do
3           sum[$w$] := 0
4       for each independent feature set,$f$ do
5           build a classifier,$C$, using $f$,$l$
6               for each website,$w$, in test do
7                   [prediction,confidence] = Classify($w$,$C$)
8                   if prediction == 1.0
9                       add confidence to sum[$w$]
10      for each website,$w$, in test do
11          if sum[$w$] > 0.65
12              assign label $l$ to $w$
```

Table 6 shows the classification result of the combined classifier based on bag-of-words, POS tags and named entities, and structural properties. While combining these classifiers, we choose the IG threshold for each view of feature based on the highest performance of the classifier in terms of micro f-measure as shown in Table 4. For instance, at depth of three we choose classifiers with IG threshold 60%, 90% and 30% for bag-of-words, POS bigrams+Named Entity, and structural property respectively and combine them with the sum rule. Table 6 shows that the combined classifiers performed significantly better than bag-of-words alone at depths of zero and one. The results also show that the performance of the combined classifier at depth 1 is greater than that of bag-of-words at depth 2 and similar to the combined classifier at depth 2. This tells us that collecting words from deeper depth of a website may not be of much help. Moreover, the classifier at depth 1 requires significantly less time to train because the number of features based on bag-of-words is much less than that of depth 2. The performance of the combined classifier at depth 0, although better than bag-of-words alone at depth 0, is the least of all three depths. This indicates that important words available at some shallow depth greater than

Table 6. Result of combining the classifiers using the sum-rule

Depth	Features	Micro F	Macro F
zero	BOW (IG=100%)	0.57	0.56
	Combined	**0.66**	**0.65**
one	BOW (IG=100%)	0.64	0.62
	BOW (IG=70%)	0.65	0.65
	Combined	**0.73**	**0.73**
two	BOW (IG=100%)	0.62	0.59
	BOW (IG=60%)	0.70	0.69
	Combined	**0.73**	**0.74**

zero play an important role in the classification. This also shows that words are important features for non-topical classification and augmenting it with structural properties and POS bigrams and named entities improves the classification.

6 Conclusion

We showed that structural, part-of-speech and named entity based features help to improve the classification performance when combined with the bag-of-words approach. We analyzed performance of each type of feature at various depths of the website. As the number of words increased with the depth of the website, there was a slight increase in performance of bag-of-words from depth 1 to depth 2, at a cost of significant training and crawling time. We achieved similar performance using words from depth of 1, which takes significantly less amount of time for crawling and training the classifier, combined with structural and part-of-speech based features. The performance of bag-of-words increased significantly from depth of 0 to depth 1 showing that index pages at depth zero seldom contain informative word based features to categorize *public*, *private*, *non-profit* and *franchise*. The non-topical features comprising of part-of-speech, named entity and structural features boosted the performance of bag-of-words at all three depths of zero, one and two. However, the classification performance was the least at depth 0, which showed that words are important features even for non-topical classification. We selected features using the ALA based information gain and combined the most powerful classifiers of each feature-type by sum-rule to achieve 8% gain on micro and macro f-measure at depth of 1.

As topical (bag-of-words) and non-topical (POS bigram and named entity, structural property) features provide different views for the website, combining both of them is extremely helpful. Future work for non-topical website classification might involve trying various ways to combine the classifiers based on these independent views such that performance can be further improved.

Acknowledgments. We thank the Public Health Agency of Canada for supporting this research.

References

1. Mishne, G.: Experiments with mood classification in blog posts. In: Proc. of the 1st Workshop on Stylistic Analysis of Text for Information Access, at SIGIR (2005)
2. Kessler, B., Nunberg, G., Schütze, H.: Automatic detection of text genre. In: Proc. of the Association of Computational Linguistics Conference, pp. 32–38 (1997)
3. Turney, P.D.: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In: Proc. of the Association of Computational Linguistics Conference, pp. 417–424 (2002)
4. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. Computing Research Repository, cs. CL/0205070 (2002)
5. Bekkerman, R., Eguchi, K., Allan, J.: Unsupervised non-topical classification of documents. Technical Report IR-472, UMass Amherst (2006)
6. Dai, H.K., Zhao, L., Nie, Z., Wen, J., Wang, L., Li, Y.: Detecting Online Commercial Intention (OCI). In: Proc. of the World Wide Web Conference, pp. 829–837 (2006)
7. Ester, M., Kriegel, H., Schubert, M.: Web Site Mining: A New Way to Spot Competitors, Customers and Suppliers in the World Wide Web. In: Proc. of the Knowledge and Data Discovery Conference (2002)
8. Pierre, J.M.: On the automated classification of Web sites. Linköping Electronic Articles in Computer and Information Science 6 (2001)
9. Eickhoff, C., Serdyukov, P., Vries, A.P.: A combined topical/non-topical approach to identifying web sites for children. In: Proc. of WSDM Conference, pp. 505–514 (2011)
10. Amitay, E., Carmel, D., Darlow, A., Lempel, R., Soffer, A.: The Connectivity Sonar: Detecting Site Functionality by Structural Patterns. In: Proc. of Hypertext and Hypermedia Conference, Nottingham, United Kingdom (2003)
11. Lindemann, C., Littig, L.: Coarse-grained classification of web sites by their structural properties. In: Proc. of the 8th International Workshop on Web Information and Data Management, Arlington, VA (2006)
12. Lindemann, C., Littig, L.: Classification of web sites at super-genre level. In: Mehler, A., et al. (eds.) Genres on the Web: Text, Speech and Language Technology, vol. 42, pp. 211–235. Springer (2010)
13. Yu, Z.: High Accuracy Postal Address Extraction from Web Pages. Master's Thesis. Dalhousie University. Halifax, Nova Scotia, Canada (2007)
14. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. on Intelligent Systems and Technology 2, 27:1–27:27 (2011)
15. Chen, W., Yan, J., Zhang, B., Chen, Z., Yang, Q.: Document transformation for multi-label feature selection in text categorization. In: Proc. of Seventh IEEE International Conference on Data Mining, USA, pp. 451–456 (2007)
16. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On Combining Classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence 20, 226–239 (1998)
17. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in Large Margin Classifiers (2000)
18. Lin, H.-T., Lin, C.-J., Weng, R.C.: A Note on Platt's Probabilistic Outputs for Support Vector Machines. Machine Learning 68(3), 267–276 (2007)
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
20. Tsoumakas, G., Katakis, K.: Multi label classification: An overview. International Journal of Data Warehousing and Mining 3(3) (2007)

Improving Cross-Document Knowledge Discovery Using Explicit Semantic Analysis

Peng Yan and Wei Jin

Department of Computer Science, North Dakota State University
1340 Administration Ave., Fargo, ND 58102, USA
{peng.yan,wei.jin}@ndsu.edu

Abstract. Cross-document knowledge discovery is dedicated to exploring meaningful (but maybe unapparent) information from a large volume of textual data. The sparsity and high dimensionality of text data present great challenges for representing the semantics of natural language. Our previously introduced Concept Chain Queries (CCQ) was specifically designed to discover semantic relationships between two concepts across documents where relationships found reveal semantic paths linking two concepts across multiple text units. However, answering such queries only employed the Bag of Words (BOW) representation in our previous solution, and therefore terms not appearing in the text literally are not taken into consideration. Explicit Semantic Analysis (ESA) is a novel method proposed to represent the meaning of texts in a higher dimensional space of concepts which are derived from large-scale human built repositories such as Wikipedia. In this paper, we propose to integrate the ESA technique into our query processing, which is capable of using vast knowledge from Wikipedia to complement existing information from text corpus and alleviate the limitations resulted from the BOW representation. The experiments demonstrate the search quality has been greatly improved when incorporating ESA into answering CCQ, compared with using a BOW-based approach.

Keywords: Knowledge Discovery, Semantic Relatedness, Cross-Document Knowledge Discovery, Document Representation.

1 Introduction

Text is the most traditional method for information recording and knowledge representation. Text mining focuses on mining high-quality information from mass text. The widely used text representation is based on the Bag of Words (BOW) model which represents text as a collection of words, however, this representation is limited to the terms appearing in the text literally, which could lead to great semantic loss because terms that are closely related to each other will be viewed as completely irrelevant unless they are both mentioned in the text. Our previous work [1] introduced a special case of text mining focusing on detecting semantic relationships between two concepts across documents, which we refer to as Concept Chain Queries (CCQ). A concept chain query involving concepts A and B has the following meaning: find the most plausible relationship between concept A and concept B assuming that one or more instances of both concepts occur in the corpus, but not necessarily in the same document. For example, both may be football lovers, but mentioned in different documents. The previous solution used the BOW model for text representation, i.e., relationships between important terms that do not co-appear literally in the text are neglected, and thus could not contribute to the generation of the links. For instance, "Albert Gore" is closely related to "George W. Bush" since two men together produced the most controversial presidential election in 2000, which was the only time in American history that the Supreme Court has determined the outcome of a presidential election. However, "Albert Gore" will not be taken into account if it does not occur in the document collection where the concept chain queries are performed on.

In our BOW based approach for answering concept chain queries [1], the weight of each term was measured by its $TF * IDF$ based value in the document collection. In this work, we propose to further improve the model by incorporating the Explicit Semantic Analysis (ESA) technique [3]. Basically, ESA maps a given text or a term to a conceptual vector space which is spanned by all Wikipedia articles, and thus more background knowledge can be integrated into semantic representation of each term, which is able to help overcome the shortcomings resulted from the BOW representation. We also attempt to identify only the most relevant concepts generated from ESA for semantic relatedness computation. To achieve this goal, we further develop a sequence of heuristic steps for noise removal. Therefore, our method will not bring as much noise as [3] does. To validate the proposed techniques, a significant amount of queries covering different scenarios were conducted to show that we could rank those most relevant concepts to the given topics in the top positions.

Our contribution of this effort can be summarized as follows. First, compared with the solution using a BOW based approach, the proposed technique is able to provide a much more comprehensive knowledge repository to support various queries and effectively complements existing knowledge contained in text corpus. Second, we further improve the ESA technique by providing a sequence of heuristic strategies to clean the interpretation vector which we observe contains a fair amount of noise and is not precise enough to represent the contextual clues related to topics of interest. Third, to the best of our knowledge, little work has been done to consider ESA as an effective aid in cross-document knowledge discovery. In this work, built on the traditional BOW text representation for content analysis, we successfully integrate ESA into the discovery process to help measure the semantic relatedness between concepts. We envision this integration would also benefit other related tasks such as question answering and cross–document summarization. Last, the approach presented here is able to boost concepts that are most closely related to the topics to higher rankings, compared to the widely used TF-IDF based ranking scheme.

The remainder of this paper is organized as follows: Section 2 describes the related work. Section 3 briefly introduces how concept chain queries work. In Section 4, we discuss the ESA model in detail and present our method to integrate the ESA approach into concept chain queries. Experimental results are presented and analysed in Section 5, and is followed by the conclusion and future work given in Section 6.

2 Related Work

There has been work on discovering connections between concepts across documents using social network graphs, where nodes represent documents and links represent connections (typically URL links) between documents. However, much of the work on social network analysis has focused on special problems, such as detecting communities [7] [12]. Our previous work [1] introduced Concept Chain Queries (CCQ), a special case of text mining focusing on detecting cross-document links between concepts in general document collections (without hyperlinks). This was motivated by Srinivsan's closed text mining algorithm which was built within the discovery framework established by Swanson and Smalheiser [4]. Specifically, the solution proposed attempted to generate concept chains based on the "Bag of Words" (BOW) representation and extended the technique in [2] by considering multiple levels of interesting concepts instead of just one level as in the original method. Each document in [1] was represented as a vector containing all the words appearing in the relevant text snippets in the corpus but did not take any auxiliary knowledge into consideration, whereas in this new solution, in addition to content analysis, we further examine the potential of integrating the Explicit Semantic Analysis (ESA) technique to better serve this task which effectively incorporates more comprehensive knowledge from Wikipedia. Related works attempting to overcome the limitations of BOW approach and integrate the background knowledge into text representation have also been reported in categorization and knowledge discovery applications. For example, WordNet was utilized in [6] to improve the BOW text representation and Scott et al [8] proposed a new representation of text based on WordNet hypernyms. These WordNet-based approaches were shown to alleviate the problems of BOW model but are subject to relatively limited coverage of Wordnet compared to Wikipedia, the world's largest knowledge base to date. Gabrilovich et al [5] applied machine learning techniques to Wikipedia and proposed a new method to enrich document representation from this huge knowledge repository. Specifically, they built a feature generator to identify most relevant Wikipedia articles for each document, and then used concepts corresponding to these articles to create new features. The experimental evaluation showed great improvements across a diverse collection of datasets. However, with the process of feature generation so complicated, a considerable computational effort is required.

In terms of improving semantic relatedness computation using Wikipedia, Gabrilovich et al also [3] presented a novel method, Explicit Semantic Analysis (ESA), for fine-grained semantic representation of unrestricted natural language texts. Using this approach, the meaning of any text can be represented as a weighted vector of Wikipedia-based concepts (articles), called an interpretation vector [3]. [3] also discussed the problem of possibly containing noise concepts in the vector, especially for text fragments containing multi-word phrases (e.g., multi-word names like George Bush). Our proposed solution is motivated by this work and to tackle the above problem we further develop a sequence of heuristic strategies to filter out irrelevant concepts and clean the vector. Another interesting work is an application of ESA in a cross-lingual information retrieval setting to allow retrieval across languages [9]. In that effort the authors performed article selection to filter out those irrelevant Wikipedia articles (concepts). However, we observe the selection process resulted in the loss of many dimensions in the following mapping process, whereas in our proposed approach, the process of article selection is postponed until two semantic profiles have been merged so that the semantic loss could be possibly reduced to the minimum.

3 Concept Chain Queries

As described earlier, concept chain query (CCQ) is attempting to detect links between two concepts (e.g., two person names) across documents. A concept chain query involving concept A and concept B intends to find the best path linking concept A to concept B. The paths found stand for potential conceptual connections between them.

3.1 Semantic Profile for Topic Representation

A semantic profile is essentially a set of concepts that together represent the corresponding topic. To further differentiate between the concepts, semantic type (ontological information) is employed in profile generation. Table 1 illustrates part of semantic type - concept mappings. Thus each profile is defined as a vector composed of a number of semantic types.

$$profile(T) = \{ST_1, ST_2, ..., ST_n\} \tag{1}$$

Where ST_i represents a semantic type to which the concepts appearing in the topic-related text snippets belong. We used sentence as window size to measure relevance of appearing concepts to the topic term. Under this representation each semantic type is again referred to as an additional level of vector composed of a number of terms that belong to this semantic type.

$$ST_i = \{w_{i,1}m_1, w_{i,2}m_2, ..., w_{i,n}m_n\} \tag{2}$$

Where m_j represents a concept belonging to semantic type ST_i, and $w_{i,j}$ represents its weight under the context of ST_i and sentence level closeness. When generating the profile we replace each semantic type in (1) with (2).

In (2), to compute the weight of each concept, we employ a variation of $TF * IDF$ weighting scheme and then normalize the weights:

$$w_{i,j} = s_{i,j} / highest(s_{i,l}) \tag{3}$$

Where $l = 1, 2, ..., r$ and there are totally r concepts for ST_i, $s_{i,j} = df_{i,j} * Log(N / df_j)$, where N is the number of sentences in the collection, df_j is the number of sentences concept m_j occurs, and $df_{i,j}$ is the number of sentences in which topic T and concept m_j co-occur and m_j belongs to semantic type ST_i. By using the above three formulae we can build the corresponding profile representing any given topic.

Table 1. Semantic Type - Concept Mapping

Semantic Type	**Instances**
Religion	Islam, Muslim
Human Action	attack, killing, covert action, international terrorism
Leader	vice president, chief, governor
Country	Iraq, Afghanistan, Pakistan, Kuwait
Infrastructure	World Trade Centre
Diplomatic Building	consulate, pentagon, UAE Embassy

3.2 Concept Chain Generation

We adapt Srinivasan's closed discovery algorithm [2] to build concept chains for any two given topics. Each concept chain generated reveals a plausible path from concept A to concept C (suppose A and C are two given topics of interest). The algorithm of generating concept chains connecting A to C is composed of the following three steps.

1. Conduct independent searches for A and C. Build the A and C profiles. Call these profiles AP and CP respectively.
2. Compute a B profile (BP) composed of terms in common between AP and CP. The weight of a concept in BP is the sum of its weights in AP and CP. This is the first level of intermediate potential concepts.
3. Expand the concept chain using the created BP profile together with the topics to build additional levels of intermediate concept lists which (i) connect the topics to each concept in BP profile in the sentence level within each semantic type, and (ii) also normalize and rank them (as detailed in section 3.1).

4 Utilizing Wikipedia Knowledge in Concept Chain Queries

Wikipedia is currently the largest human-built repository in the world. In this effort, we are attempting to improve our query model through integrating the Explicit Semantic Analysis (ESA) [3] technique that uses the space of Wikipedia articles to compute semantic relatedness between texts. In ESA, each term is represented by a vector storing the term's association strengths to Wikipedia articles and each text fragment is mapped to a weighted vector of Wikipedia concepts called an interpretation vector. Therefore, computing semantic relatedness between any two text fragments is naturally transformed into computing the Cosine similarity between interpretation vectors of two texts.

4.1 Document Representation with ESA

In ESA, each article in Wikipedia is treated as a Wikipedia concept (the title of an article is used as a representative concept to represent the article content), and each document given is represented by an interpretation vector containing related Wikipedia concepts (articles) with regard to this document. Formally, a document d can be represented as follows:

$$\phi(d) = < as(d, a_1), \ldots, as(d, a_n) > \tag{4}$$

Where $as(d, a_i)$ denotes the association strength between document d and Wikipedia article a_i. Suppose d is spanned by all words appearing in it, i.e., $d = < w_1, w_2, \ldots, w_j >$, and the association strength $as(d, a_i)$ is computed by the following function:

$$as(d, a_i) = \sum_{w_j \in d} tf_d(w_j) tf \bullet idf_{a_i}(w_j) \tag{5}$$

Where $tf_d(w_j)$ is the occurrence frequency of word w_j in document d, and $tf \bullet idf_{a_i}(w_j)$ is the $tf \bullet idf$ value of word w_j in Wikipedia article a_i. As a result, the vector for a document is represented by a list of real values indicating the association strength of a given document with respect to Wikipedia articles. By using efficient indexing strategies such as single-pass in memory indexing, the computational cost of building these vectors can be reduced to within 200-300 ms. In concept chain queries, the topic input is always a single concept (a single term or phrase), and thus equation (5) can be simplified as below as $tf_d(w_j)$ always equals to 1:

$$as(d, a_i) = \sum_{w_j \in d} tf \bullet idf_{a_i}(w_j) \tag{6}$$

4.2 Interpretation Vector Cleaning

As discussed above, the original ESA method is subject to the noise concepts introduced, especially when dealing with multi-word phases. For example, when the input is “George Bush”, the generated interpretation vector will contain a fair amount of noise concepts such as “That’s My Bush”, which is actually an American comedy television series. This Wikipedia concept (article) is selected and ranked in the second place in the list because “Bush” occurs many times in the article “That’s My Bush”, but obviously this article is irrelevant to the given topic “George Bush”.

In order to make the interpretation vector more precise and relevant to the topic, we have developed a sequence of heuristics to clean the vector. Basically, we use a modified Levenshtein Distance algorithm to measure the relevance of the given topic to each Wikipedia concept generated in the interpretation vector. Instead of using allowable edit operations of a single character to measure the similarity between two strings as in the original Levenshtein Distance algorithm, we view a single word as a

```
Input:  a topic T of interest
        an interpretation vector V representing the topic T

Output: a cleaned Wikipedia-based concept vector V' representing the topic T

    1. If T is a single word topic, then count the number of occurrences of T in the article texts represented by each concept $v_i$ in V, respectively. If T occurs more than 3 times, then keep $v_i$ in V, otherwise, remove $v_i$ from V.

    2. If T is a multi-word topic, then the adapted Levenshtein distance algorithm applies to measure the relevance of each Wikipedia concept (article) $v_i$ in V to topic T.

        2.1. If NumOfWords(T) $\le 2$, then extract all text snippets $TS_j$ within the window size NumOfWords(T) from the article text of $v_i$. If there exists a j such that LevenshteinDistance(T, $TS_j$) = 0, then keep $v_i$ in V, otherwise, remove $v_i$ from V.

        2.2. If NumOfWords(T) > 2, then extract all text snippets $TS_j$ within the window size NumOfWords(T)+1 from the article text of $v_i$. If there exists j such that LevenshteinDistance(T, $TS_j$) $\le 1$, then keep $v_i$ in V, otherwise, remove $v_i$ from V.
```

Fig. 1. The Interpretation Vector Cleaning Procedure

unit for edit operations, and thus the adapted algorithm can be used to compute the similarity between any two text snippets. The heuristic steps used to remove noise concepts are illustrated in Figure 1.

4.3 Integrating ESA into Concept Chain Queries

Given the advantages of using ESA as a semantic representation method, we integrate the kernel of ESA into our concept chain queries, aiming to improve search quality. Specifically, we build interpretation vectors for both of the two given topics as well as each intermediate concept in the merged BP profile, and then apply the cleaning procedure on these vectors to remove noise concepts. Finally, we compute Cosine similarities between interpretation vectors for the topics and each concept in the BP profile. The new model of answering concept chain queries is illustrated in Figure 2.

A combination of techniques of BOW representation and ESA method is considered in this new solution, and therefore two types of ranking schemes are integrated as follows.

TF*IDF-Based Similarity. As the most widely used document representation, the BOW representation has demonstrated its advantages. It is simple to compute and strictly sticking to the terms occurring in the document, thereby preventing outside noise concepts that do not appear in the document from flowing into the feature space of the representation. Given these benefits, a variation of $TF * IDF$ weighting scheme under the context of BOW representation and semantic types (detailed in Section 3.1) is incorporated into our final ranking where a sentence window is employed to further filter the noise that may incur. We call this kind of similarity TF*IDF-based similarity.

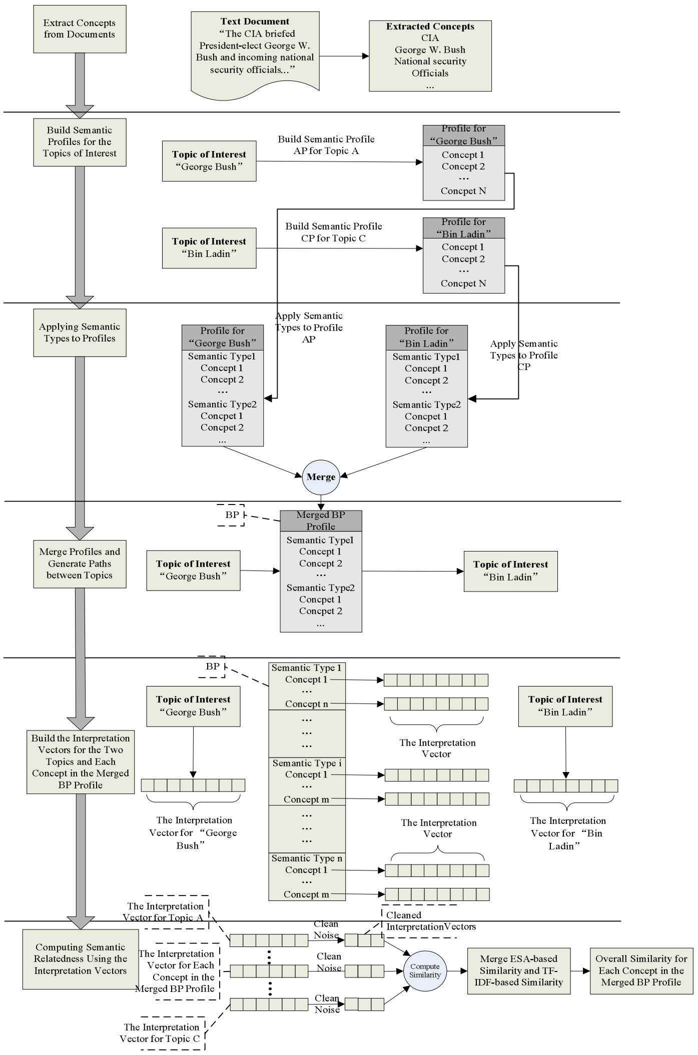

Extract Concepts from Documents
Text Document "The CIA briefed President-elect George W. Bush and incoming national security officials..."
Extracted Concepts CIA George W. Bush National security Officials ...
Build Semantic Profiles for the Topics of Interest
Topic of Interest "George Bush"
Build Semantic Profile AP for Topic A
Profile for "George Bush" Concept 1 Concept 2 ... Concpet N
Topic of Interest "Bin Ladin"
Build Semantic Profile CP for Topic C
Profile for "Bin Ladin" Concept 1 Concept 2 ... Concpet N
Applying Semantic Types to Profiles
Apply Semantic Types to Profile AP
Apply Semantic Types to Profile CP
Semantic Type1 Concept 1 Concept 2 ... Semantic Type2 Concept 1 Concept 2 ...
Merge
Merge Profiles and Generate Paths between Topics
BP
Merged BP Profile
Build the Interpretation Vectors for the Two Topics and Each Concept in the Merged BP Profile
Semantic Type 1 Concept 1 ... Concept n
Semantic Type i Concept 1 ... Concept m
Semantic Type n Concept 1 ... Concept m
The Interpretation Vector
The Interpretation Vector for "George Bush"
The Interpretation Vector for "Bin Ladin"
Computing Semantic Relatedness Using the Interpretation Vectors
The Interpretation Vector for Topic A
The Interpretation Vector for Each Concept in the Merged BP Profile
The Interpretation Vector for Topic C
Clean Noise
Cleaned InterpretationVectors
Compute Similarity
Merge ESA-based Similarity and TF-IDF-based Similarity
Overall Similarity for Each Concept in the Merged BP Profile

Fig. 2. The new model of answering concept chain queries

ESA-Based Similarity. Unlike the BOW model, ESA makes use of the knowledge outside the documents themselves to compute semantic relatedness. It well compensates for the semantic loss resulted from the BOW technique. The relatedness between two concepts in ESA is computed using their corresponding interpretation vectors containing related concepts derived from Wikipedia. In the context of concept chain queries, we compute the Cosine similarity between interpretation vectors of topic A and each concept V_i in the intermediate BP profile, as well as between topic C and each concept V_i, and take the average of two Cosine similarities as the overall similarity for each concept V_i in BP. We call this kind of similarity ESA-based Similarity.

Integrating TF*IDF-Based Similarity and ESA-Based Similarity into the Final Ranking. The TF*IDF-based similarity and ESA-based similarity are finally linearly combined to form a final ranking for concepts generated in the intermediate profiles:

$$S_{overall} = \lambda \bullet S_{TFIDF} + (1-\lambda) S_{ESA} \quad (7)$$

Where λ is a tuning parameter that can be adjusted based on the preference on the two similarity schemes in the experiments. S_{TFIDF} refers to the TF*IDF-based similarity and S_{ESA} the ESA-based similarity.

5 Empirical Evaluation

We performed our evaluation using the 9/11 counterterrorism corpus. The Wikipedia snapshot used in the experiments was dumped on April 05, 2011.

5.1 Processing Wikipedia Dumps

As an open source project, the entire content of Wikipedia is easily obtainable. All the information from Wikipedia is available in the form of database dumps that are released periodically, from several days to several weeks apart. The version used in this work was released on April 05, 2011, which was separated into 15 compressed XML files and totally occupies 29.5 GB after decompression, containing articles, templates, image descriptions, and primary meta-pages. We leveraged MWDumper [13] to import the XML dumps into our MediaWiki database, and after the parsing process, we identified 5,553,542 articles.

5.2 Evaluation Data

We performed concept chain queries on the 9/11 counterterrorism corpus. This involves processing a large open source document collection pertaining to the 9/11 attack, including the publicly available 9/11 commission report. The report consists of Executive Summary, Preface, 13 chapters, Appendix and Notes. Each of them was considered as a separate document resulting in 337 documents. The whole collection was processed using Semantex [11] and concepts were extracted and selected as shown in Table 1. The evaluation data generated includes 1,346 chains of length 1, 6,709 chains of length 2, 6,036 chains of length 3, and 400 chains of length 4.

5.3 Experimental Results

Parameter Settings. As mentioned in Section 4.3, we use a combination of TF*IDF-based similarity and ESA-based similarity to rank the links detected by our system. λ in Equation 7 is a parameter that needs to be tuned so that similarities between concepts best match the judgements from our assessors. To accomplish this, we first built a set of training data composed of 10 query pairs randomly selected from the evaluation set, and then generated B profiles for each of them using our proposed method. Among each B profile, we selected the top 5 concepts (links) within each semantic type, and compared their rankings with the assessors' judgements. The value of λ was tuned in the range of [0.1, 1] and the best performance was obtained when λ was set to 0.2.

Query Results. The effectiveness of our approach is measured by precision and recall of the concept chains the system generated. Table 2 makes a comparison between the searching results of concept chain queries using a BOW-based approach (CCQ-BOW) and concept chain queries with ESA integrated (CCQ-BOW-ESA). In Table 2, S_N means we only keep the top *N* concepts within each semantic type in the searching results and L_N indicates the resulting chains of length *N*. The entries of Table 2 stand for the precision and recall values (P for precision, while R for recall).

Table 2. Searching Results of Top N Concepts

		CCQ-BOW/CCQ-BOW-ESA					
		S1	**S2**	**S5**	**S10**	**S20**	**S30**
L1	P	0.664/0.710	0.659/0.686	0.656/0.663	0.664/0.672	0.662/0.675	0.668/0.674
	R	0.340/0.339	0.474/0.474	0.627/0.643	0.767/0.782	0.848/0.878	0.902/0.918
L2	P	0.741/0.762	0.733/0.755	0.714/0.749	0.714/0.744	0.709/0.742	0.710/0.784
	R	0.269/0.285	0.390/0.410	0.547/0.591	0 660/0.720	0.746/0.825	0.784/0.866
L3	P	0.754/0.769	0.745/0.765	0.723/0.761	0.721/0.754	0.716/0.751	0.716/0.748
	R	0.261/0.279	0.380/0.403	0.538/0.585	0.649/0.713	0.734/0.819	0.771/0.860
L4	P	0.261/0.432	0.296/0.369	0.578/0.693	0.252/0.286	0.261/0.279	0.260/0.268
	R	0.233/0.340	0.450/0.480	0.670/0.700	0.630/0.810	0.940/0.940	0.970/0.970

Table 3. Searching Results of Query Pair "Abdel Rahman :: Blind Sheikh"

BP Term	**Semantic Type**	**Term Rank**					
		CCQ-BOW			**CCQ-BOW-ESA**		
		P=5	P=10	P=20	P=5	P=10	P=20
Islamic Group	Corporation	4	4	4	2	2	2
Saudi Arabia	Country	3	3	3	1	1	1
CIA	Government	-	7	7	3	3	3
President Clinton	Man	-	9	9	-	8	8
Jihad	Organization	-	-	-	-	-	15
Khifa Refugee Center	Organization	-	7	7	-	7	7
Omar Abdel Rahman	Man	1	1	1	1	1	1
Terrorist	Person	-	10	10	2	2	2
Islamist	Religion	-	9	9	3	3	3
Abdullah	Man	-	-	11	-	7	7
Muslim	Religion	4	4	4	2	2	2
Government	Government	-	8	8	2	2	2
Bin Ladin	Person	3	3	3	2	2	2
Attack	Human Action	3	3	3	2	2	2

Table 3 shows an example of the improvement of concept rankings of key BP terms by integrating ESA into answering concept chain queries. The terms in this table were produced by running a query: "Abdel Rahman" and "Blind Sheikh". P=5 means we keep the top 5 concepts within each semantic type of BP. Each entry is the ranking position of the corresponding key concept in BP. The entry value "-" means the concept cannot be found in the results. It is obvious that for most of the key BP terms, the ranks are boosted. The concepts in Table 3 are strongly related to Abdel Rahman who is also known as "The Blind Sheikh". For instance, Abdel Rahman was a blind Egyptian Muslim leader and accused of being the leader of "The Islamic Group" which is considered as a terrorist organization by the United States government. Therefore, the concepts "Islamic Group", "Islamist", "Muslim" and "Terrorist" typically characterize his identity.

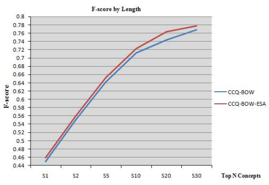


Fig. 3. Result of Chains of Length 1

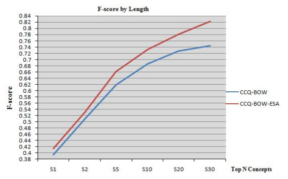


Fig. 4. Result of Chains of Length 2

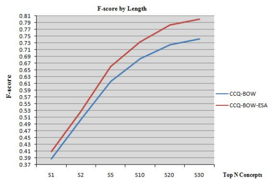


Fig. 5. Result of Chains of Length 3

We further use $F-measure$ to interpret the query results as a weighted average of the precision and recall. Figures 3 through 5 compare the searching results graphically between concept chain queries with BOW (CCQ-BOW) and concept chain queries with ESA integrated (CCQ-BOW-ESA) in terms of how the integrated solution would improve the query model for chains of different lengths. The X-axis indicates the number of concepts kept in each semantic type in the searching results (S_N means we keep the top N), while the Y-axis indicates the *F-score*. We can see that the achieved *F-score* continues to rise as we increase the number of top concepts kept in the search results, and the most significant upward trend was observed when the number of top concepts kept increases from 1 and 5. It is also obvious that our new model consistently achieves better performances for different lengths than the solution based on a BOW approach.

6 Conclusion and Future Work

This paper proposes a new solution for improving cross-document knowledge discovery through our previously introduced concept chain queries, which focus on detecting semantic relationships between topics across documents where revealed semantic paths may lead to early discovery of hypotheses. In this effort, we propose a hybrid approach that integrates Wikipedia knowledge into the traditional BOW model, which complements existing knowledge in text corpus and further improves search quality. We present experiments that demonstrate the effectiveness of this new approach. Specifically, the key terms representing significant relationships between topics are greatly boosted, compared with the method using the $TF*IDF$ ranking scheme.

Future direction includes exploration of other potential resources provided by Wikipedia to further improve query processing, such as categories that relevant Wiki articles belong to and the underlying category hierarchy. These valuable information resources may be combined with our defined semantic types to further contribute to ontology modeling. As a cross language knowledge base, we also plan to combine Wiki knowledge in a cross-lingual setting to better serve different query purposes.

Acknowledgments. We would like to thank the anonymous reviewers for their comments and suggestions. This publication was made possible by Grant Number EPS-0814442 from NSF ND EPSCoR Infrastructure Improvement Programs (IIP).

References

1. Jin, W., Srihari, R.K.: Knowledge Discovery across Documents through Concept Chain Queries. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006) Workshop on Foundation of Data Mining and Novel Techniques in High Dimensional Structural and Unstructured Data, Hong Kong, China (2006)
2. Srinivasan, P.: Text Mining: Generating hypothesis from Medline. JASIST 55, 396–413 (2004)
3. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1606–1611 (2007)
4. Swason, D.R., Smalheiser, N.R.: Implicit text linkage between Medline records; using Arrowsmith as an aid to scientific discovery. Library Trends 48(1), 48–59 (1999)
5. Gabrilovich, E., Markovitch, S.: Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In: AAAI 2006 (2006)
6. Hotho, A., Staab, S., Stumme, G.: Wordnet improves text document clustering. In: Proceedings of the Semantic Web Workshop at SIGIR 2003 (2003)
7. Gibson, D., Kleinberg, J., Raghavan: Inferring web communities from link topology. In: Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia, pp. 225–234
8. Scott, S., Matwin, S.: Text Classification Using WordNet Hypernyms, pp. 45–52. Association for Computational Linguistics (1998)
9. Sorg, P., Cimiano, P.: Cross-lingual Information Retrieval with Explicit Semantic Analysis. In: CLEF Workshop (2008)
10. Wang, P., Hu, J., Zeng, H.-J., Chen, L., Chen, Z.: Improving Text Classification by Using Encyclopedia Knowledge. In: Seventh IEEE International Conference on Data Mining (ICDM) (2007)
11. Srihari, R.K., Li, W., Niu, C., Cornell, T.: InfoXtract: A Customizable Intermediate Level Information Extraction Engine. Journal of Natural Language Engineering (2006)
12. Faloutsos, C., McCurley, K., Tomkins, A.: Fast discovery of connection subgraphs. In: Proceeding of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2004, pp. 118–127 (2004)
13. MWDumper. Software, `http://www.mediawiki.org/wiki/Manual:MWDumper`

Implementing a Data Lineage Tracker

Colin Puri, Doo Soon Kim, Peter Z. Yeh, and Kunal Verma

Accenture Technology Labs,
50 W. San Fernando St. Suite 1200, San Jose,
California 95113, U.S.A
{colin.puri,doo.soon.kim,peter.z.yeh,k.verma}@accenture.com

Abstract. Everyday business users face the tracking of the origin of information used in calculations and business decisions. Knowing the origin and lineage of data can help in the decision making process, provide a clear audit trail for regulation, and answer key questions such as: who, what, where, when, why, and how. In the case of tracking data lineage, many issues and challenges arise in trying to track and support a heterogeneous enterprise environment. This paper presents one method of tackling data lineage to answer the questions needed for business users, for both new and old applications in a heterogeneous infrastructure environment. Using trace logs from data sources, we show how our system performs by effectively tracking data lineage and determining data flows of information as it moves from one data source to another through the execution of applications. Utilizing SQL and NoSQL systems, we demonstrate the recall and precision of our proposed data lineage tracking system.

Keywords: Lineage, Data, Data Lineage, Meta-data Management, Data Flow.

1 Introduction

1.1 What Is Data Lineage

The fluidity of information flows from all points within a company and passes through systems and subsystems. Applications consume and export data and may or may not modify the data. Data can eventually become an aggregate of several other data points. Data may eventually be stored in one or more locations and can reside in databases, documents, spreadsheets, or even emails. Along the way, the origination point of data, its lineage information (who, what, where, when, why, and how) is obscured, may contain gaps, or may be lost. Data lineage is meta-data that captures information about the history and provenance of data, which is critical to answering key business questions such as:

- Who created, who modified, which business process touched, and what modifications were introduced into the data?
- What were the previous values of the data and why were modifications made?
- Were there any elements of uncertainty introduced into the data?

1.2 Why Is Data Lineage Important and Our Motivation

Many analysts, consultants, managers and business users have limited time and limited resources to answer questions concerning calculations and the origins of information. Highly trafficked documents and databases contain a plethora of data. With the copious amounts of data come errors, omissions, and modifications through its lifetime. Tracking down information concerning the lifecycle of data values can take time as conversations and emails pass rapidly between people, it can be prone to errors due to interactions as individuals endeavor to meet deadlines to answer questions (e.g. what is the origin and accuracy of the data). Integrating a tool to answer many of these questions can reduce efforts involved and decrease the time to resolve these issues. The lineage of data can be imperative for businesses such as financial institutions that must abide by governmental regulations, such as Basel II [1] and the Sarbanes-Oxley Act [2] as enforced by the Federal Reserve [3] and the U.S. Securities and Exchange Commission (SEC) [4]. Such regulations require knowledge of the life and timeline of data and institutions must provide proof to the veracity and authenticity of information within a timely manner (i.e. section 409 of the Sarbanes-Oxley Act [2]).

Our motivation is to develop a differentiated technology asset that can effectively capture lineage, manage meta-data, track and produce data flows, infer reasoning for changes, perform anomaly detection, and report rich data lineage information across heterogeneous platforms and applications. The utilization of the captured information would be in the service of key enterprise activities such as producing shorter decision-making cycles, enabling more efficient and cost-effective compliance and audit cycles, enhancing data loss prevention (especially in data aggregation situations), allowing for finer grain access control, and enriching data analytics. The current aim is to integrate our approach with the creation of a data lineage infrastructure that provides lineage tracking and reference capabilities. In addition, we look to integrate common business user applications and tools with the support of an omnipresent back-end tracking capable infrastructure.

1.3 Previous Work and Goals

Research has been performed to trace data lineage and data flow [5] [6] in the scope of a data warehouse environment. Data lineage has also been examined in the context of file systems [7]. Previous research has proposed new ways of tracking information and bringing to light the pitfalls of tracing data lineage on a large scale. This includes looking at lineage as a set of probabilities [8]. However, much of the research has led to limited implementation in the enterprise environment or the concepts in general have had limited domain use. Despite the importance of data lineage, major software producers have seen limited manifestations in their business intelligence platforms [9]. The implementations that exist work well in a homogenous environment (e.g. IBM [10], Oracle [11]), but are lacking when it comes to a heterogeneous environment and is limited to the content contained within change logs. It is often too coarse for many enterprise activities.

In order to fully track lineage, a solution must be able to handle data as it flows from one system to another, be it between like or unalike systems such as an RDBMS systems (e.g. Microsoft SQL Server [12], MySQL [13], Oracle[11]) or NoSQL database systems (e.g. Apache Cassandra [14]). Furthermore, a solution must conceptualize data as it flows from one data source to another, via an application as that application may or may not change the data along its path (e.g. Microsoft Excel [15], etc.).

Our goal is to create a tool that can effectively track and produce data flows with high precision as traces increase in size and complexity while producing results in a timely manner. This paper introduces our framework for tracking data lineage. We lay the foundation from the perspective of the user interface, describe the architecture and algorithm, experimental setup, evaluation of results, and finally this paper finishes with a conclusion of our findings.

2 Application User Interface

In order for full utilization of the system, our web interface provides administration configuration functionality for adding: assets, resources, and applications. Assets are analogous to the server or computer on which a data source of interest may reside. Resources are analogous to the actual data source of interest. Applications refer to programs that perform manipulations and retrieval of data from the data sources. Assets and resources are required as a part of the setup and configuration, whereas applications are optional. Each resource resides on an asset, contains a data source name, a login, a password, the SQL dialect, and type of lineage modality to utilize for tracking. In the case of a monitor modality choice, a user must also provide the location of the trace logs as well as the frequency of polling. If the available assets are unknown, the data lineage tool has auto-discover capabilities.

A user may configure an application, although it is not required. A registered application provides the ability to enrich discovered data lineage with reasons for operations. Application registration provides additional information when operating in the monitor modality as acquisition of information performs after-the-fact.

While an administrator can configure the tool, our real goal is to empower the business user with information to execute their job. In a scenario where a business user may open a document and distress over numbers presented before them, integration of our tool into a browser or other software allows for easy utilization. Through tool integration, acquisition of information concerning any values executes quickly with reduced effort required to contact people. The data lineage tool allows for integration with a myriad of external business user tools using a web service interface. The data lineage tool integrates using the web configuration interface with Microsoft Excel (Fig. 1(a)) and Microsoft Word, for example. The integration with external tools such as the Microsoft Office suite also allows the common business user to track data and embeds data persistent lineage information within documents.

A unique feature of the tool integration is the ability to monitor and detect anomalies. When a value is determined to be a certain degree different from previous values (e.g. in the case of a numeric value a rule may be added to the system in its configuration where an anomaly is detected if the current and previous value differ by 20%), it is highlighted (shown in the right pane in Fig. 1(a)) in light red in order to alert a user. This provides a starting point for an individual to investigate problems that may exist within a system. The anomaly detection capability is based on rules that are provided to the data lineage platform. In the future, we look to integrate previous work in the detection of data quality rules [16] for automatic anomaly detection.

The data lineage tool provides information of a value at the table level; however, the discovered data flow can enrich the experience by illustrating how the data arrived at its current value as it passed from one database to another through applications (see Fig. 1(b)). The data lineage tool is able to track data flow not only within a database, but also across multiple databases. Additionally, we have integrated our anomaly detection mechanism with the enterprise social platform "Jive" [17] with email alerts, discussion groups, user notification, conversation tagging, and tracking of events to spur discussion and awareness of data issues.

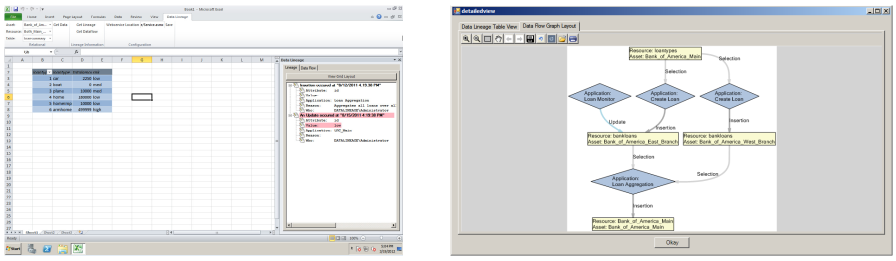
3 Architecture

In order to accommodate both new and existing assets within an enterprise environment, a multiple modal architecture is necessary which is minimally invasive with a mediation mode and a monitor mode. Existing assets and their owners may be unwilling or unable to modify aspects of their assets to utilize a data lineage system, as opposed to newly created systems in development, where modifications are easier to integrate into the main development trunk or a branch.

The overall architecture (see Fig. 2) consists of: a mediation mode interface for new assets; a monitoring mode interface that polls existing assets; a web service interface for integration of external tools (e.g. configuration tool, Microsoft Word, Microsoft Excel, etc.); and the core data lineage engine connected to the repository engine. The different operation modes, mediation and monitor, allow the system to provide different levels of granularity of captured lineage information in addition to the connectivity options for a given asset type (new or existing). The data lineage tool furthers our goal of providing lineage at the table or local level while for a global enhanced view we provide data flow discovery capabilities. For our proof of concept, we initially support data lineage for Microsoft SQL Server and My SQL data sources; however, in theory the data source types are not a limiting factor.

3.1 Mediation and Monitor Modalities

The mediation mode interface, primarily for newly developed assets, transfers calls through a web service and in-turn routes them to their intended destination. This allows the lineage tool to acquire the data as well as glean additional statistics of the operation such as alias information, source operating system information, invocation information, source machine name, IP information, etc. This mode allows for


detailedview
Data Lineage Table View
Data Flow Graph Layout
Resource: loantypes Asset: Bank_of_America_Main
Selection
Application: Loan Monitor
Application: Create Loan
Update
Insertion
Resource: bankloans Asset: Bank_of_America_East_Branch
Resource: bankloans Asset: Bank_of_America_West_Branch
Application: Loan Aggregation
Resource: Bank_of_America_Main Asset: Bank_of_America_Main
Okay

(a) (b)

Fig. 1. (a) The lineage tool integration with Microsoft Excel; (b) The value flow graph as seen from Excel

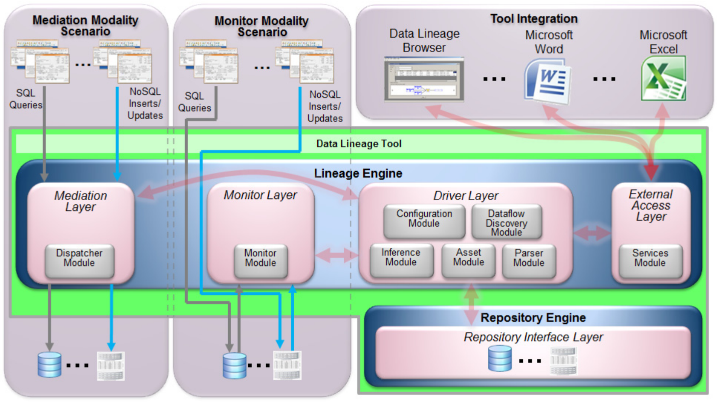


Fig. 2. System Architecture

enrichment of additional meta-data information for the association with an action for later retrieval and provides the context of data.

The monitor modality is another mechanism by which the tool tracks data lineage, with minimal impact on existing enterprise assets. The monitor modality operates by polling a data source, retrieving information concerning all queries executing on a resource, processes captured trace entries, and stores them for later retrieval. While in monitor mode, the polling rate for each resource can differ. Furthermore, the system only captures the traces with subsequent timestamps from the previous capture or polling event.

3.2 The Data Lineage Tool: The Lineage Engine, the Repository Engine, Data Lineage Tracking, and Data Flow Discovery

The Lineage Engine

The Lineage Engine consists of several modules: the mediation layer, the monitor layer, the driver layer, and the external access layer. The mediation and monitor layers are for the mediation and monitor modalities, respectively. The external access layer provides access and services for the integration of external tools and applications.

Contained within the driver layer are several sub modules: the configuration module, the asset module, the graph-flow discovery module, the inference module, and the parser module.

The configuration module provides a mechanism for registration of assets, resources, and applications. In addition, this module acts as a bridge for all settings between all other modules and layers.

The asset module keeps track of all assets and resources. It serves to maintain the connections for all assets and the resources that reside on them. The asset module is also in charge of maintaining and formatting the data lineage information. Not only does the asset module track registered assets and resource, but it also provides a mechanism for the tracking of the lineage utilizing the repository engine.

The graph-flow discovery module discovers data flows that exist within tracked data lineage. This module does this by clustering and discovering patterns in the SQL traces.

The inference module determines the reasoning of actions, why was an action taken and how. This module knows the "why" of information concerning asset, resource, and application actions based on information registered at configuration time of the data lineage tool. Asset, resource, and application information that the system captures from trace entries are utilized to search for previously provided knowledge and on the discovered asset, resource, and application combinations. In future work, it will also determine any manipulations that may have occurred to data as it passes through databases and applications.

The parser module integrates the known grammar sets used to interpret the SQL traces. Each resource type may have its own dialect and language; therefore, the parser module automatically determines which grammar to use and breaks down the SQL traces into actionable objects that have gone through interpretation. The interpretations are associated with the related actions of previous SQL traces within the assets module. After linking the actionable SQL traces to each other the result transfers to the repository engine for storage and later retrieval in data linage and data flow discovery requests.

The Repository Engine

The repository engine houses all of the data traces transformed into a data lineage friendly format. The repository engine is compatible with both SQL and NoSQL storage formats (MS SQL and Apache Cassandra respectively). The backend engine is swappable with either, although performance may vary depending on the configuration. SQL repository engine support is included for those environments that may be wary of untried technologies and may prefer technical support. Integration with a SQL database engine offers reliability and support; however, it may encounter some speed issues. In an SQL environment configuration for lineage tracking is more cumbersome and associations of related lineage traces can be complex. Apache Cassandra integration is included for its organization qualities. Apache Cassandra integration offers fewer schema constraints, which lend to easier and quicker setup. Additionally, Apache Cassandra provides super column families and keys that allow for increased ease of linking lineage traces. However, despite the existence of the Cassandra Query Language (CQL), one to one conversions of SQL to CQL statements may not exist and make it difficult to run some queries and performance can degrade in some cases. Cassandra also lacks services that the traditional data stores, such as auto incrementing, would otherwise make organization easier to prevent duplications. Integration of both standard a SQL and NoSQL demonstrates the flexibility of the data lineage tool.

Local Data Lineage Tracking

Our system is flexible in its configuration and utilizes a My SQL server, Microsoft SQL server, or Apache Cassandra system for storage. It can monitor a My SQL or Microsoft SQL server on the front end. To this end, capturing and discovering lineage is a two-step process beginning with capturing information at the local table level as described in this section.

The lineage information is stored using a backlog trace system as described in [18] with some modifications to suit our particular lineage needs. Our backlog trace algorithm gathers information and keeps track of the differences, groups of tuples that reference each other, SQL trace statistics, origination information, timestamp of lineage creation, timestamp of the SQL trace command, unique identifiers for each trace, a dirty flag or liveliness indicator, and finally the actual SQL command.

The following example illustrates a backlog trace implementation. The example is a simplification for ease of readability and excludes some captured meta-data and statistics. Table 1 illustrates a typical aggregate view of data in a database. Table 2 illustrates an aggregate view of data in the same typical database after a series of operations. Without data lineage tracking, it is unclear what exactly has occurred. At first glance, the aggregate shows two records underwent modification with an update (the user “Alice” has a different age of “25” and the user name “Carlos” appears to have been changed to “Dan”). Analyzing the change in the records, it becomes unclear as to who made modifications, the command order, what type of commands were issued, if an update was executed on the effected records, or were other operations issued to arrive at the same view.

The backlog trace compensates and captures the information that is lost in aggregation of trace entries. Furthermore, all commands are linked to related commands. An injection of the original data executes and inserts information so that subsequent manipulations on records within the database are linked. Table 3 shows the initial bootstrap of data as a series of inserts.

Following an update and insert command, “UPDATE users SET age=25 WHERE userid=1” and “INSERT INTO users VALUES (1,’Dan’, 30)”, the backlog trace is transformed. The data lineage tool will perform an insertion of any command that manipulates any existing data and link it to the last command that performed any manipulation. The tool performs a query on the backlog trace to find all records affected by the command, which is an update command in this case, and then links

Table 1. Table from a database (table "users"), an initial aggregate view

UserId	Name	Age
1	Alice	10
2	Bob	20
3	Carlos	30

Table 2. Final view of the "users" table after all issued commands. Deleted records are not shown. An aggregate view does show the loss of lineage information.

UserId	Name	Age
1	Alice	25
2	Bob	20
3	Dan	30

Table 3. List of initial SQL traces placed in the backlog trace creating initial aggregated view of the database table with a Apache Cassandra backend

Key	UID	SQL Command	App	UserId	Name	Age	Reference	Alive
1	1	INSERT INTO users VALUES (1, 'Alice', 10)	Create User	1	'Alice'	'10'	0	1
2	2	INSERT INTO users VALUES (1, 'Bob', 20)	Create User	2	'Bob'	'20'	0	1
3	3	INSERT INTO users VALUES (1, 'Carlos', 30)	Create User	3	'Carlos'	'30'	0	1

them. After finding all related commands, the new command is inserted and has its "Reference" column set to the unique id value of a previous command (as is the case with an "UPDATE"). In the case of an "INSERT" command, the tool inserts a new entry in the backlog trace table. When using Apache Cassandra, a key groups and relates commands.

Further illustrating the backlog trace example, Table 4 shows the effect of a "DELETE" and an "UPDATE" command. On a replay of events, this sequence of commands produce the result in Table 2 and demonstrates that without a backlog trace it is unclear as to the sequence of events that led to the final view. This is critical for producing the data lineage trace at the table level.

Global Data Flow Discovery

With tracking available at the level of a table, extraction at the global view must now occur. Our algorithm then focuses on the following problem: given a linked set of backlog trace entries from multiple databases, can it discover the data lineage flow for a set of SQL commands that contribute to an instance of existing values across multiple backlog trace entries. This problem is challenging for several reasons. Often the trace log entries for a database do not specify data lineage flows but rather only the SQL command, timestamp, and application issuing the particular command. Secondly, the trace entries may be noisy, or disjointed, due to missing entries or

Table 4. Example backlog trace after a delete and a final update is issued

Key	UID	SQL Command	App	UserId	Name	Age	Reference	Alive
1	1	INSERT INTO users VALUES (1, 'Alice', 10)	Create User	1	'Alice'	'10'	0	1
	4	UPDATE users SET Age=25 WHERE userid=1	Audit User	--	--	'25'	1	1
2	2	INSERT INTO users VALUES (1, 'Bob', 20)	Create User	2	'Bob'	'20'	0	1
3	3	INSERT INTO users VALUES (1, 'Carlos', 30)	Create User	3	'Carlos'	'30'	0	0
	6	DELETE FROM users WHERE id=3	Audit User	--	--	--	3	0
4	5	INSERT INTO users VALUES (1, 'Dan', 30)	Create User	4	'Dan'	'30'	0	1
	7	UPDATE users SET userid=3 WHERE userid=4	Audit Users	3	--	--	5	1

inadvertent modification from runtime issues. Finally, the algorithm should be able to scale itself with large trace log sizes.

We address the challenges by developing a statistic-based data mining approach that discovers the sequence of traces that an application typically performs (e.g. select, insert, update, delete, etc.). To discover the sequences for applications, the algorithm makes use of the following observation: an application executes a sequence of commands repeatedly with consistent time intervals between commands. Specifically, our algorithm performs the following steps:

1. For each application, the system gathers the trace log commands issued.
2. For each application, the system clusters similar trace log entries that relate according to the tables that are affected and attributes involved. For example, the statements "SELECT name FROM employee WHERE id=3" and "SELECT name FROM employee WHERE id=5" are clustered together because the command type, table name, and attribute name are identical.
3. For every pair of command type clusters, the algorithm measures the variability of the temporal difference between trace entries in the two clusters. Given two command clusters, c_1 and c_2, the algorithm identifies the mappings of traces between them. If a command-timestamp pair, (s_1, t_1), is mapped to another command-timestamp pair, (s_2, t_2), (where s_1 and s_2 are trace entry commands from c_1 and c_2, and t_1 and t_2 are timestamps for the corresponding trace entries) then the mappings should satisfy the following conditions:
   - There does not exist an s_1' in c_1 such that $| t_1' - t_2| < | t_1 - t_2|$, where s_1' is any command in the first cluster and t_1' is the timestamp for s_1'.

– There does not exist an s_2' in c_2 such that $| t_1' - t_2| < | t_1 - t_2|$, where s_2' is any command in the first cluster and t_2' is the timestamp for s_2'.

Once the mappings are identified, our algorithm calculates the entropy of the temporal differences (for example, t_1-t_2) to measure the consistency of the temporal differences. Specifically, the entropy is defined in equation (1) where $x_k = \frac{N_k}{N}$, N is total number of values in the list, and N_k is the total number of values that are k.

$$\sum_{k=0}^{n} p(x_k) \log p(x_k) \qquad (1)$$

If the temporal difference varies significantly, the entropy score will be high and the algorithm discards them. The algorithm proceeds to construct a cluster chain, C = {c_1, c_2, …, c_n}, of low entropies which corresponds to a sequence of trace entries that an application typically performs.

The algorithm is robust and can detect data flows in low signal-to-noise situations with the use of statistics and the frequency of information. The algorithm has a time complexity of $O(n * k)$ where "n" is the total number of traces entries and "k" is the average number of type of trace entries per application. Fig. 3 illustrates an example.

4 Experimental Setup and Evaluation of Results

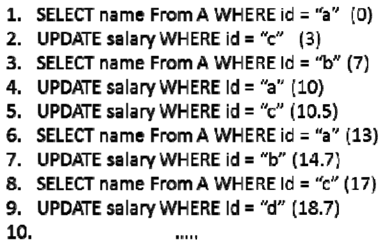
In this section, we present our evaluation of the data flow algorithm. Specifically, we first evaluate the system's accuracy by comparing the flow structure produced by the system to that of a gold standard. We also evaluate the system in the presence of noisy input and evaluate the scalability of the data flow algorithm under varying trace log entry sizes.

4.1 Experimental Setup

To evaluate the data lineage tool proof of concept, we generated synthetic data with a simulator that models heterogeneous environments. Our simulator defines the number of applications, databases, tables, and the attributes.

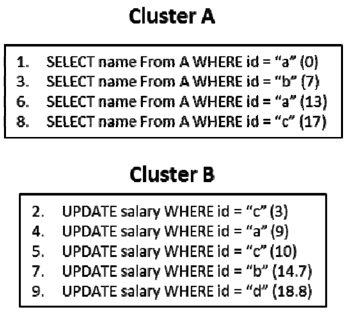
The simulator defines command sequence templates that an application should perform repeatedly. A template S has the form of $((c_1,0), (c_2, t_2) \ldots, (c_n,t_n))$ where c_i defines an abstract structure of a command such as "SELECT name FROM table$_1$ WHERE id=<VALUE>" where t_i represents the mean temporal difference between c_{i-1} and c_i. A timestamp for the first command of a template is chosen at random. We also define a noise probability parameter p, which indicates that a command deletion occurs or that an additional command insertion occurs with the same probability at each timestamp.

The algorithm creates two outputs: flow structure, and command sequence. The flow structure defines which applications issue which SQL statements to particular database tables denoted by tuple (A,S,T) where A is the application name, S is the



```
1.  SELECT name From A WHERE id = "a"  (0)
2.  UPDATE salary WHERE id = "c"  (3)
3.  SELECT name From A WHERE Id = "b" (7)
4.  UPDATE salary WHERE Id = "a" (10)
5.  UPDATE salary WHERE id = "c" (10.5)
6.  SELECT name From A WHERE id = "a" (13)
7.  UPDATE salary WHERE id = "b" (14.7)
8.  SELECT name From A WHERE Id = "c" (17)
9.  UPDATE salary WHERE Id = "d" (18.7)
10.                   .....
```

(a)



Cluster A

```
1.  SELECT name From A WHERE id = "a" (0)
3.  SELECT name From A WHERE id = "b" (7)
6.  SELECT name From A WHERE id = "a" (13)
8.  SELECT name From A WHERE id = "c" (17)
```

Cluster B

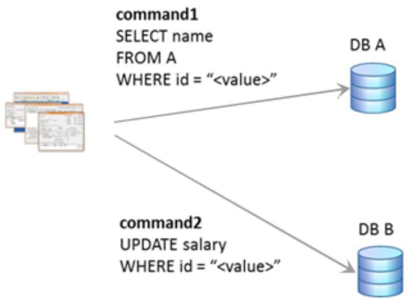
```
2.  UPDATE salary WHERE id = "c" (3)
4.  UPDATE salary WHERE id = "a" (9)
5.  UPDATE salary WHERE id = "c" (10)
7.  UPDATE salary WHERE id = "b" (14.7)
9.  UPDATE salary WHERE id = "d" (18.8)
```

(b)

Mappings		Temporal Difference
1	2	3
3	3	3
6	7	1.7
8	9	1.7

Entropy: -1/2log(1/2) – 1/2log(1/2) = 1

(c)



(d)

Fig. 3. (a) Step1: the data lineage system returns SQL commands for an application. The simplified forms of the timestamps are shown in the parentheses; (b) Step2: The SQL commands are grouped into two clusters; (c) Step3: The mappings from Cluster A to Cluster B (the first two columns) and their temporal differences (third column) are produced. Then, the entropy is calculated for the temporal differences; (d) Output: If the entropy score is less than a specified threshold, the lineage flow is created, in which command 1 and 2 correspond to Cluster A and B respectively and command 1 occurs before command 2.

SQL statement, and *T* is the table name within a database. The flow structure output displays the overall relationship between applications and databases. The command sequence is the pairing of trace entry sequences (C_1, C_2) where C_i represents an entry in the flow structure output. The pairing (C_1 ,C_2) means that C_1 and C_2 are in the same sequence and that C_1 is executed earlier than C_2. For both the flow structure and command sequences we measure the performance with precision (see equation (2)), recall (see equation (3)), and the F1-score (see equation (4)). In equations (2),(3), and (4) N_{OG} is the number of trace log entries that appear in both the system output and in the templates, N_O is the number of command entries in the system output, N_G is the number of command entries in the templates used by the simulator.

$$Precision = \frac{N_{OG}}{N_O} \quad (2)$$

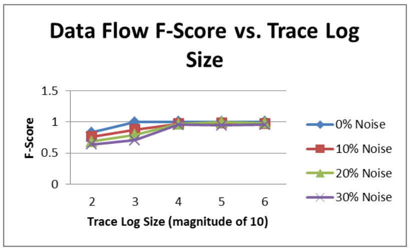
$$Recall = \frac{N_{OG}}{N_G} \quad (3)$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4)$$

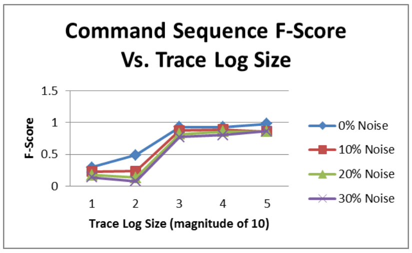
For each noise probability measure, we repeat the evaluation 10 times and report the average of the 10 repetitions.

4.2 Evaluation: Performance

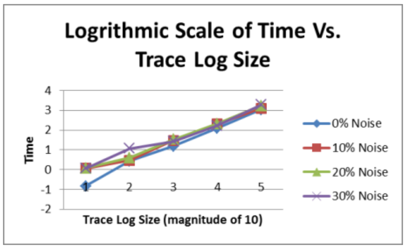
The F1-Score performance evaluation measures the accuracy of our system given a trace log with increasing sizes of 10^2, 10^3, 10^4, 10^5, and 10^6. Our results reflect measurements for robustness at increasing noise levels of 0%, 10%, 20%, and 30%. At the smallest trace log size, the data flow F1-Score values range 0.63 and 0.83, while at larger trace log sizes it ranges between 0.95 and 1. The command sequence F1-Scores trend similarly (see Fig. 4, graphs a and b, for systems accuracy as the trace log size increases). This confirms our hypothesis that the system can make accurate decisions with sufficient and logs of increasing size. Furthermore, the figures demonstrate the robustness to noise due to the statistical nature of our algorithm.



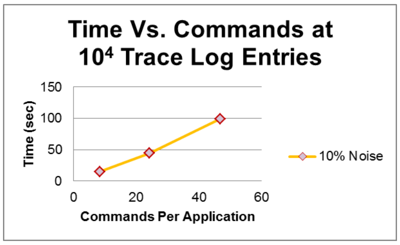
(a)



(b)



(c)



(d)

Fig. 4. (a) F1-Score for data flow; (b) F1-Score for command sequences; (c) Scalability of the Data Lineage Tool as dataset size increases; (d) Linear scaling as command sequence complexity increases

4.3 Evaluation: Scalability

The evaluation of scalability takes into consideration the total size of a given trace log as well as the average number of commands executed per application.

Fig. 4 (graphs c and d) show the linear runtime of our data flow discovery algorithm. As we increase the parameters for the trace log size and noise levels, the runtime increases as expected in a linear fashion. This confirms our hypothesis that our algorithm can reliably handle increasing sizes of trace logs.

5 Conclusion

Our motivation was to create a proof of concept tool that could track data lineage and generate data flows effectively and efficiently on a heterogeneous environment. What is of key importance is the ability to scale and deliver in a timely manner as the dataset increases. We have shown that our data flow detection improves as the number of records increase and is very robust to noise. This means that data traces may be lost up to 30% of the original data and a reasonable data flow generation still occurs. While the number of commands for several applications can affect accuracy, increasing the dataset size can mitigate this effect due to the frequency analysis of the information. Our results show that the execution time of our tool scales linearly to the number of records and to the number of commands per application.

We are able to capture high-level patterns by clustering trace log entries and discovering temporal relationships between clusters. As a result, our tool is able to accurately discover flow and sequence structures, is robust to noise, and scales to the size of data.

Acknowledgments. We would like to thank John Akred for his encouragement and continuing support of this work. We would also like to thank Jim Solderitsch for his invaluable input and guidance of our research as it developed. In addition, we would like to thank Harsha Srivatsa for providing subject matter expertise and advice during the early stages of this research. Lastly, we would like to thank the reviewers for their feedback and comments on improving this paper.

References

1. Basel Committee on Banking Supervision, `http://www.bis.org/bcbs/index.html`
2. Sarbanes Oxley Act of 2002, `http://www.sec.gov/about/laws/soa2002.pdf`
3. The Federal Reserve Board, `http://www.federalreserve.gov`
4. United States Securities and Exchange Commission, `http://www.sec.gov`
5. Cui, Y., Widom, J., Wiener, J.: Tracing the Lineage of View Data in a Warehouse Environment. ACM Transactions on Database Systems 25 (1997)
6. Cui, Y., Widom, J.: Lineage Tracing for General Data Warehouse Transformations. The VLDB Journal 12, 41–58 (2003)

7. Gehani, A., Kim, M.: Mendel: Efficiently Verifying the Lineage of Data Modified in Multiple Trust Domains. In: 19th ACM International Symposium on High-Performance Parallel and Distributed Computing, pp. 227–239 (2010)
8. Agrawal, P., Benjelloun, O., Sarma, A., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A System for Data, Uncertainty, and Lineage. In: 32nd International Conference on Very Large Databases, pp. 1151–1154 (2006)
9. Hagerty, J., Sallam, R., Richardson, J.: Magic Quadrant for Business Intelligence Platforms. Gartner for Business Leaders (February 6, 2012)
10. IBM, `http://www.ibm.com`
11. Oracle, `http://www.oracle.com`
12. Microsoft SQL Server, `http://www.microsoft.com/sqlserver`
13. My SQL, `http://www.mysql.com`
14. Apache Cassandra, `http://cassandra.apache.org`
15. Microsoft Excel, `http://office.microsoft.com/en-us/excel`
16. Yeh, P., Puri, C., Wagman, M., Easo, A.: Accelerating the Discovery of Data Quality Rules: A Case Study. In: 23rd Conference on Innovative Aplications of Artifical Intelligence (2011)
17. Jive Software, `http://www.jivesoftware.com`
18. Fabbri, D., LeFevre, K., Zhu, Q.: Policy Replay: Misconfiguration-Response Queries for Data Breach Reporting. In: 36th International Conference of Verly Large Data Bases, vol. 3, pp. 36–47 (2010)

Evaluating the Feasibility Issues of Data Confidentiality Solutions from a Data Warehousing Perspective

Ricardo Jorge Santos[1], Jorge Bernardino[1,2], and Marco Vieira[1]

[1] CISUC – Centre of Informatics and Systems of the University of Coimbra
University of Coimbra, 3030-290 Coimbra, Portugal
[2] ISEC – Superior Institute of Engineering of Coimbra
Polytechnic Institute of Coimbra, 3030-190 Coimbra, Portugal
lionsoftware.ricardo@gmail.com,
jorge@isec.pt, mvieira@dei.uc.pt

Abstract. Data Warehouses (DWs) are the core of enterprise sensitive data, which makes protecting confidentiality in DWs a critical task. Published research and best practice guides state that encryption is the best way to achieve this and maintain high performance. However, although encryption algorithms strongly fulfill their security purpose, we demonstrate that they introduce massive storage space and response time overheads, which mostly result in unacceptable security-performance tradeoffs, compromising their feasibility in DW environments. In this paper, we enumerate state-of-the-art data masking and encryption solutions and discuss the issues involving their use from a data warehousing perspective. Experimental evaluations using the TPC-H decision support benchmark and a real-world sales DW support our remarks, implemented in Oracle 11g and Microsoft SQL Server 2008. We conclude that the development of alternate solutions specifically tailored for DWs that are able to balance security with performance still remains a challenge and an open research issue.

Keywords: Data masking, Encryption, Data security, Confidentiality, Database Performance, Data Warehousing.

1 Introduction

Data Warehouses (DWs) store the secrets of the business and are used to produce business knowledge, which makes them a major target for attackers [4, 7, 20]. As the number and complexity of attacks increase, efficiently securing the confidentiality of DWs is critical [15, 18, 23]. To accomplish this, data masking and encryption solutions are widely used. Although data masking routines are simpler than encryption routines, they provide lower security strength. Moreover, data masking routines provided by most commercial tools typically change data in an irreversible manner, *i.e.*, after masking data it not possible to subsequently retrieve the original true values, making them useless for real live DW databases. This has made masking solutions the main choice for protecting published data or production data, instead of real-live databases [5, 8, 11, 12, 16, 20, 21]. Published research and best practice guides have stated that encryption is the best method to protect sensitive data at the database level while maintaining high database performance [3, 7, 8, 11, 12, 13, 14, 18, 23].

Despite their security strength, encryption techniques introduce performance key costs from a data warehousing point of view:

- Large processing time/resources for encrypting sensitive data, given routine or hardware access in very large databases such as those in DWs;
- Extra storage space of encrypted data. Since DWs usually have many millions or billions of rows, even a small modification of any column size to accommodate encrypted output introduces large storage space overheads;
- Overhead query response time and allocated resources for decrypting data to process queries. Given the huge amount of data typically accessed by DW queries, this is probably the most significant drawback in DWs [3].

The two main features that differentiate one confidentiality solution from the other are its security strength and its execution speed and efficiency. Given the specificity of DW environments, we believe there are specific performance issues to evaluate and discuss, regarding the use of encryption solutions. This is the foundation of this research work. We present the state-of-the-art solutions for protecting stored data confidentiality and evaluate their feasibility from a data warehousing perspective. It is not within the scope of this paper to discuss the scientific merit or soundness of the security strength of each technique, but rather to evaluate their impact on database performance and applicability in data warehousing environments.

Thus, in this paper we analyze and discuss the technical issues involving the implementation of the available state-of-the-art data confidentiality solutions, and use storage space and query response time as measures for evaluating performance in both loading and querying DW data. To support our remarks and claims, we include experimental evaluations using the TPC-H decision support benchmark [17] and a real-world sales DW, implemented in two leading commercial DataBase Management Systems (DBMS), such as Oracle 11g and Microsoft SQL Server 2008.

The main contributions of our work are as follows:

- We enumerate and describe the current state-of-the-art techniques for protecting stored data and discuss their application from a data warehousing perspective;
- We present the results of several experimental evaluations using two leading commercial DBMS and one leading open-source DBMS on a well-known benchmark (TPC-H) and a real-world DW, analyzing the impact in database performance due to using encryption techniques for protecting data confidentiality;
- The issues discussed and the results from our experimental evaluations allow us to state that currently available encryption solutions are not suitable for most DWs. Our work shows that the development solutions specifically tailored for DWs that are able to present better tradeoffs in balancing security strength with database performance remains a challenge and a relevant research issue.

The remainder of the paper is structured as follows. In section 2, we describe state-of-the-art data masking and encryption solutions for databases and discuss their issues from the DW perspective. Section 3 provides experimental evaluations of those solutions using the well-known TPC-H decision support benchmark and a real-world sales

DW implemented in two leading commercial DBMS. In section 4 we point out research challenges and opportunities regarding specific confidentiality solutions for DWs, from the lessons learned. Finally, section 5 presents our conclusions.

2 State-of-the-Art Data Confidentiality

2.1 Data Masking Solutions

An extensive survey on data masking is given in [16]. Many organizations have strived to solve confidentiality issues with hand-crafted solutions within the enterprise to solve the problem of sharing sensitive information. The most common solution is probably to use scripts with triggers in order to mask and unmask each value, or to embed the masking/unmasking logic within the user applications themselves.

Many commercial data masking packages have also been developed, such as the Oracle Data Masking (ODM) pack [11, 12], protecting data by replacing real values with realist-looking data with the same type and characteristics as the original data. Once applied, the ODM masking process is irreversible. Oracle states ODM is to be used as a fast and easy way to generate production databases for supporting outsourcing and software application development. It can also be used to mask Microsoft SQL Server and DB2 databases for the same purpose. ODM requires new data to be loaded into the database first, and only applies the masking procedures afterwards. It is not possible to load previously masked data; masking in ODM is an *a posteriori* process. Most commercial data masking solutions work in a similar fashion as ODM.

Recently, research has proposed non-deterministic methods for masking data, such as **perturbation techniques** [4, 14, 19]. The work in [4] proposes a solution based on data perturbation techniques and explains data reconstruction for responding to queries, in a DW environment. Recent similar work proposed data anonymization solutions relying on perturbation or differential techniques [14] and [19].

2.2 Data Encryption Algorithms

Typical encryption algorithms include executing bit shifting and exclusive Or (XOR) operations within a predefined number of rounds. These operations rely on a key, which influences the "data mix" output of each round. There are mainly two types of encryption techniques: **Block Ciphers** and **Stream Ciphers**.

A block cipher is a type of symmetric-key encryption algorithm that transforms a fixed-length block of *plaintext* (unencrypted text) data into a block of *ciphertext* (encrypted text) data of the same length, under the action of a user-provided secret key. Decryption is performed by applying the reverse transformation to the ciphertext block using the same secret key. Stream ciphers take a string (the encryption key) and deterministically generate a set of random-seeming text (called *keystream*) from that key. That keystream is then XORed against the message to encipher. To decipher the text, the recipient hands the same key to the stream cipher to produce an identical keystream and XORs it with the ciphertext, thus retrieving the original message.

The **Data Encryption Standard (DES)** became the first encryption standard [6]. DES is a 64 bit block cipher, which means that data is encrypted and decrypted in 64 bit chunks, and uses a 56 bit encryption key. As an enhancement of DES, the **Triple**

DES (3DES) standard was proposed [1]. The 3DES encryption algorithm is similar to the original DES, but it is applied three times to increase the encryption level, using three different 56 bit keys. Thus, the effective key length is 168 bits. Since the algorithm increases the number of cryptographic operations it needs to execute, it is a well known fact that the 3DES algorithm is one of the slowest block cipher methods.

The **Advanced Encryption Standard (AES)** is a symmetric block cipher algorithm [2]. These algorithms are the latest generation of block ciphers, and have a significant increase in the block size, 128 bits. AES provides three approved key lengths: 128, 192 and 256 bits. AES is considered fast and able to provide stronger encryption than other well-known encryption algorithms such as DES [10, 25]. Brute force attack (where the attacker tries all the possible key combinations to unlock the encryption) is the only known effective attack known against it.

In [3] an Order Preserving Encryption Scheme (**OPES**) for numeric data is proposed, flattening and transforming plaintext distributions onto target distributions defined from value-based buckets. This solution allows any comparison operation to be directly applied on encrypted data, such as equality and range queries, as well as SUM, AVG, MAX, MIN and COUNT queries. A lightweight database **encryption scheme for column-oriented DBMS** is proposed in [7].

The **Blowfish encryption algorithm** [24] is one of the most common public domain encryption algorithms. It is a 64 bit block cipher, allowing a variable key length. Each round of the algorithm consists of a key-dependent permutation and a key-and-data-dependent substitution. All operations are based on XORs and additions on 32-bit words. The key has a variable length (maximum of 448 bits). Though it suffers from weak keys problem, no attack is known to be successful against it [10].

More recently, the **Snuffle 2005** encryption algorithm (also known as **Salsa20**) was proposed [22]. It is a stream cipher based on a hash function with a long chain of simple operations (32-bit additions, 32-bit XORs, and constant distance 32-bit rotations), instead of a short chain of more complex operations (typical in standard encryption algorithms). Salsa20 produces a 64-bit block given a key, nonce and block counter. Salsa20 simply works by running the hash function in counter mode, generating the keystream by hashing the key with a message based nonce and sequential integers (1, 2, 3, etc) appended. This solution is relatively simple when compared with other standard encryption algorithms and has been recognized by the cryptology research community as an interesting alternative in certain contexts.

Since we focus on discussing if current data encryption algorithms are too slow or not for DWs, we are not interested in discussing the security details of each algorithm, but rather in pointing out their generic guidelines and how their performance is affected. In cryptography, an S-box (Substitution-box) is a basic component of symmetric key algorithms. They are typically used to obscure the relationship between the keys and the generated ciphertext. In general, an S-box takes a number of input bits, m, and transforms them into a number of output bits, n; an $m \times n$ S-box can be implemented as a lookup table with $2m$ words of n bits each. In many cases, the S-boxes are carefully chosen to resist cryptanalysis. Fixed tables are normally used, as in the Data Encryption Standard (DES) [6], but in some ciphers the tables are generated dynamically from the key; *e.g.* the Blowfish encryption algorithm [18].

The argument in favor of using complicated operations such as S-boxes is that a single table lookup can mangle its input quite thoroughly – more thoroughly than a chain of simple integer operations – in fewer rounds. This provides a large amount of mixing at reasonable speed on many CPUs, reaching many desired security levels more quickly than simple operations. The counterargument is that potential speedup is fairly small and is accompanied by huge slowdowns on other CPUs. On the other hand, simple operations such as bit additions and XORs are consistently fast, independently from the CPU. It is also not obvious that a series of S-box lookups (even with rather large S-boxes, as in AES, increasing L1 cache pressure on large CPUs and forcing different implementation techniques for small CPUs) is generally faster than a comparably complex series of simpler integer operations.

Table 1 shows the number of rounds for achieving minimum and recommended security strength, respectively, along with block size and encryption key lengths, for Salsa20, 3DES and AES. The performance of Salsa20 in the ENCRYPT's test framework reports the speeds (in CPU cycles per encrypted byte) for encrypting a 576-byte packet (or a long stream) on several CPUs [25] and are shown in Table 2. The values for the AES encryption algorithm are from [26].

Table 1. Encryption algorithm variables w/ performance impact

	Salsa20	3DES	AES
Recommended nr. of rounds	20	16	14
Minimum nr. of rounds	8	12	10
Block size	512 bits	64 bits	128 bits
Encryption key length	128 or 256 bits	168 bits	128 or 256 bits

Table 2. Encryption algorithms CPU Cycles p/ Encrypted Byte

	CPU Cycles p/Encrypted Byte	
CPU/Algorithm	**Salsa20**	**AES128**
AMD64 3GHz Intel Xeon 5160 (6f6)	4.3	9.2
Intel Core 2Duo 2.1GHz (6f6)	4.3	9.2
AMD64 3GHz Intel Pentium D (f64)	11.7	16.2
Intel Pentium 4 3GHz (f41)	13.4	19.8

Other encryption solutions, such as [3], distribute data in well-defined groups to allow direct operations on encrypted data. However, the impact in performance produced by these solutions, in response time and storage space overhead, depends on the skew in the target distributions, which can be a very serious problem in DWs. There is no easy way around this. The proposal from [23] also suffers from the same problem. The lightweight encryption in column-oriented DBMS proposed in [7] aims on providing low decryption overheads. However, their experiments show at least 50% of response time overhead to retrieve the encrypted tuples, which is still extremely high for many DW scenarios, such as long running queries.

Analyzing the features of the referred encryption solutions that influence performance (and security tradeoff), we have found the following conclusions:

- Specific DW encryption solutions still show large performance overheads;
- The type and number of operations for producing the "data mix" output in each round of the algorithm, the length of the used encryption keys, the size of the input

and output blocks, and the number of rounds to execute, are all variables that affect both security and performance;

- Typically, a secure encryption algorithm will execute between 8 and 20 rounds against 64, 128 bit (or more) sized blocks, using a 128 or 256 bit key;
- Encryption algorithms which make use of chains of simple operations such as bit additions and XORs scale better and have reduced CPU dependency than solutions that make use of more complex operations such as S-box lookups;
- Salsa20 seems to provide consistent speed in a wide variety of applications across a wide variety of platforms. It is faster and simpler than other complex approaches such as the standard algorithms 3DES and AES, while granting significant security strength. However, most commercial vendors just include AES and 3DES routines. The AES became a standard only after a five-year long standardization process that included extensive benchmarking on a variety of platforms ranging from smart cards to high end parallel machines. Thus, the adoption of encryption standards is probably only due to legal impositions and public reliability issues.

2.3 Data Masking/Encryption Architectures

There are mainly two types of architecture for data masking and encryption at the database level: 1) masking/unmasking and encryption/decryption is executed by the DBMS server itself directly on the database; or 2) all masking/unmasking and encryption/decryption is executed by a third party, typically an application or service acting as a middleware tier between user applications and the encrypted/masked database. The first type of architecture is typically used with built-in packages provided by DBMS vendors. These routines run in the DBMS kernel and are optimized to work against their data structures and across a large diversity of platforms.

Major DBMS such as Microsoft SQL Server and Oracle provide standard encryption routines. Oracle has developed TDE (Transparent Data Encryption) [11, 13] incorporating both AES and 3DES, providing column and tablespace encryption. These routines can be used transparently without requiring user application source code modifications. As Oracle, Microsoft SQL Server also provides column and datafile 3DES and AES encryption routines.

When using Oracle TDE tablespace encryption, all data in the tablespace's physical datafiles is encrypted and nearly no storage space overhead is generated. When using column encryption, a storage space overhead between 1 and 52 bytes per encrypted value is added. The generation of independently encrypted values for each column is done by using an optional feature (SALT) in the encryption, which implies adding 16 bytes of the storage space per encrypted column to each row. If NO SALT is used, those extra 16 bytes are saved, but all encrypted values in the column rely on one key only in the encryption algorithm. Tablespace encryption uses only the database master key and the tablespace's encryption key, which makes its security level lower than column encryption. Oracle recommends the use of tablespace encryption when there is no way of determining which columns are sensitive and which are not, or when the majority of the data in the tablespace is sensitive [8]. They state that column encryption should be preferred when a small number of well defined columns are sensitive. This last scenario is typical in data warehousing environments, which makes column encryption the recommended solution according to Oracle. However, as referred before, when using TDE column encryption in DWs the storage overhead will be very significant. On the other hand, since DWs store business secrets, we can assume that most of its data is sensitive. In this sense, we may also state that TDE tablespace encryption should also be highly considered.

In most enterprises, data used for analyzing business performance is mostly stored in numerical attributes, called facts [9]. Fact tables typically take up 90% or more of the DW's total storage space [9]. Standard encryption algorithms were designed for general-purpose data. Thus, they were designed for encrypting blocks of text, *i.e.*, sets of character-values by default. This has led DBMS to implement encryption routines that just output textual or binary attributes. Since most DW columns store numerical values, using encryption means they need to be converted to textual format. When the values are decrypted for query processing, they need to be converted back into numerical format in order to process sums, averages, etc. Since most decision support queries process mathematical functions and calculus against numerical attributes, conversion operations are a significant and potentially critical drawback, adding computational overheads with considerable performance impact.

Topologies involving middleware solutions such as [15] typically request the encrypted data from the database *a priori* and execute the decrypting actions themselves locally. The proposal in [15] aims to ensure efficient query execution over encrypted databases, by evaluating most queries at the application server and retrieving only the necessary records from the database server. Only one query (Q6) of the TPC-H benchmark is used in their experimental evaluation, against a very small data subset (ranging from 10MB to 50MB, where query execution time rises up to 5 times for the last). This is not a realistic dataset for DWs. In a DW environment, previously transporting all the required data from the database to the middleware is unreasonable, since the amount of data accessed for processing decision support queries is typically much larger than a few tens of MB. This would strangle the network due to bandwidth consumption of data roundtrips between middleware and database, jeopardizing data throughput and consequently, response time. Thus, all encrypted data should be processed at the DBMS itself, eliminating network overhead from the critical path.

In this sense, we have found the following conclusions:

- All major DBMS provide encryption to be used transparently by user applications;
- When using tablespace encryption, the requested data is decrypted loaded into RAM memory (in the database cache) as clear text, while column encryption does not and is thus more secure;
- Tablespace encryption does not create significant storage space overhead, while column encryption does;
- Despite well-known pros and cons, the best choice between tablespace encryption and column encryption isn't obvious;
- Leading DBMS use standard encryption algorithms AES and 3DES, producing alphanumeric or binary values as a result of the encryption process, even for numerical-typed attributes;
- In DWs, transporting encrypted data to third party decrypting agents would create unbearable communication bandwidth consumption and compromise throughput.

3 Experimental Evaluation

We implemented the TPC-H decision support benchmark (TPC-H) [17] with 1GB and 10GB scale sizes, and a real-world sales DW with 2GB of storage size. The sales DW has a star schema [9] with four dimension tables and one fact table (Sales). Its dimensional features are shown in Table 3. Each DBMS was installed on separate machines, Pentium 2.8GHz CPU with a 1.5TB SATA hard disk and 2GB RAM, with 512MB of RAM devoted to the database memory cache (SGA). The Oracle machine ran Windows XP Professional, SQL Server ran Microsoft Windows Server 2003. The TPC-H database schema has one fact table (*LineItem*) and seven dimension tables, where four columns of *LineItem* were chosen for encryption (*L_Quantity*, *L_ExtendedPrice*, *L_Tax* and *L_Discount*), given they are the fact columns used in the benchmarks queries to analyze the business. In Sales DW, five numerical fact columns were encrypted (*S_ShipToCost*, *S_Tax*, *S_Quantity*, *S_Profit*, and *S_SalesAmount*), for the same reasons. In our tests, we used the following encryption algorithms: AES with 128 bit and 256 bit keys, and 3DES168 (which uses triple DES with a 168 bit key), provided by each DBMS, in both tablespace (Tab) and column (Col) encryption modes [8, 11, 13]. Salsa20 [22] and OPES [3] were implemented in C++ and also tested.

Table 3. Dimensional features of the Sales Data Warehouse

	Times	Customers	Products	Promotions	Sales
Number of Rows	8 760	250 000	50 000	89 812	31 536 000
Storage Size	0,12 MB	90 MB	7 MB	10 MB	1 927 MB

3.1 Fact Table Loading Time

Figure 1 shows the loading time overhead percentages concerning of the fact table in the TPC-H 1GB and Sales DW for all the tested scenarios. The results in the TPC-H 10GB scenarios are similar to those of the TPC-H 1GB, with absolute values approximately proportional (10 times bigger), and due to lack of space are not included.

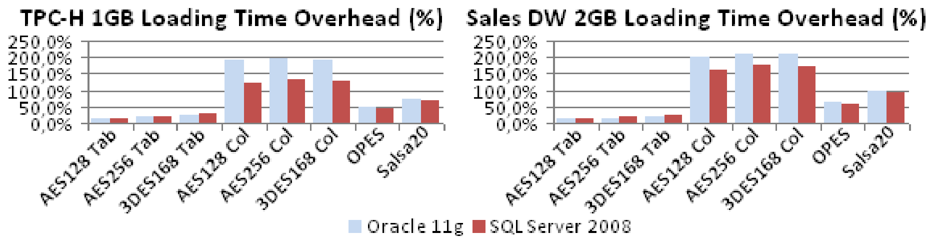


Fig. 1. TPC-H 1GB and Sales DW loading time overheads per DBMS

It can be seen that loading time overheads range from 14,8% (more 46 seconds) to 191,6% (more 594 seconds) in Oracle 11g, from 16,5% (more 35 seconds) to 129,2% (more 274 seconds) in SQL Server 2008, for loading the TPC-H 1GB LineItem fact table (approximately 800MB of original data). The results show that most overheads are indeed very considerable, although tablespace encryption present better results than column encryption, as it would be expected. AES also has better results than 3DES, since it has been proven a faster algorithm. OPES shows overheads of 43,6% and 48,7%, while Salsa20 of 70,4% and 73,3%. OPES and Salsa20 show better results than column encryption, but worse than tablespace encryption.

Loading time overheads for the Sales DW fact table (approximately 1,9GB of data) range from 13,3% (more 159 seconds) to 209,5% (more 2512 seconds) in Oracle 11g, from 15,4% (more 192 seconds) to 171,1% (more 2139 seconds) in SQL Server 2008. As in TPC-H 1GB, the results show that most overheads are indeed very considerable, tablespace encryption presents the best results and column encryption the worst, AES also has better results than 3DES, and OPES better than Salsa20.

3.2 Fact Table Storage Space

Figure 2 shows storage space overhead percentages concerning the fact table in the TPC-H 1GB and Sales DW. As with loading time results, storage space results in the TPC-H 10GB scenarios are similar those of TPC-H 1GB, with absolute values approximately proportional (10 times bigger), and due to lack of space are not included.

Since tablespace encryption affects the datafiles' contents as a whole, it is known that they do not increase storage space (this is because they use a sole key for encrypting data in the entire datafile as a single entity). OPES also minimally increases storage space, with almost no overhead (ranging from 1,9% to 3,7%), while Salsa20 has a space storage overhead of 26,4% to 94,1%. For column encryption in the TPC-H 1GB fact table, storage space overhead is 153,9% (more 1188MB) and 103,6% (more 800MB) in Oracle 11g, 94,8% (more 1173MB) and 76,3% (more 994MB) in SQL Server 2008. Column encryption storage space overhead for the Sales DW fact table is 461,5% (more 7680MB) and 307,7% (more 5120MB) in Oracle 11g, 591,3% (more 11424MB) and 389,9% (more 7532MB) in SQL Server 2008. The results show that storage space overheads for column encryption are very considerable. As expected, 3DES presents better results than AES because it outputs a smaller block size.

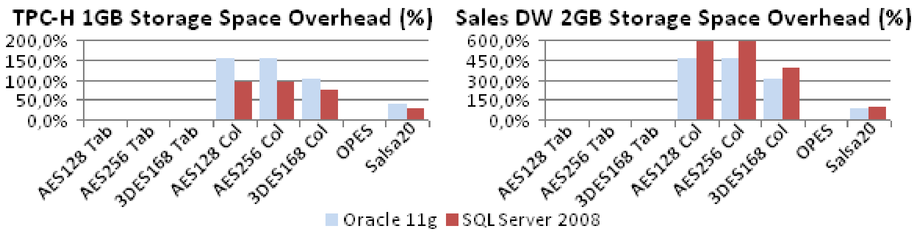


Fig. 2. TPC-H 1GB and Sales DW storage space overheads per DBMS

3.3 OLAP Query Execution

For TPC-H, the test workload was benchmark's queries 1, 3, 6, 7, 8, 10, 12, 14, 15, 17, 19, and 20, all queries accessing the *LineItem* fact table. For the Sales DW, the workload was a set of 29 queries, all processing facts in the *Sales* fact table, representing typical decision support queries such as customer product and promotion sales daily, monthly and annually values, including actions such as selection, joining, aggregates, and ordering. All results shown in this section are the average response time (in seconds) values obtained from six executions (with standard deviations between [0.52, 54.65] and [0.64, 70.10] for 1GB and 10GB TPC-H, respectively, and [0.32, 33.11] for the Sales DW, for individual query response times).

Figure 3 shows total query workload response time overhead percentages in the TPC-H 1GB and the Sales DW. The TPC-H 10GB results are similar to those of the TPC-H 1GB, with absolute values approximately proportional (10 times bigger), and due to lack of space are not included. It can be seen that for TPC-H 1GB the response time overheads range from 28,3% (more 177 seconds) to 203,0% (more 1271 seconds) in Oracle 11g, from 35,2% (more 204 seconds) to 195,2% (more 1132 seconds) in SQL Server 2008. For the Sales DW, response time overheads range from 79,5% (more 497 seconds) to 810,5% (more 5069 seconds) in Oracle 11g, from 82,8% (more 518 seconds) to 758,9% (more 4746 seconds) in SQL Server 2008.

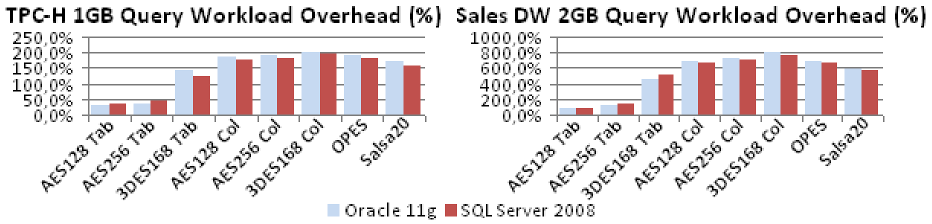


Fig. 3. TPC-H 1GB and Sales DW query workload response time overheads per DBMS

The results show most overheads are extremely high, and that tablespace encryption presents much better performance than column encryption and OPES and Salsa20. AES also has better results than 3DES, since it is a faster algorithm.

3.4 Discussion and Conclusions

Regarding the TPC-H 1GB results, notice that for TPC-H 10GB (which is ten times bigger), since all overheads are approximately proportional, absolute values of storage space, loading and response time overheads are nearly ten times bigger. This means that TPC-H 10GB has approximately 8GB to 12GB of increased storage space, for column encryption scenarios; tablespace encryption does not introduce extra storage space. In what concerns loading time, TPC-H 10GB can expect an extra 6 to 13 minutes to load the data in tablespace encryption, and 44 to 99 minutes for the column encryption scenarios. In the TPC-H 10GB column encryption, OPES and Salsa20 setups, query workload response time rises up from 2 to 3 hours. Given that 10GB is actually a small size for a DW database, it is easy to conjecture that the introduced overheads due to encryption in DWs are extremely significant and may in fact be unacceptable. Although Oracle argues that TDE will only increase response time an average of 5% to 10% [13], this has shown not to be true. The results show that response time overhead is, on average, many orders of magnitude higher. The same occurs in SQL Server. Conclusively, the findings were the following:

- Using encryption does in fact introduce huge storage space, data loading time and query response time overheads;
- Given that decision support environments typically execute long running queries (*i.e.*, queries that run for many minutes up to hours), those response time overheads represent high absolute values that can easily make query responses overdue and jeopardize the usefulness of the DW itself;
- Storage size and data loading time overheads are also very large, mainly in column encryption, with implications in database availability and storage management;
- Although security best practice recommends using column encryption in DW environments, tablespace encryption presents better performance results.

4 Research Challenges and Opportunities

In a traditional DW, data is static, *i.e.*, there is no data loading when the databases are available to its end users. In these environments, the main performance issue is not encryption, but decryption overhead for querying. Since data loading occurs in well-defined time windows in which the database is offline, there is no impact in user query response time; it only affects DW maintenance time. Nevertheless, this static data state paradigm has been changing, with the increasing implementation of real-time DW solutions. Thus, given the size of DWs and the amount of data typically processed by decision support queries, the overheads introduced by both encryption and decryption need to be dealt with, for the sake of their feasibility. The development of future solutions must consider the performance of both encryption/masking and decryption/unmasking as critical.

To improve CPU performance and scalability, using long chains of simple operations instead of short chains of complex operations may allow developing faster solutions while being able to maintain significant security strength, as argued in Salsa20. The basic argument for increasing the block size of the standard 16 bytes to a higher size of 256 bytes, for example, is that we do not need as many cipher rounds to achieve the same security level. Using a larger block size should provide just as much mixing as the first few cipher rounds and thus, saves time. The basic counterargument is that a larger block size also loses time in CPU models. On most CPUs, the communication cost of sweeping through a 256-byte block is a bottleneck, because they have been designed for computations that do not involve so much data. However, CPU trends show that evolution will allow computing larger amounts of bits. Thus, future algorithms should take advantage of this, increasing the currently typical 128 bit block size. Parallel processing is also a performance booster in speed and scalability.

Some ciphers sacrifice security strength attempting to obtain higher speed. Nowadays, 256 bit keys are used and considered secure, since the computational efforts in trying to break their security are considered nearly impracticable. However, the recent multi-core CPU trends indicate this key length will be rapidly surpassed as hardware processing power evolves. Thus, to avoid rapidly becoming useless, at least 256 bit or higher key lengths should be used in the development of new solutions. Although higher keys should, in principle, bring worse performance, in our opinion the problem is not centered on the key length, but on the used block size and the algorithm itself.

There is always a tradeoff between performance and security; research will probably lead to solutions that are better in database performance, but have less security strength. The main issue is to significantly decrease storage space, resource consumption and response time, while maintaining substantial security strength. A possibility is to develop variable-based dynamic algorithms that enable the user to choose between different key lengths and block sizes, the number of encryption/masking rounds, and other parameters allowing DBAs and application developers to fine tune the security-performance tradeoff's balance according to the specific features and requirements of each DW.

5 Conclusions

We have presented the available confidentiality solutions for databases and described the performance issues concerning their use in DWs. Experimental evaluations included in state-of-the-art standards and published research show that the storage space and response time overheads introduced by encryption algorithms dramatically degrade database performance to a magnitude that jeopardizes their feasibility in data warehousing environments. Our experiments have also confirmed this.

A data confidentiality solution may be useless if it assures a high level of protection, but is too slow to be considered acceptable in practice. Since database performance is a critical issue in data warehousing scenarios, we conclude that current encryption solutions are not suitable for DWs. DWs function in a well-determined specific environment with tight security, performance and scalability requirements and, therefore, need specific solutions able to cope with these directives. Since there is always a tradeoff between security strength and performance, developing specific data confidentiality solutions for DWs must always balance security requirements with the desire for high performance, *i.e.*, ensuring a strong level of security while keeping database performance acceptable. This is a critical issue and remains a challenge, which makes ground for opportunities in this domain, given the lack of specific solutions for data warehousing environments.

References

1. 3DES. Triple DES. National Institute of Standards and Technology (NIST), Federal Information Processing Standards (FIPS), Pub. 800-67, ISO/IEC 18033-3 (2005)
2. AES. Advanced Encryption Standard. NIST, FIPS-197 (2001)
3. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-Preserving Encryption for Numeric Data. In: ACM SIG Int. Conference on Management of Data, SIGMOD (2004)
4. Agrawal, R., Srikant, R., Thomas, D.: Privacy Preserving OLAP. In: ACM Int. Conference of the Special Interest Group on Management of Data, SIGMOD (2005)
5. Bertino, E., Sandhu, R.: Database Security – Concepts, Approaches, and Challenges. IEEE Transactions on Dependable and Secure Computing 2(1) (2005)

6. DES. Data Encryption Standard. National Inst. of Standards and Technology (NIST). FIPS Pub. 46 (1977)
7. Ge, T., Zdonik, S.: Fast, Secure Encryption for Indexing in a Column-Oriented DBMS. In: International Conference on Data Engineering, ICDE (2007)
8. Huey, P.: Oracle Database Security Guide 11g. Oracle Corporation (2008)
9. Kimball, R., Ross, M.: The Data Warehouse Toolkit, 2nd edn. Wiley & Sons, Inc. (2002)
10. Nadeem, A., Javed, M.Y.: A Performance Comparison of Data Encryption Algorithms. In: Int. Conf. on Information and Communication Technologies, ICICT (2005)
11. Natan, R.B.: Implementing Database Security and Auditing. Digital Press (2005)
12. Oracle Corporation. Data Masking Best Practices, Oracle White Paper (2010)
13. Oracle Corporation. Oracle Advanced Security Transparent Data Encryption Best Practices, Oracle White Paper (2010)
14. Procopiuc, C.M., Srivastava, D.: Efficient Table Anonymization for Aggregate Query Answering. In: Int. Conf. on Data Engineering, ICDE (2011)
15. Radha, V., Kumar, N.H.: EISA - An Enterprise Application Security Solution for Databases. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2005. LNCS, vol. 3803, pp. 164–176. Springer, Heidelberg (2005)
16. Ravikumar, G.K., Manjunath, T.N., Ravindra, S.H., Umesh, I.M.: A Survey on Recent Trends, Process and Development in Data Masking for Testing. International Journal of Computer Science Issues 8(2) (2011)
17. TPC-H. The TPC Decision Support Benchmark H, `http://www.tpc.org/tpch/default.asp`
18. Vimercati, S.C., Foresti, S., Jajodia, S., Paraboschi, Samarati, P.: Over-encryption: Management of Access Control Evolution and Outsourced Data. In: International Conference on Very Large DataBases, VLDB (2007)
19. Xiao, X., Bender, G., Hay, M., Gehrke, J.: iReduct: Differential Privacy with Reduced Relative Errors. In: ACM SIG Int. Conf. on Management of Data, SIGMOD (2009)
20. Yuhanna, N.: Your Enterprise Database Security Strategy. Forrester Research (2010)
21. Gartner Inc. Selection Criteria for Data-Masking Technologies. Research Report ID G00165388 (February 2009)
22. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008)
23. Hacıgümüş, H., Iyer, B., Mehrotra, S.: Efficient Execution of Aggregation Queries over Encrypted Relational Databases. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 125–136. Springer, Heidelberg (2004)
24. Schneier, B.: The Blowfish Encryption Algorithm, `http://www.schneier.com/blowfish.html`
25. Elminaam, D., Kader, H., Hadhoud, M.: Evaluating the Performance of Symmetric Encryption Algorithms. Int. Journal of Network Security 10(3) (2010)
26. Bernstein, D.J., Schwabe, P.: New AES Software Speed Records. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 322–336. Springer, Heidelberg (2008)

A New Paradigm for Collaborating Distributed Query Engines

Qiming Chen and Meichun Hsu

HP Labs
Palo Alto, California, USA
Hewlett Packard Co.
{qiming.chen,meichun.hsu}@hp.com

Abstract. In this work we focus on using Distributed Caching Platform (DCP) to scale out database applications and to support relational data communication of multiple individual query-engines in a general graph-structured SQL dataflow process. While the use of DCP has gained popularity lately, transferring query results from one query-engine to another tuple-by-tuple through DCP is often inefficient; this is because the granularity of cache access is too small, and the overhead of data conversion and interpretation is too large.

To deal with these issues, we leverage DCP's binary protocol and query-engine's buffer management to deliver query results at the storage level directly. We extend the database buffer pool over multiple memory nodes to enable low-latency access to large volumes of data, and introduce the novel ***page-feed*** mechanism to allow the query results of the collaborative query-engines to be communicated as data ***pages*** (***blocks***), namely, the producer query puts its result relation as pages in the DCP to be got by the consumer query. In this way, data are transferred as pages directly under DCP's binary protocol, where the contained tuples are exactly in the format required by the relational operators, and the use of pages, as mini-batches of tuples, provides the balanced efficiency of query processing and DCP access. Pushing relation data communication down to the *storage (buffer pool) level* from the *application level* offers significant performance gain, and is naturally consistent with the SQL semantics. We have implemented these specific mechanisms on a cluster of PostgreSQL engines. Our experiment results are documented in this paper.

1 Introduction

1.1 Support Graph-Structured Dataflow with Distributed Query Engines

In-DB analytics offers the benefits of fast data access, reduced data transfer and SQL's rich expressive power [2]. Since a general graph-structured dataflow cannot be modeled by a single tree-structured SQL query (regardless of the query runs on a single node DB or on a parallel DB), we model it as a process with multiple correlated queries which form sequential, parallel or conditional steps. A very simple dataflow process example for network traffic analysis is given in Fig. 1(a). It has three *named queries* Q_1, Q_2 and Q_3, the source table of Q_1 is *Traffic,* with schema [*tid, second, fromIP, toIP, bytes*], describing the IP-to-IP network traffic records. Q_1 retrieves the

IP-to-IP network traffic records, converts them to minute-based, host-to-host traffic; the result of Q_1 is forked to Q_2 and Q_3 for aggregation. These queries are specified as follows.

Q_1 := SELECT tid, FLOOR(time/60)::INTEGER AS minute, h1.host-id AS from-host, h2.host-id AS to-host, bytes FROM Traffic, hosts h1, hosts h2 WHERE h1.ip = from-ip AND h2.ip = to-ip

Q_2 := SELECT minute, from-host, to-host, SUM(bytes) FROM $$Q_1$ GROUP BY minute

Q_3 := SELECT minute, from-host, to-host, SUM(bytes) FROM $$Q_1$ GROUP BY from-host, to-host

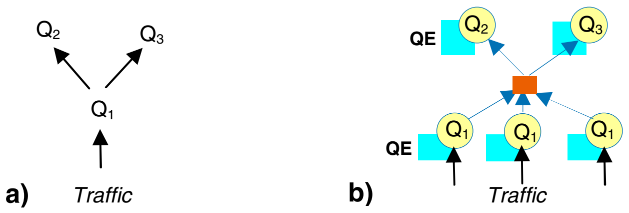


Fig. 1. (a) A SQL dataflow process with 3 queries; (b) Execute dataflow process in terms of multiple distributed query engines (QEs)

One way to scale out the above SQL dataflow process is to execute the queries by multiple distributed query-engines. As illustrated in Fig 1(b), *Traffic* tuples are identified and hash partitioned by *tid* across three query-engine nodes, to be processed by three Q_1 instances in parallel; the union of the results from all Q_1 executions, denoted by $\$Q_1$, is sent to the query-engines running Q_2 and Q_3 as their input data, for further processing.

In a Query-Engine Net, the efficient transfer of data among query-engines is a critical issue, and we investigate the use of the unified distributed cache across multiple server nodes, generically referred to as a Distributed Cache Platform (DCP) [1,5-9], for delivering query results among distributed query-engines.

1.2 The Issues

Refer to Fig 2, the upper arc illustrates the current practice of using DCP for data sharing and exchange at the application level, which often incurs significant overhead in data conversion and interpretation. For example, to take the result of a query Q_1 as the input of another query Q_2, the typical way is to encode Q_1's resulting tuple-set as a CSV array, and then convert it back to tuples before passing to Q_2, which may not be suitable for every kind of application, and with large or continuous data feed, become a serious performance bottleneck. Further, transferring query results from one query-engine to another tuple-by-tuple through DCP is often inefficient due to the tiny granularity of cache access and the overhead of data conversion and interpretation.

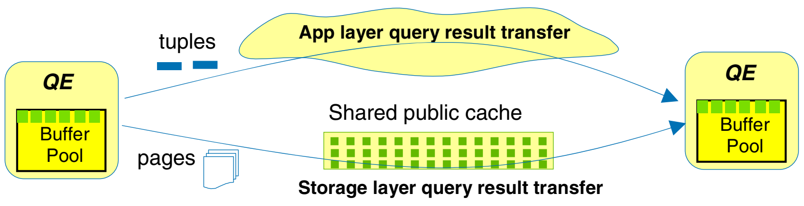

tuples
App layer query result transfer
QE
Buffer Pool
Shared public cache
pages
QE
Buffer Pool
Storage layer query result transfer

Fig. 2. Transfer query result between query engines directly as pages (blocks)

1.3 The Proposed Approach

The above observations have motivated us to leverage DCP's binary protocol for transporting relation data *directly at the storage level as pages* (*blocks*) instead of at the application level. This idea is illustrated by the lower arc in Fig 2.

Our solution includes two steps. First, we extend the database buffer pool to the DCP, resulting in the unified database cache across distributed memory spaces on multiple machines, which allows scaled-out, low-latency in-memory data access. The query-engine uses the normal buffer pool management mechanism to access and interpret the content stored in DCP, thus eliminating the need for supporting these capabilities by the DCP. However, we introduce the inclusion model for extending the buffer pool to DCP, which is different from the existing approach and required for the data transfer among multiple query-engines.

Next, we extend the above solution to provide the shared memory-based paradigm for the collaboration of multiple query-engines in a dataflow process, in the sense of allowing a query to feed its result efficiently to another query through DCP. However, different from having a query result passed as an *application layer* "object" through DCP, we have the query result relation delivered and retrieved directly at the storage layer using a ***page-feed*** mechanism, namely, the producer query puts its resulting relation as pages (blocks) in DCP (through its buffer pool), and the consumer query gets these pages from DCP directly. Since the query result pages are emitted and retrieved with the binary protocol, and the tuples in these pages are exactly in the format required by the relational operators, this approach avoids the overhead of application specific encoding/decoding. Further, the use of pages as the mini-batches of tuples reduces the overall latency of DCP access.

We have built the proposed infrastructure by integrating the buffer management of multiple PostgreSQL engines with DCP (Memcached). Our experience reveals the value of pushing data communication from the *application layer* down to the *storage layer* in supporting large scale, low-latency in-memory data access, as well as efficient communication among multiple collaborative query-engines.

The rest of this paper is organized as follows: Section 2 describes how to scale out the buffer pool with DCP under the proposed inclusion model; Section 3 deals with the buffer externalization of multiple query-engines for their collaboration; Section 4 shows our experimental results; Section 5 compares with the related work and concludes the paper.

2 Extend Database Buffer Pool to Distributed Caching Platform

2.1 Database Buffer Management

The description of our solution is based on the PostgreSQL engine. In a PostgreSQL database, each table is physically stored in the file system under a subdirectory with several of files. A single file holds certain amount, up to 1GB of data, as a series of fixed-sized *blocks* (i.e. ***pages,*** typically 8K in size although configurable). A tuple may not span multiple pages where a large tuple is sliced to multiple physical ones by the TOAST utility transparently.

A database buffer pool is a shared in-memory data structure - a simple array of *pages* (blocks), with each page entry pointing to a binary memory of 8K size. A page in the buffer pool is used to buffer a block of data in the corresponding file, and is identified by a ***tag*** - the IDs of table space, relation, file and the sequence number of the block in the file, as <*table-space-id, relation-id, file-id, block#*>. Maintaining the buffer pool allows the pages to be efficiently accessed in memory without going to disks.

The buffer pool is accompanied by a corresponding array of data structures called buffer descriptors, with each recording the information about one page, such as the *tag* as its identifier, the usage frequency, the last access time, whether the data is ***dirty*** (updated), and whether it is ***extended*** (i.e. a newly allocated page being filled by inserted tuples to *extend* the relation). An "extended" page is a "dirty" page as well.

When a query process needs a page corresponding to a specific file/block, if the block is already cached in the buffer pool, the corresponding buffered page gets pinned; otherwise, a page slot must be found to hold this data. If there are no slots free, the process selects a page to evict to make space for the requested one. If the page to be evicted is dirty, it is written out to disk asynchronously. Then the requested block on disk is read into the page in memory.

Pages all start out pinned until the process that requested the data releases (unpins) them. Deciding which page to remove from the buffer pool to make space for a new one is the classic cache replacement problem. The usual strategy is Least Recently Used (LRU). The timestamp when each page was last used is kept in the corresponding buffer descriptor in order for the system to determine the LRU page. Another way to implement LRU is to keep pages sorted in order of recent access. There exist several other page eviction strategies such as Clock-Sweep and Usage-Count. Hereafter we simply refer to the page to be evicted as the LRU page.

2.2 Distributed Cache Platform (DCP)

A DCP provides the unified cache view over multiple machine nodes, which allows multiple processes to access and update shared data [1,5-9] using the key-value based APIs such as *get(), put*(), *delete*(), etc, where keys and values are objects.

Memcached [5] is a general-purpose distributed memory caching system that holds data in a large hash table distributed across multiple machines, or memory nodes. The data are hash partitioned to these memory nodes. When the hash table on a node is full, subsequent insert causes Least Recently Used (LRU) data to be purged. Memchached uses the client–server architecture. The servers maintain a key–value associative array; the clients populate this array and query it. Keys are up to 250 bytes long and values can be at most 1 megabyte in size. Clients know all servers and use client side libraries to contact the servers. If a client wishes to *set* or *get* the value corresponding to a certain key, the client's library first computes a hash of the key to determine which server to use; then it contacts that server. The server will compute a second hash of the key to determine where to store or read the corresponding value.

2.3 Convert Buffer Pages to Key-Value Pairs

In integrating PostgreSQL engine with the Memcached-based DCP infrastructure, the query-engine acts as the DCP client that connects to the Memcached server pool. The basic idea of extending the buffer pool with Memcached is to store the buffered pages as key-value pairs hash-partitioned to separate portions of the unified hash table residing on separate nodes. The mapping of a buffered page to a key-value pair is handled in the following way.

Key. The tag for identifying a page, composed of the table-space-id, relation-id, file-id and the series number of the block in the file, is serialized to a string *key*. The mapping from *tag* to *key* is provided for Memcached access.

Value. The 8KB binary content of a page is treated as the *value* corresponding to the page key. Adopting Memcached's Binary Protocol this value is passed to the API functions of data transfer by the entry pointer plus the length of bytes.

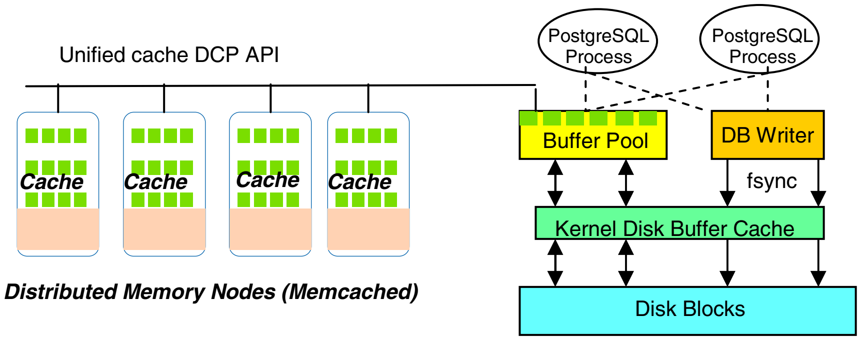


Fig. 3. Extend PostgreSQL shared buffer pool to DCP

2.4 Scale-Out Buffer Pool with Memcached

We extend the buffer manager of the PostgreSQL engine to allow buffered pages to be moved to and retrieved from the Memcached. The query engine acts as a DCP client. The buffered pages may be sent to different physical Memcached sites based on the hash value of the page key. As a result, these pages are placed in multiple "memory nodes" but can be accessed with the unified interface.

While treating DCP as additional buffer space, the concurrency control, page eviction management and file I/O are still handled by the database buffer pool manager. Any page to be cached in or retrieved from Memcached always goes through the buffer pool manager. There are no file I/O and database buffer management functionalities such as page locking, on the Memcached sites. This greatly simplifies the overall system design and ensures data consistency.

Potentially there exist two page buffering models.

- **Overflow Model of Page Buffering**, where an LRU page to be evicted from the buffer pool is written to Memcached; and if it is dirty, it is also "fsync'ed" to disk. This way, if a page is not in the buffer pool but in the Memcached, its content in the Memcached is up to date. A page can only be pinned in the buffer pool. In general given the buffer pool B and the DCP buffering space D (physically located in distributed memory nodes), the unified page buffer is B $\cup$ D, and B $\cap$ D = empty.
- **Inclusion Model of Page Buffering**, where a new page is copied to DCP soon after it is filled rather than waiting until the eviction time. Tuus given the buffer pool B and the DCP buffering space D, the unified page buffer is B $\cup$ D, and B $\subseteq$ D, assuming that D is larger than B [13]. We will show next that this model is required for externalizing pages to DCP for other query-engines to access.

On a single query-engine, under both models, when a process requests a page, the system first tries to get the page from the local buffer pool; if the page is not found then it tries to get the page from DCP; if the page is not in DCP, then the page is loaded from disk. In general, which model will deliver better performance depends on the workload characteristics. However, in this research, introducing the inclusion model does not focus on performance variety, but on externalizing pages to DCP for multiple collaborative query-engines to share them.

3 Globalize Buffers of Multiple Query-Engines

Our next step is to externalize the buffer pools of multiple distributed query-engines to DCP for shared memory-based collaboration in a dataflow process. This concept is illustrated in Fig 4 based on the SQL dataflow process example shown in Fig 1.

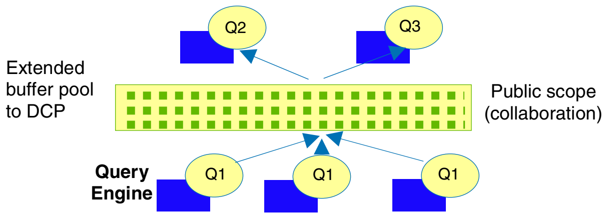


Fig. 4. Connect queries and transfer results through DCP

In this context, a dataflow process is made of multiple "cascading" queries running on distributed query-engines, where a query, say Q_2, is applied to the result of its upstream query, say Q_1; the query-engine for executing Q_2 checks DCP for the availability of Q_1's results, and retrieves them as its own input data. The buffered query result will be invalidated after consumed. If the DCP space reaches its limit (a very rare case), first the already consumed upstream query results will get evicted, and next the DCP data content get extended to the file system.

For efficiency, we allow a query result relation to be delivered and retrieved as pages (blocks), referred to as ***page-feed***. Since pages (blocks) are fix-length binaries, they are transferred with the commonly applicable binary protocol, while tuples in the pages are in exactly the format required by the relational operators, avoiding the performance overhead of data encoding/decoding.

The page-feed approach assumes the use of homogeneous query-engines (in our implementation, they are all PostgreSQL engines). The specification of the collaborative SQL dataflow process is known to all the participating query-engines, such that the name of a query, say *Q*, and its result relation *$Q*, are known, and *$Q*'s schema is created at each related query-engine. The query result relations ***externalized*** (populated) to DCP from the query-engines, referred to as the ***external relations***, form the **public data scope** of these engines where each engine still has its own **private data scope**. External relations always reside in DCP.

3.1 Support Insert-Oriented Inclusion Model for Externalizing Buffered Pages

For retrieving a page by a local query-engine, as far as the page is located in the buffer pool, DCP or disk, it is accessible; and therefore the overflow model for extending the buffer pool to DCP is sufficient. However, to have a page generated by one query-engine accessible by another, that page must be located in the DCP, since a query-engine cannot access the local buffer and disk of a foreign database. For the above reason we introduce the inclusion model to extend and externalize the buffer pool of a query-engine to the public DCP space.

Specifically, we focus on the "insert-oriented" inclusion model, namely, the DCP content includes the buffer pool content in the insertion context only; this is because we assume the query results to feed in a foreign query-engine are only appended but *never got updated.*

Under the overflow model, a page is written to Memcached only when it becomes the LRU page to be evicted from the buffer pool. However, under the ***inclusion model***, when a page is loaded from file, or newly inserted into the buffer pool, it is copied to DCP immediately rather than waiting until the eviction time. When a LRU page is evicted, if it is *dirty*, it is written to the disk, as well as transmitted to DCP to refresh its copy in DCP. Fig 4 shows the buffer management in a single query-engine under the inclusion model.

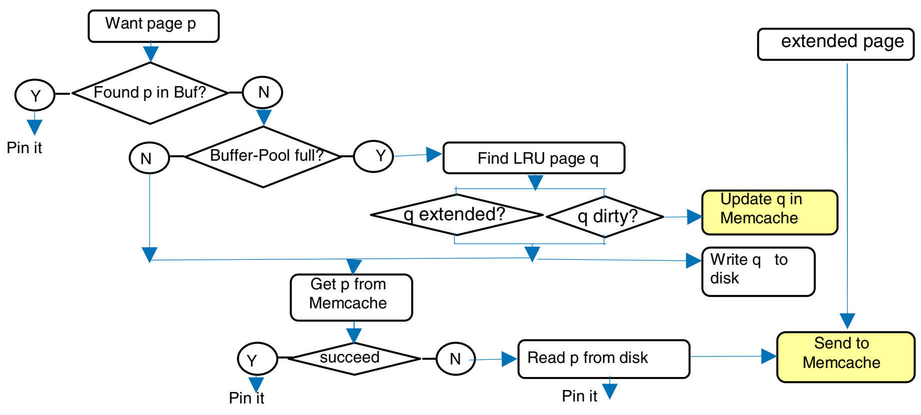


Fig. 5. Extend **local** buffer pool to DCP under *inclusion* model

To implement the inclusion model, it is important to figure out *when* to buffer a new page to DCP, and avoid the following two treatments.

- We should avoid buffering incomplete pages in DCP when they are newly created, not "full" and under subsequent insertion, since which would result in fetching incomplete pages from DCP later, leading to data loss.
- We should also avoid overwriting the page buffered in DCP with each tuple inserted (not updated) into that page, since which would generate too much traffic.

Instead, we should buffer a page, say P, in DCP when it is "just-filled", i.e. the insert to it has been just completed. In this case, the DCP copy of that page, P_d contains the full data that is the same as the copy of that page in the buffer pool, P_b. This mechanism ensures that the data in the DCP and in the buffer pool are consistent, and ensures the inclusion semantics.

For identifying the just-filled buffer and sending it to the DCP, the major extension to the PostgreSQL engine is made in the heapam.c program under the access/heap directory. The basic logic is the following: when a tuple is to be inserted into a relation R, if the current block# of R is larger than the last-block# of R, then the last-block is considered as "just filled" and sent to DCP.

Extending the buffer pool to DCP under the inclusion model is critical to ***externalize*** the buffered pages for passing query results from one query-engine to another.

3.2 Externalize Buffered Pages to DCP

To externalize a page as a key-value pair, its key, called ***external key*** must contain a *site-id* field for indicating the query-engine where the page is originated, and the local relation ID must be replaced by a globally known *relation name*. At each query-engine, paging is still handled by the local buffer manager, but only the local pages may be updated. The external pages are retrieved from DCP as ***read-only***. For the external page generated locally, the mapping between its local tag and external-key is provided.

There is a conceptual difference between scaling out the buffer pool of a single query-engine using DCP and externalizing a query result relation to be shared by other query-engines using DCP. In the former case, a page in DCP should be made up to date only when it no longer exists in the buffer pool. In the latter case, the pages of an *external relation* in DCP must always be up to date since the DCP is the primary place to share them. Our Inclusion Model mechanism ensures this property as explained below:

- First, an external relation *R* is always produced as a query result (e.g. Select * *into R* from T) of a query executed on the producer query-engine.
- Next, whenever a new page *p* of the external relation *R* is created and full with newly inserted tuples, or whenever the computation of *R* terminates, *p* becomes a regular page and is immediately transferred to DCP to satisfy the Inclusion Model.
- Further, once the query that produces *R* is completed, *R* as the input to other queries is *read-only,* so when *R*'s pages are evicted from the consumer query-engine's buffer pool, updating their counterparts in DCP is unnecessary; as a result, the content of *R's* pages in the DCP are kept up to date at all times under the inclusion model.
- As we assume the query results to feed in a foreign query-engine are only appended to the DCP but *never got updated*, our inclusion model is "insert-oriented".

3.3 Deliver Query Result to DCP as Pages

We assign a name, say *Q*, to a query participated in a SQL dataflow process, and have the query-engine convert a named query *Q* into a SELECT INTO query, and put the query result in the "into-relation", *$Q*, with its pages being held in the local buffer pool as well as externalized to the DCP to be accessed by distributed query-engines. When a page is externalized to DCP, its *tag* (local ID) and *content* are converted to the following key-value pair.

- **External-Key.** For DCP access, the string key of a page is serialized from <***site-id,*** *table-space-id,* ***relation-name,*** *file-id, block#*>. Different from the page key with a single query-engine, the ***site-id*** is introduced to identify the query-engine where the page is originated, and the local *relation-id* is replaced by the commonly-known ***relation-name***. The mapping between the local tag of a page and its external-key is provided.
- **Value.** The 8KB binary content of the page is treated as the value corresponding to the external page key.

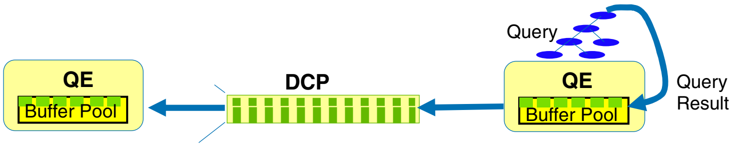


Fig. 6. Deliver query result in pages at the storage layer of query engine (QE)

Further, a query in the dataflow process may run in parallel at multiple sites with each generating a partition of the result relation with the same name. This requires the following generalization:

- Given an external relation, for each applicable site, a site-specific **master-key** is composed by the *relation-name*, say, *T* and *site-id,* say *k,* as *T.k.* A key-value pair <*master-key*, *page-key-list*> of *T* is created and stored in the DCP when the list is completed. Then for all the applicable sites, say site *1,…, 8*, the *page-key-lists* of *T*, keyed by "*T.1*", …, "*T.8*" are provided in the DCP.
- More specifically, at the site *k*, the pages of *T* are loaded to DCP with their page keys kept in a list, and then the list is itself loaded to DCP with *T.k* as the key.
- Since the site-ids and the resulting relation are known to every participating query-engine, the above site-specific master keys for a relation are known to all of them.
- When *T* is to be retrieved from DCP by the consumer query-engine, the known list of site-ids, say, *1,…, 8*, are first used to compose master-keys, *T.1*, …, *T.8*, which are in turn used by the consumer query-engine to retrieve (using the *mget,* or multi-get call) all the page keys belonging to *T*; then these page keys are used as keys to *get* those pages. As an example, it is easy to see that this approach naturally handles the delivery of Map query results to the Reduce sites in a Map-Reduce style dataflow process.

3.4 Fetch External Pages from DCP

To explain how an external page cached in DCP is accessed, we first review how a local page is accessed.

A local page is identified by a tag <*table-space-id, relation-id, file-id, block#*>. A regular full-table-scan (FTS) first gets all the page tags from the system, say Data Dictionary (DD), indices, etc, and then retrieves the corresponding pages through the storage engine.

In the situation discussed here, at a particular query-engine, a query gets input data as an external relation (the results of other queries) from the physically distributed but logically unified DCP cache space. Since the information about the external relation partitions on the foreign sites are not kept in the local Data Dictionary, such cache access cannot be guided by the local DD in the same way as the regular FTS. This requires us to provide a particular cache access method.

Cache access to external pages is handled by the buffer pool manager of the requesting query-engine with the following constraints: the Full-Table-Cache-Scan (FTCS) is assumed, i.e. retrieving all pages of a relation from the DCP memory space; further, FTCS is made on the **read-only** basis.

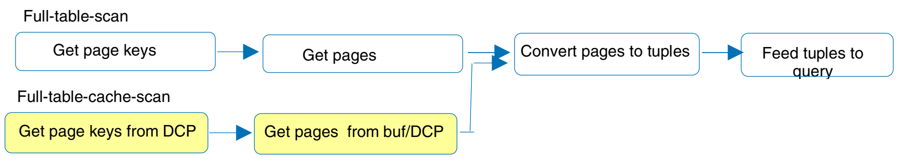


Fig. 7. Access query input as pages from DCP

As shown in Fig 7, different from the FTCS that gets page tags (IDs) from Data Dictionary, the Full Table Cache Scan first uses the master-keys of the requested relation, *T*, to *mget* (multi-get) from DCP all the page keys of *T*, with each composed with <*site-id, table-space-id, relation-name, file-id, block#*>; then in the second phase, gets the pages using these keys, to the buffer pool of the requesting query-engine

4 Preliminary Experimental Results

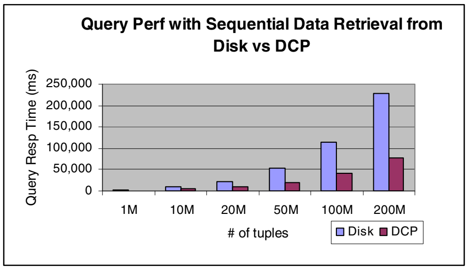
4.1 Performance of Extending Buffer Pool to DCP `under the Inclusion Model

We first test the effect of extending the buffer pool of a single query-engine to DCP. We used Linux servers with gcc version 4.1.2 20080704 (Red Hat 4.1.2-50), 32G RAM, 400G disk and 8 Quad-Core AMD Opteron Processor 2354 (2200.082 MHz, 512 KB cache). One server has PostgreSQL 8.4 installed; the other 8 have Memcached installed. For our experiments, we configure each of these systems with a buffer cache size of at least 1/8th of the database size, while varying the database sizes from 50MB to 10GB. In other words in our current experiments we assume that Memcached is big enough to hold the entire database. The database consists of a single table *T* with 1 million to 200 million tuples, where each tuple is 50 bytes containing 3 attributes: *pid, x,* and *y*. All experiments are performed with a warm start; i.e., after the buffer cache on both the PostgreSQL node and the Memcached nodes are filled with data by first running a sequence of sequential scan queries.

The comparison is made between a conventional PostgreSQL engine under the regular buffer pool management and an extended engine where the data are additionally buffered on the distributed Memcached nodes.

The performance comparison of sequential data retrieval (query pattern: *Select * from T where …*) is shown in Fig. 8 (left). The speedup ratio under our approach range from 3x to 8x with the number of input tuples from 1 million to 200 million.

The performance comparison of indexed retrieval is given in Fig. 8 (right). The average performance gains for varying database sizes range from approximately 6x to 10x. In general, the query performance gain with DCP strongly depends on the query workload characteristics.



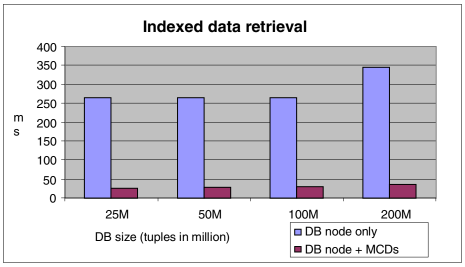


Fig. 8. Performance gain by extend buffer pool to DCP

Since only the inclusion model is suitable for consistently externalizing buffer pool to DCP, we need to compare the performance using our inclusion model and using the existing overflow model. Fig 9 compares the data access latency under the inclusion model and the overflow model. In this experiment the table with 10M, 50M, 100M and 200M are first loaded into the buffer pool and Memcached cache by the sequential scan query Q1 (First Query Run), after that Q1 is run again (Second Query Run). In the first run the performance of the overflow model is about the same as that of the inclusion model. However, in the second and all subsequent runs, the inclusion model shows a gain up to 40%. To explain the above experiment results, note that in our experiments, the database size is larger than the buffer pool size, B, and the total cache size of Memcached, M, is larger than the database size. In the initial data loading phase, under the overflow model, most pages loaded to B will eventually "overflow" (evicted and moved) to M, therefore the costs of loading pages to Memcached under the overflow model and the inclusion model are similar. However, after most or all pages already kept in M, under the overflow model, every page evicted from B has to be moved to M; but under the inclusion model, only the dirty pages evicted from B will involve moving the content to M (to refresh the corresponding content in M); for non-dirty pages, only a notification to M is performed.

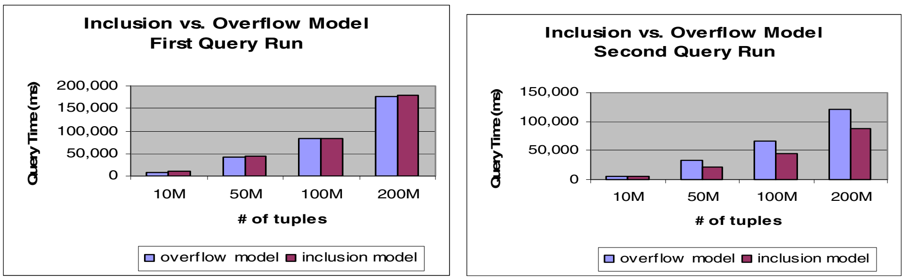
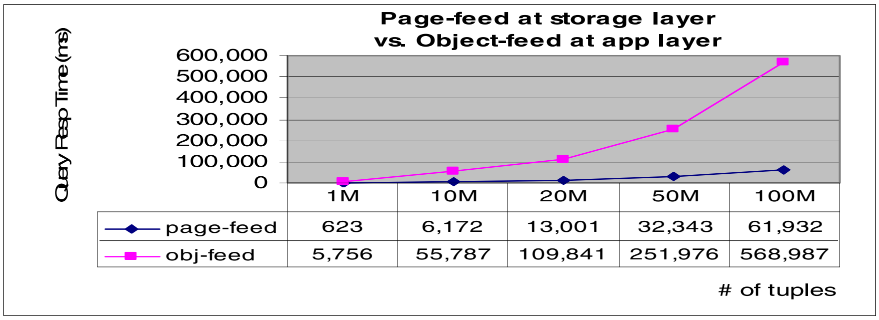


Fig. 9. Comparison of Inclusion Model and Overflow Model

4.2 Performance Gain by Feeding Query Results as Binary Pages

This work is primarily motivated by leveraging DCP's binary protocol and query engine's buffer management for delivering query results from one query-engine to another efficiently. We compare two different ways to put-to /get-from DCP the query results: as a CSV array at the application layer, and as the externalized pages at the storage layer as we proposed in this work. Converting query results tuple by tuple as CSVs incurs the latency of per-tuple processing, as well as the data conversion overhead for each input tuple. Under the proposed *page-feed* approach, however, can overcome these problems since the query results are transferred in mini-batch (page) with lower latency, and the data conversion overhead can be eliminated. The performance comparison results given in Fig 10 show that our page-feed mechanism significantly out-performs the mechanism that goes through the application layer.



	1M	10M	20M	50M	100M
page-feed	623	6,172	13,001	32,343	61,932
obj-feed	5,756	55,787	109,841	251,976	568,987

Fig. 10. Performance gain by using page-feed

5 Conclusions

We have examined the mechanisms of extending database buffer pool with a Distributed Caching Platform (DCP), externalizing the buffered pages of multiple query engines, and transferring intermediate query results among them directly through *page-feed*. In running a graph-structured SQL dataflow process with multiple query-engines, this approach supports efficient data communication.

Several database products such as Oracle, MySQL, EnterpriseDB are provided with DCP interface for database applications coded in PL-SQL, PSQL, or User Defined Functions (UDFs)[1,5-9]. While these efforts also take advantage of DCP's unified cache view for storing application data, they are orthogonal to extending database buffer pool with DCP. The proposed approach also differentiates from the existing DCP stores such as Membase[12] without the SQL interface. In using DCP at the database buffer pool layer, our approach differs from the Waffle Grid Project on MySQL [11] by using the inclusion model, rather than the overflow model, for page externalization.

The proposed page-feed approach out-performs caching query results as "application objects" in DCP [5-8] by eliminating the data conversion overhead. Pushing query-engines' data communication from the application-oriented layer down to the system-oriented buffer pool layer is the unique feature of our approach. Note that the page-feed mechanism is also applicable to *chunk-wise stream processing* [3, 10].

There exist quite a few inter-server data communication mechanisms such as messaging, RPC, etc, with each having the strength for a class of applications. Our approach for multiple query-engines to transfer intermediate query results as buffer pages through distributed shared memory is unique. Viewing distributed caching as an abstraction of messaging, we will continue examining its benefits and trade-offs in enabling scalable distributed analytics.

Although our investigation to the Query Engine Net is in the initial phase, we envisage its potential in the distributed data intensive computation, and based on this vision to guide our future research direction.

References

[1] Nori, A.: Distributed Caching Platforms. In: VLDB 2010 (2010)

[2] Bryant, R.E.: Data-Intensive Supercomputing: The case for DISC, CMU-CS-07-128 (2007)

[3] Chen, Q., Hsu, M., Zeller, H.: Experience in Continuous analytics as a Service (CaaaS). In: EDBT 2011 (2011)

[4] Chen, Q., Hsu, M.: Query Engine Net for Streaming Analytics. In: Proc. 19th International Conference on Cooperative Information Systems, CoopIS (2011)

[5] Memcached (2010), `http://www.memcached.org/`

[6] EhCache (2010), `http://www.terracotta.org/`

[7] Vmware vFabric GemFire (2010), `http://www.gemstone.com/`

[8] IBM Websphere Extreme Scale Cache (2010), `http://www.ibm.com/`

[9] AppFabric Cache (2010), `http://msdn.microsoft.com/`

[10] Liarou, E., et al.: Exploiting the Power of Relational Databases for Efficient Stream Processing. In: EDBT 2009 (2009)

[11] The Wafflegrid Project, `http://www.wafflegrid.com/`

[12] Membase, `http://www.couchbase.com/`

[13] Baer, J., Wang, W.: On the inclusion properties for multi-level cache hierarchies. In: ISCA 1988 (1988)

A Lightweight Stream-Based Join with Limited Resource Consumption

M. Asif Naeem[1], Gillian Dobbie[2], and Gerald Weber[2]

[1] School of Computing and Mathematical Sciences, Auckland University of Technology
[2] Department of Computer Science, The University of Auckland
mnaeem@aut.ac.nz
{gill,gerald}@cs.auckland.ac.nz

Abstract. Many stream-based applications have plenty of resources available to them, but there are also applications where resource consumption must be limited. For one important class of stream-based joins, where a stream is joined with a non-stream master data set, the algorithm called MESHJOIN was proposed. MESHJOIN uses limited memory and is a candidate for a resource-aware system setup. The problem that is considered in this paper is that MESHJOIN is not very selective. In particular, the performance of the algorithm is always inversely proportional to the size of the master data table. As a consequence, the resource consumption is in some scenarios sub-optimal. We present an algorithm CACHEJOIN, which performs asymptotically at least as well as MESHJOIN but performs better in realistic scenarios, particularly if parts of the master data are used with different frequencies. In order to quantify the performance differences, we compare both algorithms using a synthetic data set with a known skewed distribution.

Keywords: Stream processing, Join operator, Performance measurement.

1 Introduction

Stream-based joins are important operations in modern system architectures, where just-in-time delivery of data is expected. We consider a particular class of stream-based join, namely the join of a single stream with a slowly changing table. Such a join can be applied in real-time data warehousing [7,5]. In this application, the slowly changing table is typically a master data table. Incoming real-time sales data may comprise the stream. The stream-based join can be used for example to enrich the stream data with master data. The most natural type of join in this scenario would be an equijoin, performed for example on a foreign key in the stream data.

For executing stream-based operations, the large capacity of current main memories as well as the availability of powerful cloud computing platforms means, that considerable computing resources can be utilized. For master data of the right size for example, main-memory algorithms can be used.

However, there are several scenarios, where stream joins that use a minimum of resources are needed. One particular scenario is an organization trying to reduce the carbon footprint of the IT infrastructure. A main memory approach as well as cloud-computing approaches can be power-hungry. Also in the area of mobile computing and embedded devices a low-resource consumption approach can be advantageous. Therefore, approaches that can work with limited main memory are of interest.

In the past, the algorithm MESHJOIN [10,9] was proposed for joining a stream with a slowly changing table with limited main memory requirements. This algorithm is an interesting candidate for a resource aware system setup. The MESHJOIN algorithm also has few requirements with respect to the organization of the master data table. However, a limitation of MESHJOIN is that the performance is directly coupled to the size of the master data table, and its performance is inversely proportional to the size of the master data table. This is an undesired behavior if the master data becomes very large, and our analysis will show that a more adaptive behavior is possible with a new join algorithm that we will present. The problem with non-adaptive behavior becomes obvious if we consider a scenario where the master data table contains a large part that is never joined with the stream data. This situation can easily arise if the master data table is storing data for long term availability, and in the current business process only a fraction is used. A typical scenario would be catalogue data for seasonal products. In MESHJOIN, if one contiguous half of the master data is unused, the presence of this master data still halves the performance. This is undesirable, especially under our resource consumption viewpoint, since the algorithm still uses the same resources for half of the performance. This would put the burden on the administrator to meticulously clean master data in order to optimize system performance. Therefore it would be a great advantage if we have an algorithm that shares the advantages of MESHJOIN, but adapts itself to certain situations, for example if the algorithm is more sensitive to the usage of the master data.

In this paper we present a new algorithm, CACHEJOIN, that is able to utilize differences in the access frequency of master data. CACHEJOIN's performance is not affected, if a large set of unused data is added to the master data table. We are interested in understanding the relative performance of the MESHJOIN and CACHEJOIN algorithms. As we have said before CACHEJOIN will be unaffected by contiguous unused master data, therefore it is easy to create data sets where CACHEJOIN is arbitrarily better than MESHJOIN. However, in order to test our algorithm in a scenario that is not biased against any algorithm, we were looking for characteristics of data that are considered ubiquitous in real world scenarios. A Zipfian distribution of the foreign keys in the stream data matches distributions that are observed in a wide range of applications [2]. We therefore created a data generator that can produce such a Zipfian distribution. A Zipfian distribution is parameterized by the exponent of the underlying power law. In different scenarios, different exponents are observed, and determine whether the distribution is considered to have a short tail or a long tail. Distributions with a short tail would be more favourable for CACHEJOIN from the outset, therefore we decided not to use a distribution with a short tail in order to not bias our experiment towards CACHEJOIN. Instead we settled on a natural exponent that is observed in a variety of areas, including the original Zipf's Law in linguistics [6] that gave rise to the popular name of these distributions. The main result of our analysis is that CACHEJOIN performs better on a skewed data set that is synthetic, but following a Zipfian distribution as it is found frequently in practice. For our analysis we do not consider joins on categorical attributes in master data, e.g. we do not consider equijoins solely on attributes such as gender.

2 Related Work

In this section, we present an overview of the previous work that has been done in this area, focusing on those which are closely related to our problem domain.

R-MESHJOIN (reduced Mesh Join) [8] clarifies the dependencies among the components of MESHJOIN. As a result the performance has been improved slightly. However, R-MESHJOIN implements the same strategy as in the MESHJOIN algorithm for accessing the disk-based relation.

One approach to improve MESHJOIN has been a partition-based join algorithm [4] which can also deal with stream intermittence. It uses a two-level hash table in order to attempt to join stream tuples as soon as they arrive, and uses a partition-based waiting area for other stream tuples. For the algorithm in [4], however, the time that a tuple is waiting for execution is not bounded. We are interested in a join approach where there is a time guarantee for when a stream tuple will be joined, and therefore a guarantee that our algorithm is asymptotically as fast as MESHJOIN.

Another recent approach, Semi-Streaming Index Join (SSIJ) [3] joins stream data with disk-based data. Although SSIJ is a feasible approach for processing stream data, the algorithm does not include the mathematical cost model.

3 CACHEJOIN

The MESHJOIN algorithm reads cyclically through the whole master data table R. The throughput of the algorithm is inversely proportional for the size of R, as can be understood from the following argument: In MESHJOIN, each stream tuple must reside in main memory. The residence time of a tuple in main memory is proportional to the size of R, since MESHJOIN is reading through R. Hence the throughput is inversely proportional to the size of R.

In this paper, we propose a new algorithm, CACHEJOIN, that overcomes this strict proportionality. In order to allow CACHEJOIN to access the disk-based relation R selectively, an index is needed. However, a non-clustered index is sufficient, if we consider equijoins on a foreign key element that is stored in the stream.

The CACHEJOIN algorithm possesses two complementary hash join phases, somewhat similar to Symmetric Hash Join. One phase uses R as the probe input;

the largest part of R will be stored in tertiary memory. We call it the disk-probing phase. The other join phase uses the stream as the probe input, but will deal only with a small part of relation R. For each incoming stream tuple, CACHEJOIN first uses the stream-probing phase to find a match for frequent requests quickly, and if no match is found, the stream tuple is forwarded to the disk-probing phase.

The execution architecture for CACHEJOIN is shown in Figure 1. The largest components of CACHEJOIN with respect to memory size are two hash tables, one storing stream tuples denoted by H_S and the other storing tuples from the disk-based relation denoted by H_R. The other main components of CACHEJOIN are a disk buffer, a queue and a stream buffer. Relation R and stream S are the external input sources. Hash table H_R, for R contains the most frequently accessed part of R and is stored permanently in memory.

CACHEJOIN alternates between the stream-probing and disk-probing phases. According to the procedure described above, the hash table H_S is used to store only that part of the update stream which does not match tuples in H_R. A stream-probing phase ends if H_S is completely filled or if the stream buffer is empty. Then the disk-probing phase becomes active. The length of the disk-probing phase is determined by the fact that a few disk pages of R have to be loaded at a time in order to amortize the costly disk access. In the disk-probing phase of CACHEJOIN, the oldest tuple in the queue is used to determine the partition of R that is loaded for a single disk-probing phase into the disk buffer. The stream buffer is included in the diagram for completeness, but is in reality always a tiny component and it will not be considered in the cost model.

In this way, in CACHEJOIN it is guaranteed that every probe step processes at least one stream tuple, while in MESHJOIN there is no such guarantee. This can be extended to an argument that CACHEJOIN performs asymptotically at least as well as MESHJOIN, that is given in Section 3.1. This is also the step where CACHEJOIN needs an index on table R in order to find the partition in R that matches the oldest stream tuple. After one probe step, a sufficient number of stream tuples in H_S is matched. These tuples are deleted; the queue supports this process. After the disk-probing phase the algorithm switches back to the stream-probing phase. One phase of stream-probing with a subsequent phase of disk-probing constitutes one outer iteration of CACHEJOIN. The disk-probing phase could work on its own; without the stream-probing phase. Therefore, in performance experiments, we will also run the algorithm with the stream-probing phase switched off. This further simplifies the architecture and the memory used for the stream-probing phase should be reassigned, giving virtually a new intermediate algorithm and we call this mode of operation HYBRIDJOIN. We restrict the cost model and tuning to the full CACHEJOIN algorithm, but we provide experimental results for CACHEJOIN as well as HYBRIDJOIN. The experimental results allow us to study which contribution is made by the disk-probing phase alone (HYBRIDJOIN) and by both phases together. An analysis of our proposed algorithm is presented in the following subsections.

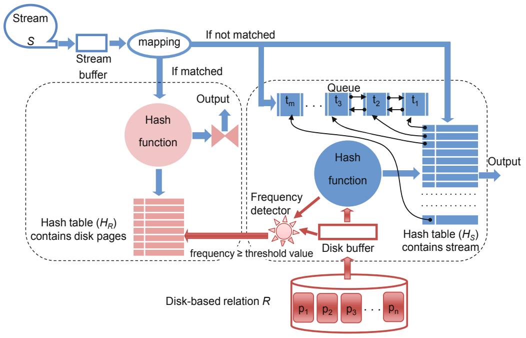


Fig. 1. Data structures and architecture of CACHEJOIN

3.1 Asymptotic Runtime Analysis

In the performance experiments we have focused on a data set that has a skewed distribution but without extreme imbalances. In particular there are no large unused master data areas. We provide a theoretical result for the most general characterization of CACHEJOIN.

We compare the asymptotic runtime of CACHEJOIN with that of MESHJOIN based on the processing time for a stream section. For such a time measure, smaller values are better, as it is known from other runtime discussions. We denote the time needed to process a stream prefix s as *MEJ(s)* for MESHJOIN and as *CHJ(s)* for CACHEJOIN. Every stream prefix represents a binary sequence, and by viewing this binary sequence as a natural number, we can apply asymptotic complexity classes to the functions. Note therefore that the following theorem does not use functions on input lengths, but on concrete inputs encoded as binary strings. We assume that the setup for CACHEJOIN and for MESHJOIN is such that they have the same number h_S of stream tuples in the hash table and in the queue accordingly.

Theorem: *CHJ(s) = O(MEJ(s))*

Proof: To prove the theorem, we have to show CACHEJOIN performs no worse than MESHJOIN. The cost of MESHJOIN is dominated by the number of accesses to R. For asymptotic runtime, random access of disk pages is as fast as sequential access (seek time is a constant factor, see below for a further discussion of the constant factor). For MESHJOIN with its cyclic access pattern for R, every page of R is accessed exactly once after every h_S stream tuples. We have to show that for CACHEJOIN no page is accessed more frequently. For that we look at an arbitrary page p of R at the time it is accessed by CACHEJOIN. The stream tuple at the front of the queue has some position i in the stream. There are h_S stream tuples currently in the hash table, and the first tuple of the stream that is not yet read into the hash table has position $i+h_S$ in the stream. All stream tuples in the hash table are joined against the disk-based master data tuples on p, and all matching tuples are removed from the queue. We now have

Table 1. Notations used in cost estimation of CACHEJOIN

Parameter name	**Symbol**
Number of stream tuples processed in each iteration through H_R	w_N
Number of stream tuples processed in each iteration through H_S	w_S
Stream tuple size *(bytes)*	v_S
Disk page size *(bytes)*	v_P
Size of disk tuple *(bytes)*	v_R
Disk buffer size *(pages)*	k
Disk buffer size *(tuples)*	$d = k\frac{v_P}{v_R}$
Size of H_R *(pages)*	l
Size of H_R *(tuples)*	$h_R = l\frac{v_P}{v_R}$
Size of H_S *(tuples)*	h_S
Disk relation size *(tuples)*	R_t
Memory weight for the hash table	α
Memory weight for the queue	$1-\alpha$
Cost to read k disk pages into the disk buffer *(nano secs)*	$c_{I/O}(k{\cdot}v_P)$
Cost to look-up one tuple in the hash table *(nano secs)*	c_H
Cost to generate the output for one tuple *(nano secs)*	c_O
Cost to remove one tuple from the hash table and the queue *(nano secs)*	c_E
Cost to read one stream tuple into the stream buffer *(nano secs)*	c_S
Cost to append one tuple in the hash table and the queue *(nano secs)*	c_A
Cost to compare the frequency of one disk tuple with the specified threshold value *(nano secs)*	c_F
Total cost for one loop iteration *(secs)*	c_{loop}

to determine the earliest time that p could be loaded again by CACHEJOIN. For p to be loaded again, a stream tuple must be at the front of the queue, and has to match a master data tuple on p. The first stream tuple that can do so is the previously mentioned stream tuple with position $i+h_S$, because all earlier stream tuples that match data on p have been deleted from the queue. This proves the theorem.

In this asymptotic argument we make use of the fact that we abstract from constant factors, particularly the influence of seek time. Our experimental results show that CACHEJOIN does not perform much worse than MESHJOIN even on uniform data.

3.2 Cost Model

In this section we develop the cost model for our proposed CACHEJOIN. The cost model presented here follows the style used for MESHJOIN [9,10]. Equation 1 represents the total memory used by the algorithm (except the stream buffer), and Equation 2 describes the processing cost for each iteration of the algorithm. The notations we used in our cost model are given in Table 1.

Memory Cost. The major portion of the total memory is assigned to the hash table H_S together with the queue while a comparatively much smaller portion is assigned to H_R and the disk buffer. The memory for each component can be calculated as follows:

Memory for disk buffer $(bytes)= k{\cdot}v_P$
Memory for H_R $(bytes){=}l{\cdot}v_P$
Memory for H_S $(bytes){=}\alpha[M-(k+l)v_P]$
Memory for the queue $(bytes) = (1-\alpha)[M-(k+l)v_P]$
By aggregating the above, the total memory M for CACHEJOIN can be calculated as shown in Equation 1.

$$M = (k+l)v_P + \alpha[M-(k+l)v_P] + (1-\alpha)[M-(k+l)v_P] \tag{1}$$

Currently, the memory for the stream buffer in not included because it is small (0.05 MB is sufficient in our experiments).

Processing Cost. In this section we calculate the processing cost for the algorithm. To make it simple we first calculate the processing cost for individual components and then sum these costs to calculate the total processing cost for one iteration.

$c_{I/O}(k \cdot v_P)$ =Cost to read k pages into the disk buffer
$w_N \cdot c_H$ =Cost to look-up w_N tuples in H_R
$d \cdot c_H$ =Cost to look-up disk buffer tuples in H_S
$d \cdot c_F$ =Cost to compare the frequency of all the tuples in disk buffer with the threshold value
$w_N \cdot c_O$ =Cost to generate the output for w_N tuples
$w_S \cdot c_O$ =Cost to generate the output for w_S tuples
$w_N \cdot c_S$ =Cost to read the w_N tuples from the stream buffer
$w_S \cdot c_S$ =Cost to read the w_S tuples from the stream buffer
$w_S \cdot c_A$ =Cost to append w_S tuples into H_S and the queue
$w_S \cdot c_E$ =Cost to delete w_S tuples from H_S and the queue
By aggregating the above costs the total cost of the algorithm for one iteration can be calculated using Equation 2.

$$c_{loop}(secs) = 10^{-9}[c_{I/O}(k{\cdot}v_P)+d(c_H+c_F)+w_S(c_O+c_E+c_S+c_A)+w_N(c_H+c_O+c_S)] \tag{2}$$

Since in c_{loop} seconds the algorithm processes w_N and w_S tuples of the stream S, the service rate μ can be calculated using Equation 3.

$$\mu = \frac{w_N + w_S}{c_{loop}} \tag{3}$$

3.3 Tuning

We now show how the cost model for CACHEJOIN can be used to obtain an optimal tuning of the components, in particular the size for disk buffer and the size of hash table H_R. The algorithm can be tuned to perform optimally using Equation 3 by knowing w_N, w_S and c_{loop}. The value of c_{loop} can be calculated from Equation 2 if we know w_N and w_S.

Mathematical Model for w_N. The main components that directly affect w_N are the size of the master data on disk and the size of H_R. To calculate the effect of both components on w_N we assume that R_t is the total number of tuples in R while h_R is the size of H_R in terms of tuples. We now use our assumption that the stream of updates S has a Zipfian distribution with exponent value one. In this case the matching probability for S in the stream-probing phase can be determined using Equation 4. The denominator is a normalization term to ensure all probabilities sum up to 1, and we use well known approximations for the harmonic series[1].

$$p_N = \frac{\sum_{x=1}^{h_R} \frac{1}{x}}{\sum_{x=1}^{R_t} \frac{1}{x}} = \frac{\ln h_R \gamma + \varepsilon_{h_R}}{\ln R_t \gamma + \varepsilon_{R_t}} \sim \frac{\ln h_R}{\ln R_t} \tag{4}$$

Now using Equation 4 we can determine the constant factors of change in p_N by changing the values of h_R and R_t individually. Let us assume that p_N decreases with constant factor ϕ_N by doubling the value of R_t and increases with constant factor ψ_N by doubling the value of h_R. Knowing these constant factors we are able to calculate the value of w_N. Let us assume the following:

$$p_N = R_t^y h_R^z \tag{5}$$

where y and z are the unknown constants whose values need to be determined.

Determination of y: We know that by doubling R_t, the matching probability p_N decreases by a constant factor ϕ_N therefore, Equation 5 becomes:

$$\phi_N p_N = (2R_t)^y h_R^z$$

Dividing the above equation by Equation 5 we get $2^y = \phi_N$ and therefore, $y = \log_2(\phi_N)$.

Determination of z: Similarly we also know that by doubling h_R the matching probability p_N increases by a constant factor ψ_N therefore, Equation 5 can be written as:

$$\psi_N p_N = R_t^y (2h_R)^z$$

By dividing the above equation by Equation 5 we get $2^z = \psi_N$ and therefore, $z = \log_2(\psi_N)$. After substituting the values of constants y and z into Equation 5 we get:

$$p_N = R_t^{\log_2(\phi_N)} h_R^{\log_2(\psi_N)}$$

Now if S_n is the total number of stream tuples that are processed (through both phases) in n outer iterations then w_N can be calculated using Equation 6.

$$w_N = \frac{(R_t^{\log_2(\phi_N)} h_R^{\log_2(\psi_N)}) S_n}{n} \tag{6}$$

Mathematical Model for w_S. The second phase of the CACHEJOIN algorithm deals with the rest of R. This part is called R', with $R' = R - h_R$. The algorithm reads R' in segments. The size of each segment is equal to the size of the disk buffer d. In each iteration the algorithm reads one segment of R' using an index on the join attribute and loads it into the disk buffer. Since we assume a skewed distribution, the matching probability is not equal, but decreases in the tail of the distribution.

We calculate the matching probability for each segment by summing over the discrete Zipfian distribution separately and then aggregating all of them as shown below.

$$\sum_{x=h_R+1}^{h_R+d} \frac{1}{x} + \sum_{x=h_R+d+1}^{h_R+2d} \frac{1}{x} + \sum_{x=h_R+2d+1}^{h_R+3d} \frac{1}{x} + \cdots + \sum_{x=h_R+(n-1)d+1}^{h_R+nd} \frac{1}{x}$$

We simplify this to:

$$\sum_{x=h_R+1}^{h_R+nd} \frac{1}{x} \Rightarrow \sum_{x=h_R+1}^{R_t} \frac{1}{x}$$

From this we can obtain the average matching probability $\overline{p}_S$ in the disk-probing phase, which we need for calculating w_S. Let N be the total number of segments in R'. In the denominator, we have to use the same normalization term as in Equation 4, and we again use the summation formula [1]:

$$\overline{p}_S = \frac{\sum_{x=h_R+1}^{R_t} \frac{1}{x}}{N \sum_{x=1}^{R_t} \frac{1}{x}} \sim \frac{\ln(R_t) - \ln(h_R)}{N(\ln(R_t) + \gamma)} \tag{7}$$

To determine the effects of d, h_R and R_t on $\overline{p}_S$, a similar argument can be used as in the case of w_N. Let's suppose we double d in Equation 7, then N will be halved and the value of $\overline{p}_S$ increases by a constant factor of θ_S. Similarly, if we double h_R or R_t respectively, then the value of $\overline{p}_S$ decreases by some constant factor of ψ_S or ϕ_S respectively. Using a similar argument for w_N, we get:

$$\overline{p}_S = d^x h_R^y R_t^z \tag{8}$$

The values for the constants x, y and z in this case will be $x = \log_2(\theta_S)$, $y = \log_2(\psi_S)$ and $z = \log_2(\phi_S)$ respectively. Therefore by replacing the values with constants Equation 8 will become.

$$\overline{p}_S = d^{\log_2(\theta_S)} h_R^{\log_2(\psi_S)} R_t^{\log_2(\phi_S)}$$

Now if h_S are the number of stream tuples stored in the hash table then the average value for w_S can be calculated using Equation 9.

$$w_S(\text{average}) = d^{\log_2(\theta_S)} h_R^{\log_2(\psi_S)} R_t^{\log_2(\phi_S)} h_S \tag{9}$$

Once the values of w_N and w_S are determined, the algorithm can be tuned using Equation 3.

Table 2. Data specification

Parameter	**value**
Size of each disk tuple	120 *bytes* (similar to MESHJOIN)
Size of each stream tuple	20 *bytes* (similar to MESHJOIN)
Size of each node in the queue	12 *bytes*
Data set	based on Zipf's law (exponent varies from 0 to 1)

4 Experimental Evaluation

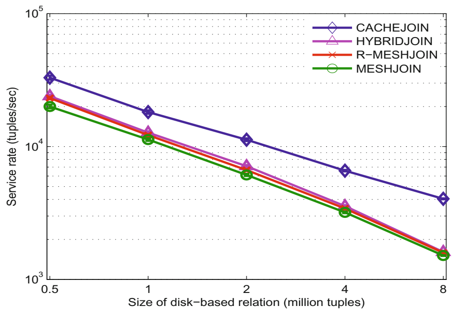
4.1 Experimental Setup Used in all Experiments

We run our experiments on *Pentium-IV 2×2.13GHz*. We implemented our experiments in *Java* using the *Eclipse IDE 3.3.1.1*. Measurements with *Apache* plugins and *nanoTime()* from *Java API*. The relation R is stored on disk using a *MySQL 5.0* database, fetch size for *ResultSet* is set equal to the disk buffer size. Synthetic data, stream data is generated with a Zipfian distribution of the foreign key with varying exponent values. The detailed specifications of the data set that we used for analysis are shown in Table 2.

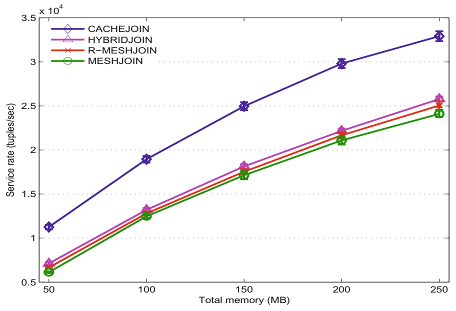
4.2 Comparison between CACHEJOIN and MESHJOIN

In this section we compare CACHEJOIN and MESHJOIN. We included two other algorithms mentioned earlier. As explained before, CACHEJOIN becomes HYBRIDJOIN if its stream-probing phase is switched-off. By including HYBRIDJOIN we can understand better, where the difference in performance comes from. We also included R-MESHJOIN [8], which is a slight optimization of MESHJOIN. We have identified three parameters, for which we want to understand the behavior of the algorithms in the case of different memory settings. The three parameters are: the size of the master data table R, the total memory available, and the exponent of the Zipfian distribution. For the sake of brevity, we restrict the discussion for each parameter to a one-dimensional variation, i.e. we vary one parameter at a time.

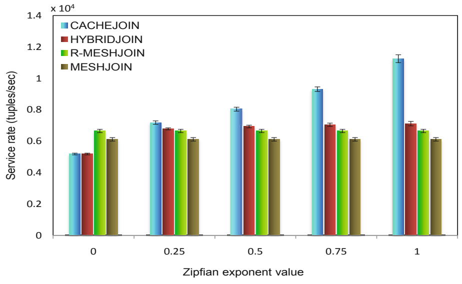
Performance Comparisons for Varying Size of *R*. The problem that gave rise to CACHEJOIN was that the performance of MESHJOIN decreases inversely proportionally with the size of R. Therefore this performance evaluation is of particular interest. For the total allocated memory, we use the value that has been used by the MESHJOIN authors in their experiments. For the exponent of the Zipfian distribution, we choose the exponent 1 as a natural type of skew that is observed frequently in practice. We choose the discrete sizes of the parameter, the size of disk-based relation R, from a simple geometric progression. The performance results are shown in Figure 2(a). In our experiments, the performance of CACHEJOIN is better for all settings of R compared to the other algorithms.



(a) Size of relation on disk varies



(b) Size of allocated memory varies



(c) Value of Zipfian exponent varies

Fig. 2. Performance comparisons of CACHEJOIN with related join algorithms

Performance Comparisons for Different Memory Budgets. In our second experiment we test the performance of all algorithms using different memory budgets while keeping the size of R fixed (*2 million tuples*) and choosing the Zipfian exponent of 1. Figure 2(b) presents the comparisons of all approaches. In our experiments, CACHEJOIN performs better for all memory budgets.

Performance Comparisons while Varying Skew in Stream S. We also test the performance of CACHEJOIN with the other related algorithms while varying the skew in input stream S. To vary the skew, we vary the value of the Zipfian exponent. In our experiments we allow it to range from 0 to 1. At 0 the input stream S is uniform while at 1 the stream has a larger skew. The results presented in Figure 2(c) shows that CACHEJOIN performs better than the other approaches even for only moderately skewed data. Also this improvement becomes more pronounced for increasing exponent values. We did not present data for exponents larger than 1, which would imply short tails. It is clear, that for such short tails the trend continues. However, for completely uniform data, CACHEJOIN performs worse than MESHJOIN by a constant factor of 1.2, due to the influence of seek time.

5 Conclusions

In this paper we propose a new semi-stream-based join called CACHEJOIN and we compare it with MESHJOIN, an earlier well-known semi-stream-based join. CACHEJOIN is designed to make use of skewed, non-uniformly distributed data as found in real-world applications. In particular we consider a Zipfian distribution of foreign keys in the stream data. Contrary to MESHJOIN, CACHEJOIN stores these most frequently accessed tuples of R permanently in memory. We have provided a cost model that can be used to tune the algorithm and validated it with experiments. We have provided experimental data showing an improvement of CACHEJOIN over the MESHJOIN algorithm.

Source URL: We have provided an open-source implementation of both algorithms used in the experiments, MESHJOIN as well as CACHEJOIN, that can be used for further analysis, at the following URL.
`https://www.cs.auckland.ac.nz/research/groups/serg/cj/`

References

1. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Dover, New York, ninth Dover printing, tenth GPO printing edition (1964)
2. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. Hyperion (2006)
3. Bornea, M.A., Deligiannakis, A., Kotidis, Y., Vassalos, V.: Semi-streamed index join for near-real time execution of ETL transformations. In: IEEE 27th International Conference on Data Engineering, ICDE 2011, pp. 159–170 (April 2011)
4. Chakraborty, A., Singh, A.: A partition-based approach to support streaming updates over persistent data in an active datawarehouse. In: IPDPS 2009: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, pp. 1–11. IEEE Computer Society, Washington, DC (2009)
5. Karakasidis, A., Vassiliadis, P., Pitoura, E.: ETL queues for active data warehousing. In: IQIS 2005: Proceedings of the 2nd International Workshop on Information Quality in Information Systems, pp. 28–39. ACM, New York (2005)
6. Knuth, D.E.: The art of computer programming, vol. 3: sorting and searching, 2nd edn. Addison Wesley Longman Publishing Co., Inc., Redwood City (1998)
7. Asif Naeem, M., Dobbie, G., Weber, G.: An event-based near real-time data integration architecture. In: EDOCW 2008: Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops, pp. 401–404. IEEE Computer Society, Washington, DC (2008)
8. Asif Naeem, M., Dobbie, G., Weber, G., Alam, S.: R-MESHJOIN for near-real-time data warehousing. In: DOLAP 2010: Proceedings of the ACM 13th International Workshop on Data Warehousing and OLAP. ACM, Toronto (2010)
9. Polyzotis, N., Skiadopoulos, S., Vassiliadis, P., Simitsis, A., Frantzell, N.E.: Supporting streaming updates in an active data warehouse. In: ICDE 2007: Proceedings of the 23rd International Conference on Data Engineering, Istanbul, Turkey, pp. 476–485 (2007)
10. Polyzotis, N., Skiadopoulos, S., Vassiliadis, P., Simitsis, A., Frantzell, N.: Meshing streaming updates with persistent data in an active data warehouse. IEEE Trans. on Knowl. and Data Eng. 20(7), 976–991 (2008)

MIRABEL DW: Managing Complex Energy Data in a Smart Grid

Laurynas Siksnys, Christian Thomsen, and Torben Bach Pedersen

Department of Computer Science,
Aalborg University
{siksnys,chr,tbp}@cs.aau.dk

Abstract. In the MIRABEL project, a data management system for a *smart grid* is developed to enable smarter scheduling of energy consumption such that, e.g., charging of car batteries is done during night when there is an overcapacity of *green energy* from windmills etc. Energy can then be requested by means of *flex-offers* which define flexibility with respect to time, amount, and/or price. In this paper, we describe MIRABEL DW, a data warehouse (DW) for the management of the large amounts of complex energy data in MIRABEL. We present a unified schema that can manage data both at the level of the entire electricity network and at the level of individual nodes, such as a single consumer node. The schema has a number of complexities compared to typical DW schemas. These include *facts about facts* and *composed non-atomic facts* and unified handling of different kinds of flex-offers and time series. We also discuss alternative data modeling strategies and present typical queries from the energy domain and a performance study.

1 Introduction

More and more *green energy* is being produced by renewable energy sources (RES) such as windmills. It is, however, not possible to store larger amounts of energy and use it later. Therefore, there often is an unused capacity, e.g., during nights when most consumers sleep, but not enough green energy during day hours when most consumers are active. The EU FP7 project MIRABEL (Micro-Request-Based Aggregation, Forecasting, Scheduling of Energy Demand. Supply and Distribution) [11] addresses this challenge by proposing a "data-driven" solution for balancing supply and demand utilizing their flexibilities. Flexible demand such as for dishwashers and charging an electric vehicle can often be shifted to a time when green energy is available. Non-flexible demand such as lights, TV, or cooking stoves must still be satisfied at demand-time. In the MIRABEL-settings, a consumer offers a so-called *flex-offer* [2,14] for every intent of flexible energy demand. The flex-offer must describe when and how much energy is needed and how flexible the demand is in time and amount. Likewise, a producer can offer a flex-offer for every intent of energy supply. The different flex-offers can then be accepted (or rejected if they cannot be fulfilled) and scheduled for execution at a given time. There will be extremely large quantities of such flex-offers and they cannot be scheduled individually. Instead flex-offers are *aggregated* into larger flex-offers which become scheduled and then *disaggregated* into the smaller flex-offers again [14].

To enable this, there will be smart *nodes* at both consumer sites and producer sites in the electricity grid which we denote a *smart grid*.

There is a a strong need for efficient data management in these nodes. In this paper, we present *MIRABEL DW* which is a data warehouse (DW) for the management of large amounts of complex energy data in the MIRABEL project. This paper is the first to present a DW schema for the important domain of energy data. The schema can represent different "actors" in different "roles" as defined by the "Harmonised Electricity Market Role Model" [4] as well as (individual and aggregated) flex-offers, and time series. In the future, the managed data is to be distributed over millions of nodes [2] in non-traditional ways. In the paper, we focus on a DW on a single node, but present a unified schema that can manage data both at the level of the entire electricity network and at the level of individual nodes, such as a single consumer node. Compared to typical DW schemas, the schema has a number of complexities which we discuss in the paper. These include *facts about facts* and *composed non-atomic facts* and unified handling of different kinds of flex-offers and time series. We also discuss alternative data modeling strategies that use denormalization and arrays, respectively. Further, we present typical queries from the energy domain and a performance study that compares the described schemas with the denormalized and array-based alternatives.

The rest of the paper is organized as follows: Our representations of flex-offers, time series and actors are presented in Sections 2, 3, and 4, respectively. These parts together form the full schema which is presented in Section 5. Examples of analytical queries on the schema are given in Section 6. A performance study is given in Section 7. Previous work related to this is presented in Section 8 before the concluding remarks and pointers to future work which are given in Section 9.

2 Modeling of Flex-Offers

In this and the following two sections, we first present the data model we use in MIRABEL DW. Then we discuss the non-standard and advanced techniques that are applied in the modeling.

2.1 Data Model

To represent MIRABEL's flex-offers (both aggregated and non-aggregated) is an essential task for MIRABEL DW. This is done by means of the tables shown in Fig. 1. We first describe the dimensions (which are recognized by the prefix D_ in their table names) and then the fact tables (recognized by the prefix F_ in their names). All dimension tables have surrogate keys with names ending with Id. The possible states for a flex-offer (such as "offered", "accepted", and "rejected") are represented in the dimension D_flexEnergyState. A flex-offer has its state for a certain reason (for example, a flex-offer becomes rejected if the offered price is too high). The possible reasons are represented in the dimension D_flexEnergyStateReason. As we expect few generic reason categories (e.g, "Price too high") and many more specific reason descriptions (e.g. "Price (499.50 euros) too high") to exist, we have columns for both the generic categories and the specific reasons such that a hierarchy exists. In MIRABEL DW, we represent time by discretized time intervals. This is done by D_timeInterval which represents

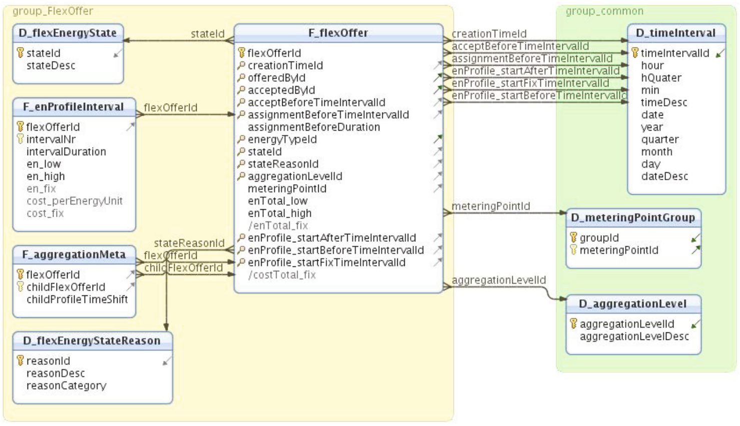


Fig. 1. Tables for representing flex-offers

15 minutes intervals (for now; other interval lengths can be chosen if needed). Flex-offers are always related to at least one metering point (at the location where the energy is to be consumed or produced), but if a flex-offer is aggregated, it will be associated with many metering points. To capture this, D_meteringPointGroup is used as bridge table [7] between the fact table and D_meteringPoint which represents the individual metering points. To represent the aggregation level of a flex-offer, D_aggregationLevel is used.

The fact table F_flexOffer holds flex-offer facts. It references all the previously described dimension tables. There are six foreign keys to D_timeInterval to represent different times such as when the flex-offer was created and when it at the latest has to be assigned etc. These foreign keys thus all represent an absolute time. There is also an attribute assignmentBeforeDuration which holds a time span telling how long before the actual execution time the assignment must take place.

Further, F_flexOffer references D_legalEntityRole (explained later) twice to represent who offered and accepted the flex-offer, respectively. Only the current information about a flex-offer is held; if a flex-offer is modified, the old fact is overwritten. There are measures to hold the lowest and highest amount of energy required by the flex-offer as well as a measure to hold the “fixed” amount of energy that becomes accepted. Further, a measure holds the total cost of the fix. Finally, each represented flex-offer is given a unique identifier in the attribute flexOfferId which technically is a degenerate dimension.

Information about the profile intervals of flex-offers is represented in the fact table F_enProfileInterval. This fact table only has a single foreign key which references the unique flexOfferId in F_flexOffer. The imported value together with a sequential intervalNr forms the primary key for F_enProfileInterval. The reason for this design is that a single flex-offer can have many profile intervals. For each represented profile interval, there is a duration specifying how many time units the profile interval spans over, and both the lowest and highest amount of energy needed in this interval. When the flex-offer becomes fixed, the actual amount of energy in the interval and the price for this energy also becomes represented. An alternative to this design would be to represent the measures of F_enProfileInterval in *arrays* in F_flexOffer such that all data about a given flex-offer would be represented in a single fact. Yet another alternative would be to represent all attributes of F_enProfileInterval in F_flexOffer, i.e., denormalize the data and have one (wide) fact in F_flexOffer for each profile interval. (For space reasons, we do not show the alternative schemas in figures.)

As flex-offers can be aggregated into larger flex-offers, we also introduce the table F_aggregationMeta which references F_flexOffer twice to point to the aggregating "parent flex-offer" and the smaller "child flex-offer" which has been aggregated, respectively. Profiles of each child flex-offer can be shifted relatively to the profile start of the parent flex-offer when aggregating child flex-offers into the parent. Therefore, for every child flex-offer, the childProfileTimeShift attribute indicates the amount of time units the profiles of the child flex-offer has been shifted in the aggregated flex-offer. This information is used in the disaggregation.

2.2 Modeling Challenges

The fact table F_flexOffer is the central fact table for representation of flex-offers. It is, however, also used as a dimension table in the sense that each fact has a unique ID such that F_enProfileInterval and F_aggregationMeta can reference F_flexOffer and in effect store *facts about facts*. Considering F_flexOffer and F_enProfileInterval, it can even be discussed *what* a fact is. An energy profile interval (in this context) always belongs to a flex-offer and any meaningful flex-offer has an energy profile interval (a flex-offer for zero consumption/production at an undefined point in time is hardly interesting). It could be argued that a single fact is represented by a single row in F_flexOffer and many rows in F_enProfileInterval. Unlike traditional DW schemas, we thus have non-atomic *composed* facts. As pointed out above, we could alternatively have modeled this by using arrays in F_flexOffer to hold the measures that currently are represented in F_enProfileInterval. This would, however, make it more cumbersome to compare different measures (e.g., en_low with the minimum energy requirement to en_fix with the assigned energy) as the interval position currently represented by intervalNr only would be implicitly represented by the position in the array. The denormalized variant (with a fact in F_flexOffer for each profile interval) would increase redundancy dramatically.

Another interesting aspect of MIRABEL DW is how it represents facts for both non-aggregated and aggregated flex-offers in a unified way. The aggregation is unlike traditional aggregation since the parent flex-offer contains other flex-offers that can be shifted within the parent flex-offer. We call the contained flex-offers *shiftable child facts*.

3 Modeling of Time Series

3.1 Data Model

In MIRABEL DW, time series are represented by means of the tables shown in Fig. 2. It is necessary to be able to represent time series of various types, for now energy, power, and price. To represent these general classes, we use the D_typeClass dimension table. Apart from its surrogate key, it has the attribute typeClassDesc which holds

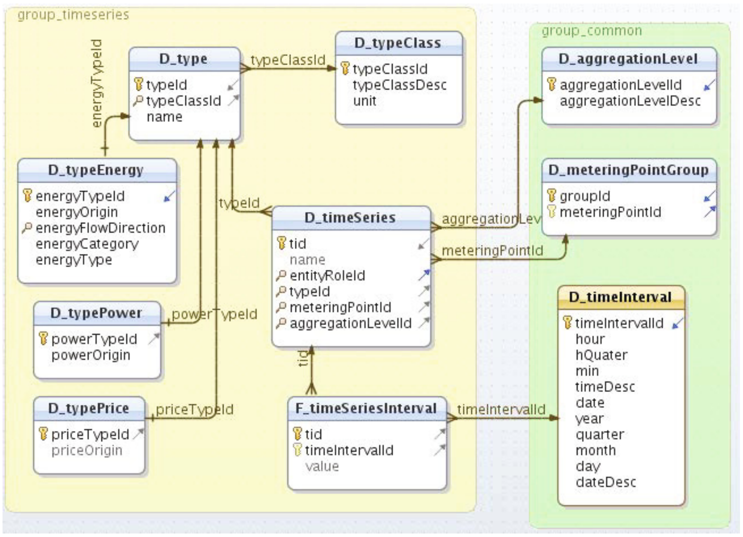


Fig. 2. Tables for representing time series

a textual description of the time series type (such as “Energy”) and the attribute unit which holds the unit of measurements (such as “kWh”). Instances of the general types are represented in the table D_type. For example, an instance of the “Energy” class is “Energy-Metered-Production-RES-Wind”. D_type references D_typeClass to represent the hierarchy between types and type classes. For different types of time series, it is, however, necessary to store different information. Therefore, we introduce the tables D_typeEnergy, D_typePower, and D_typePrice to hold the attributes that are relevant for the different types. These tables supplement, but cannot replace, D_type. The reason is that we need a single table to reference from D_timeSeries to represent the type of the time series in question. Thus D_type is referenced from D_timeSeries, but the special attributes for an energy time series are represented in D_typeEnergy. The latter table has columns to describe the origin of the time series (e.g. “Metered” or “Forecasted”), the flow direction (i.e., if it is production or consumption), the category (e.g., energy from renewable energy sources), and the type of energy (e.g. “Wind”). The design is likely to evolve in the future. For example, there is a traditional hierarchy where types roll up into categories that roll up into flow directions. A more advanced hierarchy is, however, needed to represent hybrid energy types like “At least 90% energy from renewable energy sources and the rest produced from coal”.

D_timeSeries holds a single entry for an entire time series. For each represented time series, there is a unique ID tid and a name may be given. Further, D_timeSeries references D_type (as previously described), D_aggregationLevel to represent the level of aggregation of the time series, and D_meteringPointGroup to represent which meters the time series describes. Thus, D_timeSeries is mainly used to relate different dimension values that describe the represented time series. The values of the time series are, however, represented in the fact table F_timeSeriesInterval. This table references

D_timeSeries to identify the time series a value belongs to and D_timeInterval to identify the time instant when the value occured. Finally, the table holds the value itself as the measure. A fact thus exists for each value in each time series. It can, however, also be argued that a fact consists of what it represented in F_timeSeriesInterval *and* what is represented in D_timeSeries which – apart from a possible name – only points out to other dimensions.

3.2 Modeling Challenges

Similarly to the represention of flex-offers, our representation of time series also leads to compound facts where one fact can be considered to be made up of parts in different tables (D_timeSeries and F_timeSeriesInterval). Actually, an alternative design is to merge F_timeSeriesInterval into D_timeSeries such that the values instead are represented in an array, meaning that a single time interval (and all its values) only would result in one fact. Yet another alternative is to merge D_timeSeries and F_timeSeriesInterval and have a row for each value in a time series. There are thus different possible ways to represent the complex sequence-facts arising from time series. We choose the model in Fig. 2 since it both reduces complexity (compared to the first alternative where two arrays must be processed to find the value for a given time instant) and redundancy (compared to the second alternative where there is very wide fact for each value in the time series).

In our modeling of time series, the schema is neither a traditional star schema nor a snowflake schema. One reason for this is of course the compound facts discussed above. Another reason is the support for different types of time series for which different attributes are needed. We have different tables that reference D_type which also is the dimension table referenced from the fact table. Consider for example D_typeEnergy which represents attributes that are relevant for energy time series. An alternative design would be to join all these D_type* tables into one dimension table, but for every dimension member many attribute values would then be NULL.

4 Modeling of Different Actors and Market Areas

4.1 Data Model

Many different entities are involved in different roles in energy trading and network operation. We represent the needed actors from the "Harmonised Electricity Market Role Model" [4] by means of the tables in Fig. 3.

The table D_role represents roles such as "Producer" and "Consumer". A role can belong to another parent role and this is captured by a self-reference. For example, the parent role of both "Producer" and "Consumer" is "Party Connected To Grid". Legal entities are represented by D_legalEntity. To capture when a certain legal entity plays a certain role (a single legal entity can play several roles), we use D_legalEntityRole. This table references both D_role and D_legalEntity. Further, it has an attribute to hold a unique ID for a given legal entity playing a given role. We include this ID as it makes it easy to point to a legal entity in a certain role. We do exactly that from a

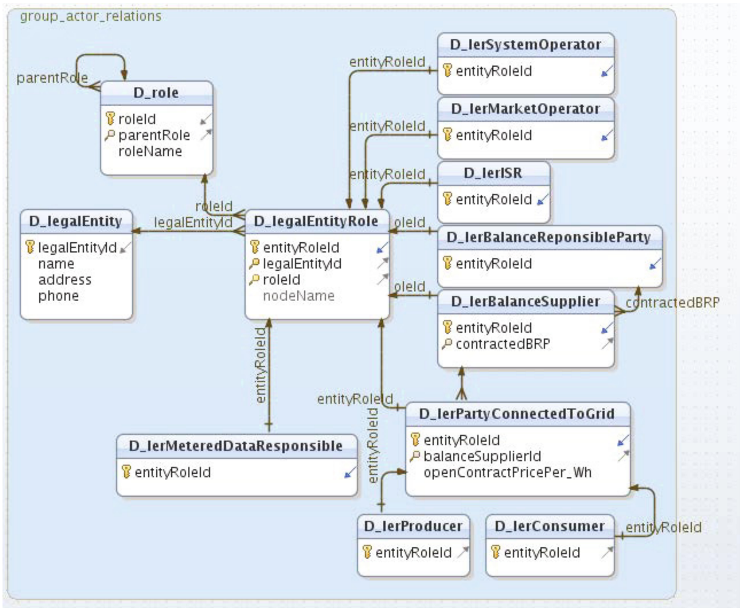


Fig. 3. Tables for representing different actors/roles

number of tables as shown in Fig. 3. For each role, there is a specialized table that (directly or indirectly through another table) references D_legalEntityRole. Some of them, like D_lerSystemOperator, are simple and do only have one attribute which is a reference to this ID. The specialized table can be referenced and it is then explicit what kind of role is referenced. For example, the table D_lerSystemOperator is referenced from D_marketBalanceArea as shown in Fig. 5. A slightly more complex example is D_lerPartyConnectedToGrid which references D_legalEntityRole and also D_lerBalanceSupplier to represent that a party connected to the grid always is so through a balance supplier. Further, D_lerPartyConnectedToGrid is itself referenced from its specializations, D_lerProducer and D_lerConsumer.

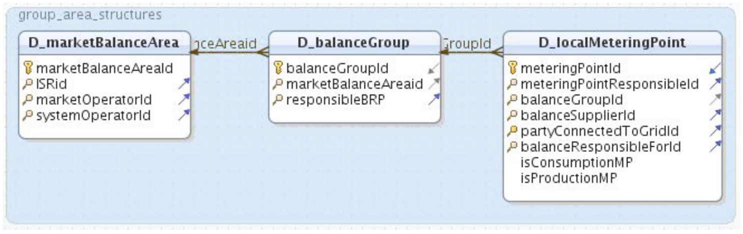


Fig. 4. Tables for representing market areas

Finally, we have tables to represent market areas as shown in Fig. 4. D_localMeteringPoint represents the meters that are connected to the grid. Such meters are installed both at the producer and consumer sites. D_localMeteringPoint references four different specializations of D_legalEntityRole. Further, it references D_balanceGroup which in turn references D_marketBalanceArea which hierarchically groups metering points.

4.2 Modeling Challenges

To the best of our knowledge, this is the first paper to describe a DW for the complex concepts of actors and roles in the "Harmonised Electricity Market Role Model" [4]. Our model captures both how legal entities can play different roles and how roles can be parts of other roles. This is captured by the tables D_legalEntity, D_role, and D_legalEntityRole. In addition to these tables, a (narrow) table has been added for each role a legal entity can play (see the D_ler* tables). It is then possible to represent attributes that are only relevant for certain roles such as done for D_lerBalanceSupplier. Further, when foreign keys reference these tables (instead of just referencing D_legalEntityRole), it is explicit what kind of role playing is referenced and it helps to avoid mistakes where, e.g., a balance supplier is referenced where a balance responsible party actually should have been referenced. We note that if no special attributes must be stored for the different roles, then instead of storing the D_ler*'s as physical tables, they can be views selecting from D_legalEntityRole. This reduces the risk of mistakes further and makes maintenance of them automatic.

5 The Full Schema

To summarize the previous descriptions, the full schema for MIRABEL DW is shown in Fig. 5. The schema can capture the (needed) roles from the Harmonised Model [4] as well as the "actor configurations" where different actors play different roles. The schema also includes specializations of legal entities. Further, the schema can capture different kinds of time series as complex sequence facts. The schema is thus general enough to hold all the data that is needed in the MIRABEL project. It should, however, be noted that no single node is intended to hold all data. Instead, a node should only hold data that is relevant for the site where it is installed. For an end-consumer this would typically be her own non-aggregated flex-offers and time series about metered energy. For a balance responsible party buying electricity on the market and selling it to end-consumers, it would include both aggregated and non-aggregated flex-offers, forecasted and metered time series, and market areas. The data will thus be distributed accordingly to the roles played by the owners of the nodes. The data will also be at different aggregation levels such that some nodes have detailed data while others have more aggregated data. For example, will a consumer know the details of her flex-offers, i.e., when she has requested energy and how much. For a balance responsible party, the individual non-aggregated flex-offers and end-users generating may not be known, but the aggregated information will be known, e.g., that x MWhs must be produced in a given time interval. Note that the different nodes can use the same schema.

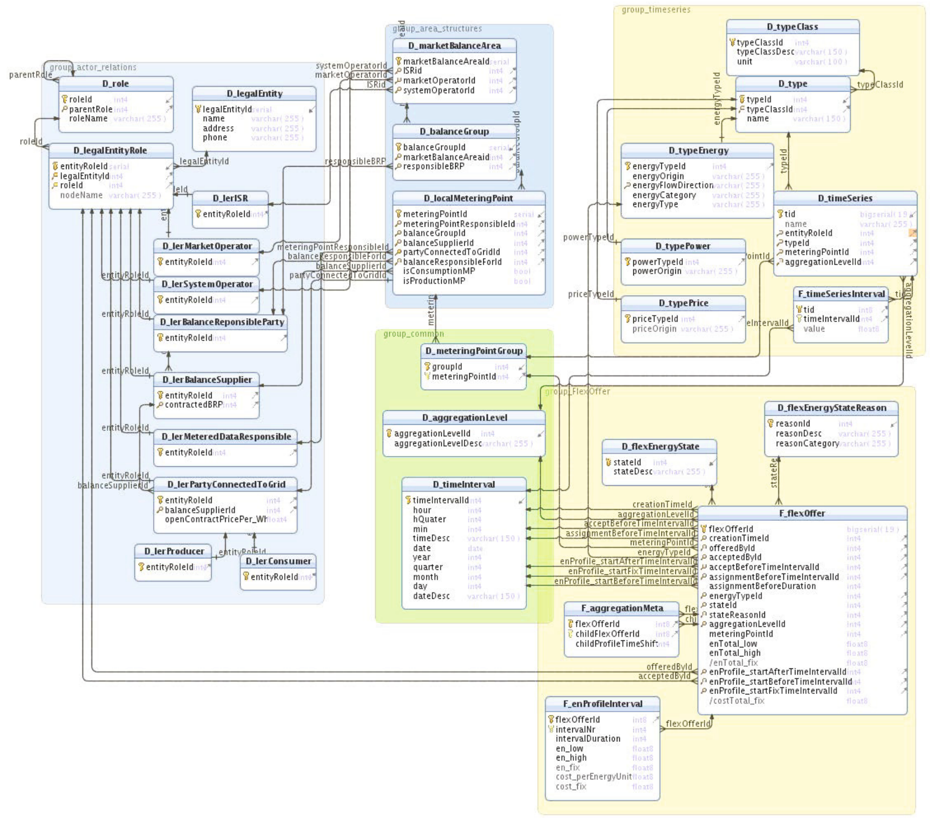


Fig. 5. The full schema for MIRABEL DW

6 Queries

In this section, we give examples of interesting queries on data in MIRABEL DW. We first focus on queries on flex-offers and then on time series.

6.1 Queries on Flex-Offers

The first example, Q1, considers the flexibility in flex-offers, both with respect to time and amount of energy.

```
Q1:  SELECT AVG((enProfile_startBeforeTimeIntervalId -
                 enProfile_startAfterTimeIntervalId) *
                (SELECT SUM((en_high - en_low) * intervalDuration)
                 FROM F_enProfileInterval i
                 WHERE i.flexOfferId = f.flexOfferId)
                )
     FROM F_flexOffer f;
```

The query uses the flexibility with respect to time, i.e., the difference between when the flex-offer at the latest *has* to be executed and when it at the earliest *can* be scheduled.

We assume that time interval IDs are assigned sequentially and thus use the difference between the IDs of the time intervals to find the flexibility. This flexibility is multiplied with the SUM of the energy flexibility in each profile interval. The energy flexibility in a profile interval is found as the length of the profile interval multiplied with the difference between the maximally required amount of energy and the minimally required amount of energy. Finally, the shown query considers the average of the combined flexibility for all flex-offers. The query is an example of a non-traditional kind of aggregation. If we consider a graph showing the relative start and end times for profile intervals on the X axis and the minimal and maximal energy amounts on the Y axis, the query `Q1` finds the *area of energy flexibility* for all flex-offers and multiplies these with the length of their time flexibilities before the entire average is found. This number is primarily of interest *before* the scheduling gets done and a high number indicates much freedom in the scheduling while a low number shows that the considered flex-offers are not very flexible.

The next example, `Q2`, is of interest *after* the scheduling and gives the total amount of scheduled energy. This is a simple query which, however, must read data from many rows in a realistic setting (the DBMS we use does currently not support materialized views).

```
Q2: SELECT SUM(en_fix)
    FROM F_enProfileInterval;
```

`Q3` is a more complex query to apply after scheduling has taken place. It builds a time series that for each time interval ID shows the amount of fixed energy.

```
Q3: SELECT timeIntervalId, SUM(en_fix_part)
    FROM (SELECT en_fix_part, ROW_NUMBER() OVER (PARTITION BY i.flexOfferId
                 ORDER BY intervalNr) - 1 + f.enProfile_startFixTimeIntervalId
                 AS timeIntervalId
          FROM (SELECT flexOfferId, intervalNr, en_fix / intervalDuration
                       AS en_fix_part, generate_series(1, intervalDuration)
                FROM F_enProfileInterval
                WHERE en_fix IS NOT NULL
               ) i, F_flexOffer f, D_flexEnergyState s
          WHERE i.flexOfferId = f.flexOfferId AND f.stateId = s.stateId
                AND s.stateDesc = 'Assigned'
         ) AS subquery
    GROUP BY timeIntervalId
    ORDER BY timeIntervalId;
```

The query computes the IDs of the time intervals where a flex-offer's profile intervals are executed. But a profile interval has a duration (in intervalDuration) which defines how many time intervals the profile interval spans. Therefore, it is necessary to (evenly) distribute the profile intervals' energy amounts over one or more time intervals. To do this, one "part" row is generated for each time interval a profile interval covers by means of generate_series. This happens in the innermost SELECT. The result of this is used by the second SELECT which also uses the SQL window function ROW_NUMBER to enumerate the rows in each partition where a partition consists of the part rows for a given flex offer and is ordered by the interval numbers. Thus, the resulting row number corresponds to the number of time intervals between the assigned start time for the entire flex offer and the part represented by the row (we subtract 1 since ROW_NUMBER counts from 1). When we add enProfile_startFixTimeInterval for the flex-offer, we get the ID of tje absolute time interval when the part executes. Finally, the outermost SELECT aggregates the sums of fixed energy amounts over all parts belonging to a given time interval.

6.2 Queries on Time Series

`Q4` is query that finds the balance, i.e., the difference between produced and consumed energy, for a 24 hours period.

```
Q4: SELECT date, timeDesc,
           SUM(CASE energyFlowDirection WHEN 'Production' THEN value
                                        ELSE 0 END) AS production,
           SUM(CASE energyFlowDirection WHEN 'Consumption' THEN value
                                        ELSE 0 END) AS consumption
           SUM(CASE energyFlowDirection WHEN 'Production' THEN value
                                        WHEN 'Consumption' THEN -1 * value
                                        ELSE 0 END) AS balance
    FROM F_timeSeriesInterval f, D_timeSeries ts, D_type ty,
         D_typeEnergy te, D_timeInterval ti
    WHERE f.tid = ts.tid AND ts.typeId = ty.typeId AND te.energyTypeId =
          ty.typeId  AND ti.timeIntervalId = f.timeIntervalId AND
          te.energyOrigin = 'Metered' AND ti.date = '2011-06-01'
    GROUP BY ti.timeIntervalId
    ORDER BY ti.timeIntervalId;
```

The query `Q4` is an example where we use the special attributes that only apply to some time series. In this example, we consider consumed and produced energy and we thus use energyFlowDirection and energyOrigin which only exist for energy time series. The query sums the production values, consumption values, and the difference between them for each time interval that belongs to a given date.

Our last example, `Q5`, is a query to find those time series where the average energy usage grouped on hours exceeds the average energy usage for the hour with 25% or more at least 10 times.

```
Q5: WITH indavguse AS (
         SELECT tid, hour, COUNT(value) AS indcnt, AVG(value) AS indavg
         FROM F_timeSeriesInterval NATURAL JOIN D_timeInterval
         GROUP BY tid, hour
    ),
    totavguse AS (
         SELECT hour, SUM(indcnt * indavg) / SUM(indcnt) AS totavg
         FROM indavguse
         GROUP BY hour
    ),
    overuse AS (
       SELECT tid, t.hour, indavg, totavg,
              COUNT(*) OVER (PARTITION BY tid) AS cnt
       FROM totavguse t, indavguse i
       WHERE t.hour = i.hour AND indavg >= 1.25 * totavg
    )
    SELECT tid, cnt, hour, indavg, totavg
    FROM overuse
    WHERE cnt > 10
    ORDER BY tid, hour;
```

The query has Common Table Expressions (CTEs) in the WITH part. In the first CTE, `indavguse`, we compute a (temporary) table with the average hourly energy usage for each time series. The result is used again to compute the second CTE, `totavguse`, where we find the average energy use per hour among all time series (we could join F_timeSeriesInterval and D_timeInterval again, but it is faster to reuse the result of the previously computed CTE). In the third CTE, `overuse`, we join the the results of the two previous CTEs to find the IDs of time series and the hours from `indavguse` where the consumption is at least 25% higher than the general hourly average consumption found in `totavguse`. Further, we use COUNT as a window function to count how many such hours we find for a given time series. Finally, we select the ID of the time series, the count of hours with an average energy usage at least 25% higher than the average, and the consumption in the last SELECT clause.

7 Performance Study

In this section, we compare the performance of the queries from the previous section. We consider them as they are on the described schema (called "MDW") and in addition, we consider alternative queries on the described schema alternatives with denormalization and arrays, respectively. In the denormalized variant, F_flexOffer and F_enProfileInterval are joined and so are F_timeSeriesInterval and D_timeSeries (however, with the name varchar attribute replaced by an integer to make it a typical fact table). In the array variant, the same tables are joined, but now grouped on all dimension references and with measures aggregated into arrays. For the tests, we use a real life data set with consumption data from 963 customers (the data originates from the MeRegio project [10]) and we synthetically generate flex-offers based on this data set. This gives rise to 963 (energy consumption) time series with 32.1 million time series values, and 3,1 million flex-offers. We test the performance on a Linux server with two Quad Core 1.86GHz Intel Xeon CPUs, 16 GB RAM, 4 SATA 7200RPM disks (with one dedicated to the DBMS). The DBMS is PostgreSQL 9.1 [12] and has the parameter shared_buffers set to 4GB, temp_buffers to 128MB, and work_mem to 96MB. All tables are "fully vacuumed" such that their disk representations only take up the needed space and do not occupy unused space. Further, the tables are "analyzed" such that their statistics are up-to-date. Each query is executed once in a warm-up round and then the queries are executed in a round-robin fashion such that each query gets executed five times. We report the average execution times. The results are shown in Figure 6.

For `Q1`, it can be seen that the MDW variant is the fastest followed by the array variant (38.3 seconds and 49.1 seconds, respectively). These two query variants have similar plans, but with arrays there are fewer rows to process. On the other hand, these rows need to have their arrays "unnested" to produce as many values as there are rows to consider in the MDW variant. When the denormalized variant is considered, there are also many rows and these rows are wide. Further, the plan is not similar to the plans for the other variants as GROUP BY is necessary with this variant. This makes the denormalized variant the slowest (123.4 seconds).

For `Q2`, the MDW variant is again the fastest (8.9 seconds) to use. Again, the array variant is the second fastest (11.1 seconds). With this variant the arrays must again be unnested to produce the values that are available in the rows in the MDW variant. The denormalized variant uses wider rows and is the slowest (16.8 seconds).

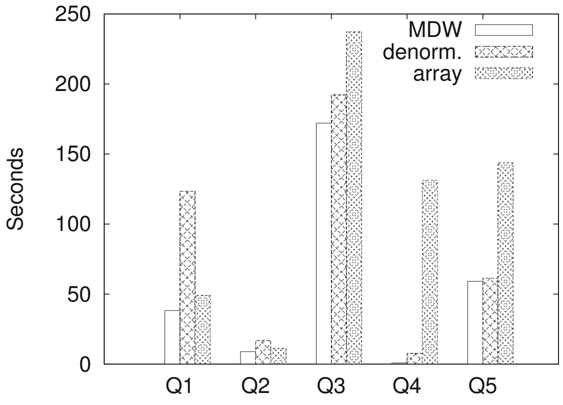


Fig. 6. Results of performance study

For Q3, the MDW variant remains the fastest (172.1 seconds) while the array variant now is the slowest (237.2 seconds) even though it avoids a join. On the other hand, the array variant requires a SELECT clause to unnest the array and an extra use of ROW_NUMBER to recreate the values from intervalNr which only are implicitly available from the array positions. The denormalized variant (192.2 seconds) is bit slower than the MDW variant even though it avoids a join.

For Q4, the MDW variant is significantly faster (0.8 seconds) than the others. The denormalized variant which avoids a join, uses an order of magnitude more time (7.7 seconds). The array variant is by far the slowest (131.9 seconds) as there is no index on timeIntervalId which is an array. Thus all rows must be processed and have their rows unnested to perform a join with D_timeInterval.

For Q5, the MDW and denormalized variants perform similarly (59.1 and 61.3 seconds, respectively). The queries involve the same number of rows and are identical apart from that the denormalized variant uses a wider table. For the array variant, the first CTE has to unnest two arrays and the query takes longer time (143.8 seconds).

To summarize, the MDW variant performs the best for all queries. Another interesting thing to consider, is the disk space usage. The tables F_flexOffer, F_enProfileInterval, F_timeSeriesInterval, and D_timeSeries take up 4.1 GB in the MDW variant (not counting indexes). Their alternative representations take up 7.0 GB in the denormalized variant and 1.9 GB in the array variant, respectively. It notable how little space the array variant uses compared to the other variants due to its fewer number of rows (and thus fewer space-consuming row headers). Overall, the MDW variant is a good choice considering both its performance and space requirements.

8 Related Work

In the energy sector, there is number of standardized data models used to represent the major objects in an electric utility enterprise [6] as well as to define administrative data internally interchanged between European electricity markets [4,5]. These models focus on various aspects of energy trading and physical electricity delivery, and specify 1) components of a power system at the electrical level, 2) actors and roles involved in the energy trading, 3) relationships and data exchange between those entities. These models are used as a basis for the MIRACLE data model [8], which further enriches them with the concept of shiftable consumption and production. All these models, however, focus on a semantic rather than the storage or the management of energy-related entities. By focusing on two most important entities in MIRABEL, i.e., time series and flex-offers, this paper, on the other hand, presents data representation models for these two types of entities offering a convenient storage and a good performance of analytical queries.

This paper is the first to dealt with the storage of flex-offers, but there are previous works which focus on time series and warehousing, e.g. UML-based modeling of time-series in DWs [15], and temporal aggregation of multidimensional data [3], and temporal DWs exploiting research results from the field of temporal databases [9]. Our modeling of different time-series types have similarities with Bauer et al.'s work [1]. They discuss "locally valid dimensional attributes" whose existence depends on values of dimensional elements. This is the case, e.g., for our attribute energyType which only exists if the D_type value represents an energy time-series. The problem of representing all these attributes in a single dimension table (as in a typical star schema) is that there will be many NULLs in the held data. Bauer et al. propose to have separate tables with the specific attributes and then create views "on top" of these with common attributes as well as textual values showing the name of the relation the data comes from which can be used for hierarchical classification. In contrast, we use tables (and not views) for the common attributes of a dimension and then represent special attributes that only exist for some dimensional values in other tables that reference the table with the common attributes. This makes it possible to declare foreign keys to the dimension table with the common attributes and also declare indexes and constraints on these tables. Bauer et al. also propose to use table inheritance to represent such cases. This would also be possible in our DBMS [12], but constraints cannot be enforced on child tables then.

In the current paper, we consider different representations of profile intervals and time series intervals which can be considered as facts with multi-valued measures. The latter case also has a many-many relationship between the time series facts and the time interval dimension. Previous work [13] has considered many-many relationships between fact tables and dimension tables. Our denormalized representation is similar to one of the methods of [13] whereas our other approaches with fact tables referencing other fact tables and measure values in arrays, respectively, are different.

9 Conclusion

In this paper, we have presented a DW schema for managing the complex energy data in a smart grid, including actors playing roles, flex-offers, and different types of time series. The schema has a number of interesting complexities such as facts about facts and composed non-atomic facts. The different nodes will hold different parts of the data accordingly to the roles of the node owners and the data will be at different aggregation levels at different nodes. The same schema can, however, be used for all kinds of nodes. We have considered different alternatives for the schema modeling using denormalization and arrays, respectively, but based on the performance and space usage, the chosen design is favourable. In the near future, we are going to perform large-scale simulations with realistic data amounts from different types of nodes. We will also address the challenges with distribution of the data on many nodes such propagation of data through the hierarchy, caching, etc. Further, we plan to investigate the possibilities for having specialized versions of the schema for different types of nodes, but such that queries can be formulated on the generic schema and automatically be translated to the specialized schemas to make the results combinable.

References

1. Bauer, A., Hümmer, W., Lehner, W.: An Alternative Relational OLAP Modeling Approach. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 189–198. Springer, Heidelberg (2000)
2. Boehm, M., et al.: Data Management in the M Smart Grid System. In: Proc. of EDBT/ICDT Workshops (2012)
3. Böhlen, M.H., Gamper, J., Jensen, C.S.: Multi-dimensional Aggregation for Temporal Data. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 257–275. Springer, Heidelberg (2006)
4. European Network of Transmission System Operators for Electricity. The Harmonised Electricity Market Role Model, version 2011-01 (June 12, 2012), http://www.ebix.org/Documents/role_model_v2011_01.pdf
5. Introduction to Business Requirements and Information Models (June 12, 2012), http://www.ebix.org/documents/Introduction%20to%20ebIX%20Models%200.0.D.pdf
6. IEC61970-301 Ed. 2, Energy management system application program interface (EMS-API) - Part 301: Common information model (CIM) base, International Electrotechnical Commission (2009)
7. Jensen, C.S., Pedersen, T.B., Thomsen, C.: Multidimensional Databases and Data Warehousing. Morgan & Claypool (2010)
8. Konsman, M.J., Rumph, F.J.: MIRABEL Deliverable 2.3: Final data model, specification of request and negotiation messages and contracts (June 12, 2012), http://www.db.inf.tu-dresden.de/miracle/files/deliverables/M18/D2.3_final.pdf
9. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design From Conventional to Spatial and Temporal Applications. Springer (2009)
10. www.meregio.de/en/ (June 12, 2012)
11. www.mirabel-project.eu/ (June 12, 2012)
12. postgresql.org (June 12, 2012)
13. Song, I.-Y., et al.: An Analysis of Many-to-Many Relationships Between Fact and Dimension Tables in Dimensional Modeling. In: Proc. of DMDW (2001)
14. Šikšnys, L., Khalefa, M.E., Pedersen, T.B.: Aggregating and Disaggregating Flexibility Objects. In: Ailamaki, A., Bowers, S. (eds.) SSDBM 2012. LNCS, vol. 7338, pp. 379–396. Springer, Heidelberg (2012)
15. Zubcoff, J., Pardillo, J., Trujillo, J.: A UML profile for the conceptual modelling of data-mining with time-series in data warehouses. Information and Software Technology 51(6), 977–992 (2008)

Author Index