

LNCS 5181

Sourav S. Bhowmick
Josef Küng
Roland Wagner (Eds.)

Database and Expert Systems Applications

19th International Conference, DEXA 2008
Turin, Italy, September 2008
Proceedings

DEXA 2008



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Sourav S. Bhowmick Josef Küng
Roland Wagner (Eds.)

Database and Expert Systems Applications

19th International Conference, DEXA 2008
Turin, Italy, September 1-5, 2008
Proceedings



Springer

Volume Editors

Sourav S. Bhowmick
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798
E-mail: assourav@ntu.edu.sg

Josef Küng
Roland Wagner
University of Linz, Altenbergerstraße 69, 4040 Linz, Austria
E-mail: {jkueng, rrwagner}@faw.at

Library of Congress Control Number: 2008933702

CR Subject Classification (1998): H.2, H.4, H.3, I.2, J.1

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

ISSN	0302-9743
ISBN-10	3-540-85653-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-85653-5 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springer.com

© Springer-Verlag Berlin Heidelberg 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12513642 06/3180 5 4 3 2 1 0

Preface

The annual international conference on Database and Expert Systems Applications (DEXA) is now well established as a reference scientific event. The reader will find in this volume a collection of scientific papers that represent the state of the art of research in the domain of data, information and knowledge management, intelligent systems, and their applications.

The 19th DEXA conference was held at the Politecnico di Torino, on September 1–5, 2008.

Several collocated conferences and workshops covered specialized and complementary topics to the main conference topic. Seven conferences – the 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), the 8th International Conference on Electronic Commerce and Web Technologies (EC-Web), the 6th International Conference on Electronic Government (EGOV), the 4th International Conference on Trust, Privacy, and Security in Digital Business (TrustBus), the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS), the 2nd International Conference on Network-Based Information Systems (NBiS), and the 1st International Conference on Data Management in Grid and P2P Systems (GLOBE) – and seventeen workshops, were collocated with DEXA 2008.

The whole forms a unique international event with a balanced depth and breadth of topics. Its much-appreciated conviviality fosters unmatched opportunities to meet, share the latest scientific results and discuss the latest technological advances in the area of information technologies with both young scientists and engineers and senior world-renowned experts.

This volume contains the papers selected for presentation at the DEXA conference. Each submitted paper was reviewed by 3 or 4 reviewers, members of the program committee or external reviewers appointed by members of the program committee. This year, for the first time in the history of DEXA, we added an additional dimension to the framework of the selection of research papers. Based on the reviews, the program committee accepted two categories of papers: 39 *regular* papers and 35 *short* papers of the 208 originally submitted papers. Regular papers were given a maximum of 14 pages in the proceedings to report their results as well as 25 minutes presentation time at the conference. Note that, we deviated from the past tradition and increased the page limit to 14 in order to give authors more space to report their research results in detail. Short papers were given an 8-page limit and a 15-minute presentation slot. We believe that this was yet another step towards further increasing the quality and competitiveness of DEXA.

The excellence brought to you in these proceedings would not have been possible without the efforts of numerous individuals and the support of several organizations.

First and foremost, we thank the authors for their hard work and for the quality of their submissions. We also thank Enrico Ferro, Curtis Dyreson, Sanjay Madria, A Min Tjoa, Roland Wagner, Gabriela Wagner, the members of the program committee, the reviewers, and the many others who assisted in the organization of DEXA 2008, for

their contribution to the success and high standard of the conference and of these proceedings.

Finally, we thank the DEXA Association, the Research Institute for Applied Knowledge Processing (FAW), the Istituto Superiore Mario Boella (ISMB), the CSI Piemonte, the Regione Piemonte, and the Politecnico di Torino, for making DEXA 2008 happen.

June 2008

Sourav S. Bhowmick
Josef Küng

Organization

General Chairpersons

Enrico Ferro	Istituto Superiore Mario Boella / Polytechnic of Turin, Italy
Emilio Paolucci	Polytechnic of Turin, Italy
Marco Cantamessa	Polytechnic of Turin, Italy

Conference Program Chairpersons

Josef Küng	University of Linz, Austria
Sourav S. Bhowmick	Nanyang Technological University, Singapore

Workshop Chairpersons

A. Min Tjoa	Technical University of Vienna, Austria
Roland R. Wagner	FAW, University of Linz, Austria

Publication Chairperson

Vladimir Marik	Czech Technical University, Czech Republic

Program Committee

Witold Abramowicz	The Poznan University of Economics, Poland
Fuat Akal	University of Basel, Switzerland
Toshiyuki Amagasa	University of Tsukuba, Japan
Ira Assent	Aachen University, Germany
Ramazan S. Aygun	University of Alabama in Huntsville, USA
Torben Bach Pedersen	Aalborg University, Denmark
James Bailey	University of Melbourne, Australia
Denilson Barbosa	University of Calgary, Canada
Peter Baumann	University of Bremen, Germany
Ladjel Bellatreche	ENSMA-Poitiers University, France
Bishwaranjan Bhattacharjee	IBM Thomas J. Watson Research Center, USA
Sourav S. Bhowmick	Nanyang Technological University, Singapore
Stephen Blott	Dublin City University, Ireland
Peter Boncz	Centrum voor Wiskunde en Informatica, Netherlands
Athman Bouguettaya	Virginia Tech University, USA

Kjell Bratbergsengen	Norwegian University of Science and Technology, Norway
Stephane Bressan	National University of Singapore, Singapore
Martin Breunig	University of Osnabrück, Germany
Luis M. Camarinha-Matos	Universidade Nova de Lisboa + Uninova, Portugal
Silvana Castano	Università degli Studi di Milano, Italy
Barbara Catania	Università di Genova, Italy
Wojciech Cellary	University of Economics at Poznan, Poland
Chee-Yong Chan	National University of Singapore, Singapore
Elizabeth Chang	Curtin University, Australia
Sudarshan S. Chawathe	University of Maine, USA
Sanjay Chawla	University of Sydney, Australia
Yi Chen	Arizona State University, USA
Cindy Chen	University of Massachusetts Lowell, USA
Reynold Cheng	Hong Kong Polytechnic University, China
Byron Choi	Nanyang Technological University, Singapore
Henning Christiansen	Roskilde University, Denmark
Rosine Cicchetti	IUT, University of Marseille, France
Frans Coenen	The University of Liverpool, UK
Gao Cong	Microsoft Research Asia, China
Bin Cui	Peking University, China
Alfredo Cuzzocrea	University of Calabria, Italy
Tran Khanh Dang	Ho Chi Minh City University of Technology, Vietnam
Gautam Das	University of Texas, Arlington, USA
John Debenham	University of Technology, Sydney, Australia
Elisabetta Di Nitto	Politecnico di Milano, Italy
Gillian Dobbie	University of Auckland, New Zealand
Dirk Draheim	Software Competence Center Hagenberg, Austria
Curtis Dyreson	Utah State University, USA
Johann Eder	University of Vienna, Austria
Suzanne M. Embury	The University of Manchester, UK
Leonidas Fegaras	The University of Texas at Arlington, USA
Ling Feng	University of Twente, The Netherlands
Eduardo Fernandez	Florida Atlantic University, USA
Ada Fu	Chinese University of Hong Kong, China
Mariagrazia Fugini	Politecnico di Milano, Italy
Antonio L. Furtado	Pontificia Universidade Catolica do R.J., Brazil
Bin Gao	Microsoft Research Asia, China
Mário J. Gaspar da Silva	University of Lisboa, Portugal
Jan Goossenaerts	Eindhoven University of Technology, The Netherlands
Fabio Grandi	University of Bologna, Italy
William Grosky	University of Michigan, USA
Le Gruenwald	University of Oklahoma, USA
Francesco Guerra	Università degli Studi Di Modena e Reggio Emilia, Italy
Abdelkader Hameurlain	University of Toulouse, France
Wook-Shin Han	Kyungpook National University, Korea

Takahiro Hara	Osaka University, Japan
Igor T. Hawryszkiewicz	University of Technology, Sydney, Australia
Vagelis Hristidis	Florida International University, USA
Wynne Hsu	National University of Singapore, Singapore
Ela Hunt	ETH Zürich, Switzerland
San-Yih Hwang	National Sun Yat-Sen University, Taiwan
Mohamed Ibrahim	University of Greenwich, UK
Mizuho Iwaihara	Kyoto University, Japan
Dimitris Karagiannis	University of Vienna, Austria
George Karypis	University of Minnesota, USA
Anastasios Kementsietsidis	IBM T.J. Watson Research Center, USA
Myoung Ho Kim	KAIST, Korea
Sang-Wook Kim	Hanyang University, Korea
Stephen Kimani	University of Rome "La Sapienza", Italy
Gary J. Koehler	University of Florida, USA
Christian König	Microsoft Research, USA
Hanna Kozankiewicz	Polish Academy of Sciences, Poland
Michal Krátký	VSB-Technical University of Ostrava, Czech Republic
John Krogstie	SINTEF, Norway
Petr Kroha	Technische Universität Chemnitz-Zwickau, Germany
Josef Küng	University of Linz, Austria
Sergey Kuznetsov	Russian Academy of Sciences, Russia
Lotfi Lakhal	University of Marseille, France
Young-Koo Lee	Kyung Hee University, Korea
Mong Li Lee	National University of Singapore, Singapore
Dongwon Lee	Pennsylvania State University, USA
Ulf Leser	Humboldt University of Berlin, Germany
Xuemin Lin	University of New South Wales, Sydney, Australia
Tok Wang Ling	National University of Singapore, Singapore
Volker Linnemann	University of Lübeck, Germany
Mengchi Liu	Carleton University, Canada
Peri Loucopoulos	The University of Manchester, UK
Sanjai Kumar Madria	University of Missouri-Rolla, USA
Vladimir Marik	Czech Technical University, Czech Republic
Simone Marinai	University of Florence, Italy
Elio Masciari	University of Southern California, Italy
Subhasish Mazumdar	New Mexico Tech, USA
Dennis McLeod	University of Southern California, USA
Xiaofeng Meng	Renmin University, China
Elisabeth Metais	CNAM, France
Klaus Meyer-Wegener	University of Erlangen and Nuremberg, Germany
Anirban Mondal	University of Tokyo, Japan
Yang-Sae Moon	Kangwon National University, Korea
Reagan Moore	San Diego Supercomputer Center, USA
Tadeusz Morzy	Poznan University of Technology, Poland
Wolfgang Nejdl	University of Hanover, Germany
Wilfred Ng	University of Science & Technology, Hong Kong

Daniela Nicklas	University of Stuttgart, Germany
Byung-Won On	Pennsylvania State University, USA
Gultekin Ozsoyoglu	University Case Western Research, USA
Oscar Pastor	Universidad Politecnica de Valencia, Spain
Verónica Peralta	Universidad de la Republica, Uruguay
Jaroslav Pokorny	Charles University in Prague, Czech Republic
Philippe Pucheral	INRIA, Université de Versailles, France
Magdalena Punceva	CERN, Switzerland
Gerald Quirchmayr	University of Vienna, Austria and Univ. of South Australia, Australia
Fausto Rabitti	ISTI, CNR Pisa, Italy
Wenny Rahayu	La Trobe University, Australia
Isidro Ramos	Technical University of Valencia, Spain
Ralf Rantzaue	IBM Silicon Valley Laboratory, USA
P. Krishna Reddy	International Institute of Information Technology, India
Colette Rolland	University Paris I, Sorbonne, France
Domenico Sacca	University of Calabria, Italy
Simonas Saltenis	Aalborg University, Denmark
Maryá Luýsa Sapino	Università degli Studi di Torino, Italy
Kai-Uwe Sattler	Technical University of Ilmenau, Germany
Ralf Schenkel	Max Planck Institute, Germany
Stefanie Scherzinger	Saarland University, Germany
Ingo Schmitt	University of Magdeburg, Germany
Harald Schöning	Software AG, Germany
Holger Schwarz	University of Stuttgart, Germany
Erich Schweighofer	University of Vienna, Austria
Sergej Sizov	University of Koblenz, Germany
Giovanni Soda	University of Florence, Italy
Dmitri Soshnikov	Moscow Aviation Technical University, Microsoft Russia, Russia
Srinath Srinivasa	IIIT-B, India
Bala Srinivasan	Monash University, Australia
Zbigniew Struzik	The University of Tokyo, Japan
Aixin Sun	Nanyang Technological University, Singapore
Keishi Tajima	Kyoto University, Japan
Makoto Takizawa	Tokyo Denki University, Japan
Kian-Lee Tan	National University of Singapore, Singapore
Katsumi Tanaka	Kyoto University, Japan
Yufei Tao	City University of Hong Kong, Hong Kong
Wei-Guang Teng	National Cheng Kung University, Taiwan
Stephanie Teufel	University of Fribourg, Switzerland
Jukka Teuhola	University of Turku, Finland
Bernhard Thalheim	University of Kiel, Germany
J.M. Thevenin	University of Toulouse, France
Helmut Thoma	University of Basel, Switzerland
A Min Tjoa	Technical University of Vienna, Austria

Roland Traunmüller	University of Linz, Austria
Anthony Tung	National University of Singapore, Singapore
Maurice van Keulen	University of Twente, Netherlands
Aparna Varde	Virginia State University, USA
Genoveva Vargas-Solar	LSR-IMAG, France
Yannis Vassiliou	National Technical University of Athens, Greece
Krishnamurthy Vidyasankar	Memorial Univ. of Newfoundland, Canada
Peter Vojtas	Charles University in Prague, Czech Republic
Wei Wang	University of New South Wales, Sydney, Australia
John Wilson	University of Strathclyde, UK
Marek Wojciechowski	Poznan University of Technology, Poland
Viacheslav Wolfengagen	Institute for Contemporary Education, Russia
Raymond Wong	University of New South Wales, Sydney, Australia
Ming-Chuan Wu	Microsoft Corporation, USA
Dong Xin	Microsoft Research, USA
Clement Yu	University of Illinois at Chicago, USA
Jeffrey Xu Yu	Chinese University of Hong Kong, China
Osmar Zaiane	University of Alberta, Canada
Gian Piero Zarri	University Paris IV, Sorbonne, France
Arkady Zaslavsky	Monash University, Australia
Yifeng Zheng	University of Pennsylvania, USA
Aoying Zhou	Fudan University, China
Yongluan Zhou	National University of Singapore, Singapore
Qiang Zhu	The University of Michigan, USA
Ester Zumpano	University of Calabria, Italy

External Reviewers

Massimo Ruffolo	Fernando Farfán
Domenico Ursino	Ramakrishna Varadarajan
Changqing Chen	Jaume Baixeries
Adegoke Ojewole	Karam Gouda
Lubomir Stanchev	Jinsoo Lee
Gang Qian	Meishan Hu
Victoria Torres	Manoranjan Dash
Pedro Valderas	An Lu
Carlo Meghini	Kenneth Leung
Matteo Mordacchini	Qiong Fong
Claudio Lucchese	Marco Grawunder
Giuseppe Amato	André Bolles
Pasquale Savino	Jonas Jacobi
Jarogniew Rykowski	Gyözö Gidófalvi
Luciano Caroprese	Man Lung Yiu
Irina Trubitsyna	Samira Jaeger
Peng Sun	Philip Groth
Yichuan Cai	Ziyang Liu

Yu Huang
Xumin Liu
Qi Yu
Zaki Malik
T.Ragunathan
M.Venugopal Reddy
R. Uday Kiran
M. Kumaraswamy
Alain Casali
Sebastien Nedjar
Viet Phan Luong
Sadok Ben Yahia
Ding Chen
Mehdi Benzine
Shaoyi Yin
Sergio Flesca
Filippo Furfaro
Giuseppe M. Mazzeo
Vincenzo Russo
Karin Koogan Breitman
Horst Pichler
Uwe Roehm
Donghui Zhang

Lev Novik
Pawel Terlecki
Rui Wang
Ninad Joshi
Vani Jain
Mitesh Naik
Derry Wijaya
Wee Hyong Tok
Wei Liu
Fangjiao Jiang
Jinchuan Chen
Xike Xie
Ki Yong Lee
José Hilario Canós
Pepe Carsí
Cristóbal Costa
Abel Gómez
Eva Onaindia
Zhifeng Bao
Jiaheng Lu
Huayu Wu
Liang Xu

Table of Contents

Invited Talk

Towards Engineering Purposeful Systems: A Requirements Engineering Perspective.....	1
<i>Colette Rolland</i>	

Session 1

Data Privacy

Hiding Frequent Patterns under Multiple Sensitive Thresholds	5
<i>Ya-Ping Kuo, Pai-Yu Lin, and Bi-Ru Dai</i>	
<i>BSGI</i> : An Effective Algorithm towards Stronger l -Diversity	19
<i>Yang Ye, Qiao Deng, Chi Wang, Dapeng Lv, Yu Liu, and Jianhua Feng</i>	

Temporal, Spatial, and High Dimensional Databases I

The Truncated Tornado in TMBB: A Spatiotemporal Uncertainty Model for Moving Objects	33
<i>Shayma Alkobaisi, Petr Vojtěchovský, Wan D. Bae, Seon Ho Kim, and Scott T. Leutenegger</i>	
Reordering of Location Identifiers for Indexing an RFID Tag Object Database	41
<i>Sungwoo Ahn and Bonghee Hong</i>	
A Free Terrain Model for Trajectory K -Anonymity	49
<i>Aris Gkoulalas-Divanis and Vassilios S. Verykios</i>	
HRG: A Graph Structure for Fast Similarity Search in Metric Spaces ...	57
<i>Omar U. Florez and SeungJin Lim</i>	

Session 2A: Semantic Web and Ontologies

Word Sense Disambiguation as the Primary Step of Ontology Integration	65
<i>Marko Banek, Boris Vrdoljak, and A Min Tjoa</i>	
Enriching Ontology for Deep Web Search	73
<i>Yoo Jung An, Soon Ae Chun, Kuo-chuan Huang, and James Geller</i>	

POEM: An Ontology Manager Based on Existence Constraints	81
<i>Nadira Lammari, Cédric du Mouza, and Elisabeth Métais</i>	

Session 2B: Query Processing

Extending Inconsistency-Tolerant Integrity Checking by Semantic Query Optimization	89
<i>Hendrik Decker</i>	
On the Evaluation of Large and Sparse Graph Reachability Queries	97
<i>Yangjun Chen</i>	
SQL TVF Controlling Forms – Express Structured Parallel Data Intensive Computing	106
<i>Qiming Chen and Meichun Hsu</i>	
A Decidable Fuzzy Description Logic $F\text{-}ALC(G)$	116
<i>Hailong Wang and Z.M. Ma</i>	

Session 3: Web and Information Retrieval

Ranking Entities Using Comparative Relations	124
<i>Takeshi Kurashima, Katsuji Bessho, Hiroyuki Toda, Toshio Uchiyama, and Ryoji Kataoka</i>	
Query Recommendation Using Large-Scale Web Access Logs and Web Page Archive	134
<i>Lin Li, Shingo Otsuka, and Masaru Kitsuregawa</i>	
Description Logic to Model a Domain Specific Information Retrieval System	142
<i>Saïd Radhouani, Gilles Falquet, and Jean-Pierre Chevalletinst</i>	
Extending the Edit Distance Using Frequencies of Common Characters	150
<i>Muhammad Marwan Muhammad Fuad and Pierre-François Marteau</i>	

Session 4: Mobile Data and Information

Tracking Moving Objects in Anonymized Trajectories	158
<i>Nikolay Vyahhi, Spiridon Bakiras, Panos Kalnis, and Gabriel Ghinita</i>	
REALM: Replication of Data for a Logical Group Based MANET Database	172
<i>Anita Vallur, Le Gruenwald, and Nick Hunter</i>	

A Cache Management Method for the Mobile Music Delivery System: JAMS	186
<i>Hiroaki Shibata, Satoshi Tomisawa, Hiroki Endo, and Yuka Kato</i>	

EcoRare: An Economic Incentive Scheme for Efficient Rare Data Accessibility in Mobile-P2P Networks	196
<i>Anirban Mondal, Sanjay Kumar Madria, and Masaru Kitsuregawa</i>	

Session 5: Data and Information Streams

Identifying Similar Subsequences in Data Streams	210
<i>Machiko Toyoda, Yasushi Sakurai, and Toshikazu Ichikawa</i>	

A Tree-Based Approach for Event Prediction Using Episode Rules over Event Streams	225
<i>Chung-Wen Cho, Ying Zheng, Yi-Hung Wu, and Arbee L.P. Chen</i>	

Effective Skyline Cardinality Estimation on Data Streams	241
<i>Yang Lu, Jiakui Zhao, Lijun Chen, Bin Cui, and Dongqing Yang</i>	

Session 6: Data Mining Algorithms

Detecting Current Outliers: Continuous Outlier Detection over Time-Series Data Streams	255
<i>Kozue Ishida and Hiroyuki Kitagawa</i>	

Component Selection to Optimize Distance Function Learning in Complex Scientific Data Sets	269
<i>Aparna Varde, Stephen Bique, Elke Rundensteiner, David Brown, Jianyu Liang, Richard Sisson, Ehsan Sheybani, and Brian Sayre</i>	

Emerging Pattern Based Classification in Relational Data Mining	283
<i>Michelangelo Ceci, Annalisa Appice, and Donato Malerba</i>	

Session 7: Multimedia Databases

Rosso Tiziano: A System for User-Centered Exploration and Discovery in Large Image Information Bases	297
<i>Giovanni Maria Sacco</i>	

NM-Tree: Flexible Approximate Similarity Search in Metric and Non-metric Spaces	312
<i>Tomáš Skopal and Jakub Lokoč</i>	

Efficient Processing of Nearest Neighbor Queries in Parallel Multimedia Databases	326
<i>Jorge Manjarrez-Sanchez, José Martinez, and Patrick Valduriez</i>	

Session 8: Data Mining Systems, Data Warehousing, OLAP

OLAP for Trajectories	340
<i>Oliver Baltzer, Frank Dehne, Susanne Hambrusch, and Andrew Rau-Chaplin</i>	
A Probabilistic Approach for Computing Approximate Iceberg Cubes...	348
<i>Alfredo Cuzzocrea, Filippo Furfaro, and Giuseppe M. Mazzeo</i>	
Noise Control Boundary Image Matching Using Time-Series Moving Average Transform	362
<i>Bum-Soo Kim, Yang-Sae Moon, and Jinho Kim</i>	
Approximate Range-Sum Queries over Data Cubes Using Cosine Transform	376
<i>Wen-Chi Hou, Cheng Luo, Zhewei Jiang, Feng Yan, and Qiang Zhu</i>	

Session 9: Temporal, Spatial, and High Dimensional Databases II

Querying Multigranular Spatio-temporal Objects	390
<i>Elena Camossi, Michela Bertolotto, and Elisa Bertino</i>	
Space-Partitioning-Based Bulk-Loading for the NSP-Tree in Non-ordered Discrete Data Spaces	404
<i>Gang Qian, Hyun-Jeong Seok, Qiang Zhu, and Sakti Pramanik</i>	
Efficient Updates for Continuous Skyline Computations	419
<i>Yu-Ling Hsueh, Roger Zimmermann, and Wei-Shinn Ku</i>	

Session 10: Data and Information Semantics

Semantic Decision Tables: Self-organizing and Reorganizable Decision Tables	434
<i>Yan Tang, Robert Meersman, and Jan Vanthienen</i>	
Translating SQL Applications to the Semantic Web	450
<i>Syed Hamid Tirmizi, Juan Sequeda, and Daniel Miranker</i>	
An Agent Framework Based on Signal Concepts for Highlighting the Image Semantic Content	465
<i>Mohammed Belkhatir</i>	

Session 11: XML Databases I

Supporting Proscriptive Metadata in an XML DBMS	479
<i>Hao Jin and Curtis Dyreson</i>	
XPath Rewriting Using Multiple Views	493
<i>Junhu Wang and Jeffrey Xu Yu</i>	

Superimposed Code-Based Indexing Method for Extracting MCTs from XML Documents	508
<i>Wenxin Liang, Takeshi Miki, and Haruo Yokota</i>	
Fast Matching of Twig Patterns	523
<i>Jiang Li and Junhu Wang</i>	

Session 12: XML Databases II

XML Filtering Using Dynamic Hierarchical Clustering of User Profiles	537
<i>Panagiotis Antonellis and Christos Makris</i>	
Person Retrieval on XML Documents by Coreference Analysis Utilizing Structural Features.....	552
<i>Yumi Yonei, Mizuho Iwaihara, and Masatoshi Yoshikawa</i>	
HFilter: Hybrid Finite Automaton Based Stream Filtering for Deep and Recursive XML Data	566
<i>Weiwei Sun, Yongrui Qin, Ping Yu, Zhuoyao Zhang, and Zhenying He</i>	

Session 13: Query Processing and Optimization

Read-Optimized, Cache-Conscious, Page Layouts for Temporal Relational Data	581
<i>Khaled Jouini, Geneviève Jomier, and Patrick Kabore</i>	
Exploiting Interactions among Query Rewrite Rules in the Teradata DBMS.....	596
<i>Ahmad Ghazal, Dawit Seid, Alain Crolotte, and Bill McKenna</i>	
Optimal Preference Elicitation for Skyline Queries over Categorical Domains	610
<i>Jongwuk Lee, Gae-won You, Seung-won Hwang, Joachim Selke, and Wolf-Tilo Balke</i>	

Session 14A: Data and Information Streams

Categorized Sliding Window in Streaming Data Management Systems	625
<i>Marios Papas, Josep-L. Larriba-Pey, and Pedro Trancoso</i>	
Time to the Rescue – Supporting Temporal Reasoning in the Rete Algorithm for Complex Event Processing	635
<i>Karen Walzer, Matthias Groch, and Tino Breddin</i>	

Classifying Evolving Data Streams Using Dynamic Streaming Random Forests	643
<i>Hanady Abdulsalam, David B. Skillicorn, and Patrick Martin</i>	

Session 14B: Potpourri

On a Parameterized Antidivision Operator for Database Flexible Querying	652
<i>Patrick Bosc and Olivier Pivert</i>	
Providing Explanations for Database Schema Validation	660
<i>Guillem Rull, Carles Farré, Ernest Teniente, and Toni Urpí</i>	
Temporal Conformance of Federated Choreographies	668
<i>Johann Eder and Amirreza Tahamtan</i>	
Relational Database Migration: A Perspective	676
<i>Abdelsalam Maatuk, Akhtar Ali, and Nick Rossiter</i>	

Session 15A: Data Mining

DB-FSG: An SQL-Based Approach for Frequent Subgraph Mining	684
<i>Sharma Chakravarthy and Subhesh Pradhan</i>	
Efficient Bounds in Finding Aggregate Nearest Neighbors	693
<i>Sansarkhuu Namnandorj, Hanxiong Chen, Kazutaka Furuse, and Nobuo Ohbo</i>	
A Grid-Based Multi-relational Approach to Process Mining	701
<i>Antonio Turi, Annalisa Appice, Michelangelo Ceci, and Donato Malerba</i>	
Extraction of Opposite Sentiments in Classified Free Format Text Reviews	710
<i>Dong (Haoyuan) Li, Anne Laurent, Mathieu Roche, and Pascal Poncelet</i>	

Session 15B: XML Databases

Navigational Path Expressions on XML Schemas	718
<i>Federico Cavalieri, Giovanna Guerrini, and Marco Mesiti</i>	
Transforming Tree Patterns with DTDs for Query Containment Test ...	727
<i>Junhu Wang, Jeffrey Xu Yu, Chengfei Liu, and Rui Zhou</i>	
XSelMark: A Micro-benchmark for Selectivity Estimation Approaches of XML Queries	735
<i>Sherif Sakr</i>	

Session 16A: Applications of Database, Information, and Decision Support Systems

A Method for Semi-automatic Standard Integration in Systems Biology	745
<i>Dagmar Köhn and Lena Strömbäck</i>	
FDBMS Application for a Survey on Educational Performance.....	753
<i>Livia Borjas, Alejandro Fernández, Jorge Ortiz, and Leonid Tineo</i>	
Hierarchy Encoding with Multiple Genes.....	761
<i>Martin van Bommel and Ping Wang</i>	
Knowledge Mining for the Business Analyst	770
<i>Themis Palpanas and Jakka Sairamesh</i>	

Session 16B: Optimization and Performance

Controlling the Behaviour of Database Servers with 2PAC and DiffServ	779
<i>Luís Fernando Orleans, Geraldo Zimbrão, and Pedro Furtado</i>	
Compressing Very Large Database Workloads for Continuous Online Index Selection	791
<i>Piotr Kołaczowski</i>	
Escaping a Dominance Region at Minimum Cost.....	800
<i>Youngdae Kim, Gae-won You, and Seung-won Hwang</i>	

Session 17: Schema, Process and Knowledge Modelling and Evolution

Evolutionary Clustering in Description Logics: Controlling Concept Formation and Drift in Ontologies	808
<i>Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito</i>	
Model-Driven, View-Based Evolution of Relational Databases.....	822
<i>Eladio Domínguez, Jorge Lloret, Ángel L. Rubio, and María A. Zapata</i>	
Inventing Less, Reusing More, and Adding Intelligence to Business Process Modeling	837
<i>Lucinéia H. Thom, Manfred Reichert, Carolina M. Chiao, Cirano Iochpe, and Guillermo N. Hess</i>	
Author Index	851

Towards Engineering Purposeful Systems: A Requirements Engineering Perspective

Colette Rolland

Université Paris1 Panthéon Sorbonne
CRI
90 Rue de Tolbiac, 75013 Paris
rolland@univ-paris1.fr

Abstract. A number of studies [1][2][3] show that systems fail due to an inadequate or insufficient understanding of the requirements they seek to address. Further, the amount of effort needed to fix these systems has been found to be very high [4]. To correct this situation, it is necessary to address the issue of requirements elicitation, validation, and specification in a relatively more focused manner. The expectation is that as a result of this, more acceptable systems will be developed in the future. The field of requirements engineering has emerged to meet this expectation. Requirements Engineering extends the ‘what is done by the system’ approach with the ‘why is the system like this’ view. This ‘why’ question is answered in terms of organizational objectives and their impact on information systems supporting the organization. In other words, information systems are seen as fulfilling a certain purpose in an organization and requirements engineering helps in the conceptualization of these purposeful systems.

The talk will focus on the above issue of conceptualising purposeful systems.

It will argue that the goal concept is central to resolving the issue of conceptualising purposeful systems and then, elaborate on goal modeling and reasoning with goals in order to demonstrate the various roles of goals in conceptualizing purposeful systems: Goal modeling proved to be an effective way to elicit requirements [7][8][9][10][11][12][13]. The assumption of goal-based requirements elicitation is that the rationale for developing a system is found outside the system itself, in the enterprise [14] in which the system shall function. In this movement from the ‘whats’ to the ‘whys’, it becomes mandatory to consider multiple view points of the various stakeholders, to explore alternative design choices and reason about them so as to make conceptual decisions on the basis of rationale arguments in favour and against the different alternatives. Alternative goal refinement proved helpful in the systematic exploration of system choices [8][14][15][16]. Recording these shall help to deal with changing requirements. Goals provide a means to ensure requirements pre-traceability [5][17][18]. They establish a conceptual link between the system and its environment, thus facilitating the propagation of organizational changes into the system functionality. This link provides the rationale for the system functionality [6][8][19][20][21] and facilitates the

explanation and justification of it to the stakeholders. Stakeholders provide useful and realistic viewpoints about the system To-Be expressed as goals. Negotiation techniques have been developed to help choosing the prevalent one [22][23]. Prioritization techniques aim at providing means to compare the different viewpoints on the basis of costs and value [24][25]. Multiple viewpoints are inherently associated to conflicts [26] and goals have been recognized to help in the detection of conflicts and their resolution [27][28][29][30].

In the rest of this talk, we will consider new challenges raised by emerging conditions of system development leading to variability in functionality modelling and customisation in the engineering process. Variability is imposed by the multi-purpose nature of information systems of today. We will use a particular goal model called goal/strategy map to illustrate how a goal model can make variability explicit and support goal-based reasoning to help in selecting the right variant for the project at hand. Our position is that variability implies a move from systems with a mono-facetted purpose to those with a multi-facetted purpose. Whereas the former concentrates on goal discovery, the multi-facetted nature of a purpose extends it to consider the many different ways of goal achievement. For example, for the goal Purchase Material, earlier it would be enough to know that an organization achieves this goal by forecasting material need. Thus, Purchase material was mono-facetted: it had exactly one strategy for its achievement. However, in the new context, it is necessary to introduce other strategies as well, say the Re-order Point strategy for purchasing material. Purchase Material now is multi-facetted; it has many strategies for goal achievement. These two strategies, among others, are made available, for example, in the SAP Materials Management module [31].

The foregoing points to the need to balance goal-orientation with the introduction of strategies for goal achievement. This is the essence of goal/strategy maps. A goal/strategy map, or map for short, is a graph, with nodes as intentions and strategies as edges. An edge entering a node identifies a strategy that can be used for achieving the intention of the node. The map therefore, shows which intentions can be achieved by which strategies once a preceding intention has been achieved. Evidently, the map is capable of expressing goals and their achievement in a declarative manner. The talk will introduce the concept of a map [32], illustrate it with an ERP system example and discuss how the model meets the aforementioned challenges.

References

- [1] Standish Group, Chaos. Standish Group Internal Report (1995)
- [2] European Software Institute, European User Survey Analysis, Report USV_EUR 2.1, ESPITI Project (1996)
- [3] META Group, Research on Requirements Realization and Relevance, report (2003)
- [4] Johnson, J.: Chaos: the Dollar Drain of IT project Failures. Application Development Trends, pp.41–47 (1995)

- [5] Gotel, O., Finkelstein, A.: Modelling the contribution structure underlying requirements. In: 1st Int. Workshop on Requirements Engineering: Foundation of Software Quality, Utrecht, Netherlands (1994)
- [6] Ross, D.T., Schoman, K.E.: Structured Analysis for Requirements Definition. *IEEE Transactions on Software Engineering* 3(1), 6–15 (1977)
- [7] Potts, C., Takahashi, K., Antón, A.I.: Inquiry-based requirements analysis. *IEEE Software* 11(2), 21–32 (1994)
- [8] Rolland, C., Souveyet, C., Ben Achour, C.: Guiding goal modelling using scenarios. *IEEE Transactions on Software Engineering, Special Issue on Scenario Management* 24(12) (1998)
- [9] Dardenne, A., Lamsweerde, A.v., Fickas, S.: Goal-directed Requirements Acquisition, *Science of Computer Programming*, pp. 3–50. Elsevier, Amsterdam (1993)
- [10] Dubois, E., Yu, E., Pettot, M.: From early to late formal requirements: a process-control case study. In: *Proc. IWSSD 1998 - 9th International Workshop on software Specification and design*, pp. 34–42. IEEE CS Press, Los Alamitos (1998)
- [11] Antón, A.I., Potts, C., Takahashi, K.: Inquiry Based Requirements Analysis. In: *IEEE Conference on Requirements Engineering* (1994)
- [12] Kaindl, H.: A design process based on a model combining scenarios with goals and functions. *IEEE Trans. on Systems, Man and Cybernetic* 30(5), 537–551 (2000)
- [13] Lamsweerde, A.v.: Goal-oriented requirements engineering: a guided tour. In: *RE 2001 International Joint Conference on Requirements Engineering*, Toronto, pp. 249–263. IEEE, Los Alamitos (2001)
- [14] Loucopoulos, P.: The f3 (from fuzzy to formal) view on requirements engineering. *Ingénierie des systèmes d’information* 2(6), 639–655 (1994)
- [15] Rolland, C., Grosz, G., Kla, R.: Experience with goal-scenario coupling. in requirements engineering. In: *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, Limerik, Ireland (1999)
- [16] Hui, B., Liaskos, S., Mylopoulos, J.: Requirements Analysis for Customizable Software: A Goals-Skills-Preferences Framework. In: *IEEE Conference on Requirements Engineering*, Monterey Bay, USA, pp. 117–126 (2003)
- [17] Ramesh, B., Powers, T., Stubbs, C., Edwards, M.: Implementing requirements traceability: a case study. In: *Proceedings of the 2nd Symposium on Requirements Engineering (RE 1995)*, UK, pp. 89–95 (1995)
- [18] Pohl, K.: *Process centred requirements engineering*. J. Wiley and Sons Ltd., Chichester (1996)
- [19] Bubenko, J., Rolland, C., Loucopoulos, P., de Antón ellis, V.: Facilitating ‘fuzzy to formal’ requirements modelling. In: *IEEE 1st Conference on Requirements Engineering, ICRE 1994*, pp. 154–158 (1994)
- [20] Sommerville, I., Sawyer, P.: *Requirements engineering*. Worldwide Series in Computer Science. Wiley, Chichester (1997)
- [21] Mostow, J.: Towards better models of the design process. *AI Magazine* 6, 44–57 (1985)
- [22] Hoh, P.: Multi-Criteria Preference Analysis for Systematic Requirements Negotiation. In: *26th Annual International Computer Software and Applications Conference*, Oxford, England, p. 887 (2002)
- [23] Boehm, B., Bose, P., Horowitz, E., Ming-June, L.: Software requirements as negotiated win conditions. In: *1st International Conference on Requirements Engineering*, USA, pp. 74–83 (1994)
- [24] Karlsson, J., Olsson, S., Ryan, K.: Improved Practical Support for Large-scale Requirements Prioritizing. *Journal of Requirements Engineering*, 51–60 (1997)

- [25] Moisiadis, F.: The Fundamentals of Prioritising Requirements Systems Engineering. In: Test & Evaluation Conference, Sydney, Australia (2002)
- [26] Nuseibeh, B., Kramer, J., Finkelstein, A.: A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering* 20, 760–773 (1994)
- [27] Lamsweerde, A.v., Letier, E.: Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering, Special Issue on Exception Handling* 26(10), 978–1005 (2000)
- [28] Robinson, W.N., Volcov, S.: Conflict Oriented Requirements Restructuring, Working Paper CIS-96-15 (1996)
- [29] Robinson, W.N., Volkov, S.: Supporting the Negotiation Life-Cycle. *Communications of the ACM*, 95–102 (1998)
- [30] Easterbrook, S.M.: Resolving Requirements Conflicts with Computer-Supported Negotiation. In: Jirotko, M., Goguen, J. (eds.) *Requirements Engineering: Social and Technical Issues*, pp. 41–65. Academic Press, London (1994)
- [31] Rolland, C., Prakash, N.: Bridging the gap between Organizational needs and ERP functionality. *Requirements Engineering Journal* 5 (2000)
- [32] Rolland, C., Salinesi, C., Etien, A.: Eliciting Gaps in Requirements Change. *Requirements Engineering Journal* 9, 1–15 (2004)

Hiding Frequent Patterns under Multiple Sensitive Thresholds

Ya-Ping Kuo, Pai-Yu Lin, and Bi-Ru Dai

Department of Computer Science and Information Engineering, National Taiwan
University of Science and Technology, Taipei, Taiwan. R.O.C.
{m9515063,m9615082}@mail.ntust.edu.tw, brdai@csie.ntust.edu.tw

Abstract. Frequent pattern mining is a popular topic in data mining. With the advance of this technique, privacy issues attract more and more attention in recent years. In this field, previous works based hiding sensitive information on a uniform support threshold or a disclosure threshold. However, in practical applications, we probably need to apply different support thresholds to different itemsets for reflecting their significance. In this paper, we propose a new hiding strategy to protect sensitive frequent patterns with multiple sensitive thresholds. Based on different sensitive thresholds, the sanitized dataset is able to highly fulfill user requirements in real applications, while preserving more information of the original dataset. Empirical studies show that our approach can protect sensitive knowledge well not only under multiple thresholds, but also under a uniform threshold. Moreover, the quality of the sanitized dataset can be maintained.

Keywords: Privacy, frequent pattern hiding, multiple threshold, sensitive knowledge, security, data sanitization.

1 Introduction

Frequent pattern and association rule mining play the important roles in data mining [1]. By this technique, we can discover interesting but hidden information from database. This technique has been applied to many application domains, such as the analysis of market basket, medical management, stock, environment, business, etc., and brings great advantages. However, most database owners are unwilling to supply their datasets to analysis, since some sensitive information or private commercial strategies are at the risk of being disclosed from the mining result. Therefore, although many benefits can be provided by this technique, it causes new threats to privacy and security. For above reason, the database should be processed before releasing so that it can contain the most of original non-sensitive knowledge and the least of sensitive information for the owner. Intuitively, the database owner can permit only partial access of dataset for analysis or directly remove all sensitive information from the mining result of database. However, it is possible that the adversary still can infer sensitive itemsets or high-level items from non-sensitive patterns or low-level items. For example, suppose that $\{1\}$ is the sensitive pattern and the set of all frequent patterns

are $\{\{1\}, \{1, 2\}, \{2, 3\}\}$. If we directly remove $\{1\}$ and release $\{\{1, 2\}, \{2, 3\}\}$, the adversary may still be able to infer that $\{1\}$ is frequent. That is because of the monotonic property of frequent patterns, which means that all non-empty subsets of a frequent pattern must be frequent. Hence the challenge is how to protect sensitive information from being attacked by inference.

1.1 Motivations

In this paper, we focus on the problem of hiding sensitive frequent itemsets from a transaction database. The motivation and the importance of hiding sensitive itemsets have been well explained in [2] as stated below. Suppose that most people who purchase milk usually also purchase Green paper. If the Dedtrees paper company mines this rule from the database of a supermarket and issues a coupon, “if you buy the Dedtrees paper, you will get a 50 cents off discount of one milk,” then the sales of Green paper will be reduced by the above commercial strategy. For this reason, the Green paper company would not like to provide a lower price to the supermarket. On the other hand, the Dedtrees paper company has already achieved its goal, and is unwilling to provide a lower price to supermarket anymore. Then, the supermarket will suffer serious losses. Hence the database should be sanitized for such sensitive information before releasing.

Most of previous sanitization algorithms only use one user-predefined support threshold without considering the following issues. First, using a uniform support threshold with different patterns is not always reasonable in real life. For instance, the supports of high price or the latest products, such as computers, are intrinsically lower than those of general or common products, such as water, but it does not imply the latter ones are more significant than the former ones. If we decrease the supports of all sensitive itemsets to be smaller than the same threshold, it may cause some itemsets are overprotected and some are not protected sufficiently. Furthermore if the support threshold used by the adversary in mining is smaller than the one used in hiding, the released database will disclose all sensitive information. On the contrary, if the support threshold which is used for hiding is too small, the released database is possible to lose too much information and becomes useless for subsequent mining. In addition, if the general items and the particular items have similar frequencies in database, the adversary will infer that some sensitive knowledge has been hidden. Therefore, based on the consideration of both privacy protection and information preservation, it is important to assign each itemset a particular threshold. For the above reasons, an algorithms using a disclosure threshold has been proposed [17]. It decreases the supports of sensitive patterns according to their distribution in database and uses a disclosure threshold directly to control the balance between privacy and knowledge discovery. However, the method does not consider the characteristics of different sensitive itemsets in different applications or the personalized requirements of different users. It totally relies on the distribution of database to do the same degree of sanitization with infrequent sensitive patterns and more frequent sensitive patterns.

1.2 Contributions

In this paper, we propose a new strategy, which combines the sanitization algorithm and the concept of multiple support thresholds [14], to solve the problem which is mentioned above. Before sanitizing, the database owner can specify the support threshold, called *sensitive threshold*, for each sensitive pattern based on his/her domain knowledge. Then our algorithm will decrease the support of each sensitive pattern to be below its sensitive threshold, respectively. Under multiple thresholds, the database owner can directly decide the sanitization degree of different patterns hence the protected database can much more satisfy the demand of the database owner. Consequently, the proposed strategy is able to reduce the probability of privacy breach and preserve as much information as possible.

The main contributions of this paper are as follows: (1) a new hiding strategy with multiple sensitive thresholds, which is more applicable in reality, is suggested; (2) the proposed algorithm can achieve better privacy protection and information preservation; (3) The new metrics are presented to measure performance for hiding frequent patterns under multiple sensitive thresholds because of the difference between under multiple thresholds and a uniform one, while the sets of the patterns which need to be hidden by user-predefined are the same.

The rest of this paper is organized as follows. The preliminary knowledge is stated in Section 2. In Section 3, we introduce our sanitization framework, the whole sanitization process, and some techniques which improve performance and efficiency. Some related hiding algorithms are reviewed in Section 4. The new metrics, experimental results and discussion are presented in Section 5. In the last part, Section 6 presents our conclusions.

2 Preliminaries

Before presenting our hiding strategy and framework, we introduce the preliminaries of frequent patterns, the transaction database, and the related concepts of privacy and multiple thresholds briefly.

Frequent Pattern and Transaction Database. Let $I = \{1, \dots, n\}$ be a non-empty set of *items*. Each non-empty subset $X \subseteq I$ is called a *pattern* or an *itemset*. A *transaction* is a pair of itemset $t \subseteq I$ with a unique identifier T_i , called the *transaction identifier* or *TID*. A *transaction database* $D = \{T_1, \dots, T_N\}$ is a set of transactions, and its size is $|D| = N$. We assume that the itemsets and the transactions are ordered in *lexicographic order*. A transaction t *supports* X , if $X \subseteq t$. Given a database D , the *support* of an itemset X , denoted $sup(X)$, is the number of transactions that support X in D . The *frequency* of X is $sup(X)/|D|$. An itemset X is said to be *frequent* if $sup(X)/|D|$ is larger than the user-predefined *minimum support*, denoted as $minsup$, $0 \leq minsup \leq 1$.

Sensitive Itemset, Sensitive Threshold, and Sensitive Transaction. Let D be a transaction database, $\mathcal{FP}$ be a set of frequent patterns, and $\{sp_1, \dots, sp_i\}$

Table 1. The summarization of notations used in the paper

I and X	I is the set of all items; X is an itemset, $X \subseteq I$
(T_i, t)	The transaction itemset t with its TID T_i
$sup(X)$	The support of X
$minsup$	The user-predefined minimum support threshold
$sp_1, \dots, sp_i \in \mathcal{SP}$	The set of patterns that need to be hidden
$\mathbf{P_s}$	The set of sensitive patterns which can infer any patterns in $\mathcal{SP}$
$\mathbf{T_s}(X)$	The set of sensitive transactions of X
$st(X)$	The sensitive threshold of X
TP_k	The unique identifier of template
SPC	The number of sensitive patterns covered of a template
MC	The minimal count of the transactions need to be modified

$\in \mathcal{SP}$ be a set of patterns that need to be hidden based on some security requirements. A set of frequent patterns which are able to infer any patterns in $\mathcal{SP}$, denoted as $\mathbf{P_s}$, is said to be *sensitive*. $\sim \mathbf{P_s}$ is the set of non-sensitive frequent patterns such that $\mathbf{P_s} \cup \sim \mathbf{P_s} = \mathcal{FP}$. As long as a transaction supports any itemsets, it is said to be sensitive, denoted as $\mathbf{T_s}$, and the set of sensitive transactions of X is denoted as $\mathbf{T_s}(X)$. The support threshold used for hiding is named *sensitive threshold*. The sensitive threshold of a sensitive pattern X is denoted as $st(X)$.

3 The Template-Based Sanitization Process

The main goal of this work is to hide sensitive information in database so that the frequent itemset mining result of new released dataset will not disclose any sensitive patterns. The challenge of this problem is to find out the balanced solution between the privacy requirement and the information preservation. We suggest assigning different sensitive threshold to each sensitive itemset to minimize the side effects with the dataset. Formally, the problem definition is stated as follows:

Frequent Pattern Hiding with Multiple Sensitive Thresholds. Given a database D and the set of patterns to be hidden, $\{sp_1, \dots, sp_i\} \in \mathcal{SP}$, with their sensitive thresholds, $st(sp_1), \dots, st(sp_i)$, the problem is how to transform D to D' such that $\mathbf{P_s}$ will not be mined from D' , and $\sim \mathbf{P_s}$ can still be contained in D' . Finally, D' can be released without violating the privacy concern.

In our sanitization process, we remove some items for each sensitive itemset from its corresponding sensitive transactions. Since a uniform sensitive threshold is usually not suitable for real cases, we apply the concept of multiple sensitive thresholds for hiding sensitive itemsets so that the itemsets with higher occurrences in reality can preserve more information. Moreover, the itemsets with lower occurrences in reality can reach better protection. Note that we will not focus on the determination of sensitive thresholds since they largely depend on applications and users requirements. Database owners can decide the sensitive threshold based on existing schemes [8][14][18] or any preferred settings.

Table 2. The sensitive pattern table and the corresponding template table

Sensitive Pattern Table		Template Table			
SP	$Count$	$victim$	UCP	SPC	MC
$\{3\}$	3	TP ₁	3	1	3
$\{2, 3\}$	5	TP ₂	2	2,3	1
$\{1, 3, 4\}$	6	TP ₃	3	2,3	2
		TP ₄	1	1,3,4	1
		TP ₅	3	1,3,4	2
		TP ₆	4	1,3,4	1
		TP ₇	3	1,2,3,4	3

3.2 Sensitive Pattern Table

There are two attributes contained in the sensitive pattern table, each sensitive itemset and its *Count*, as shown in Table 2. The *Count* of a sensitive pattern indicates that the minimal number of support which is required to be decreased will make this pattern to be infrequent. Based on multiple sensitive thresholds, we propose the lemma of *Count* as follow:

Lemma 1. *Given a sensitive pattern sp_i , the minimal number of transactions that should be sanitized for hiding this pattern is computed as $sp_i.Count = \lfloor sup(sp_i) - st(sp_i) + 1 \rfloor$*

Proof. To hide a sensitive pattern sp_i , its support, $sup(sp_i)$, should be decreased to be below its sensitive threshold, $st(sp_i)$. Hence removing some victim items contained in sp_i from the corresponding sensitive transactions will make sp_i to be infrequent. Let $sp_i.Count$ be the minimal number of sanitized transactions as $sup(sp_i) < st(sp_i)$. Because $sup(sp_i) - sp_i.Count < st(sp_i)$, and then $sp_i.Count > sup(sp_i) - st(sp_i)$. Therefore $sp_i.Count = \text{the interger part of } ((sup(sp_i) - st(sp_i)) + 1) = \lfloor sup(sp_i) - st(sp_i) + 1 \rfloor$ $\square$

3.3 Template Table

The initial template table should be built according to the sensitive pattern table, as depicted in Table 2. A template is represented in the form: $\langle TPID, victim, UCP, SPC, MC \rangle$, where TPID is the *template unique identifier*, and *victim* is the chosen item that is considered to be removed from the corresponding sensitive transactions. For a sensitive itemset with length k , there are k items that can be *victims*. Hence we can generate k templates with different victims. Take $\{1, 3, 4\}$ as example, three templates with the *victims*, $\{1\}$, $\{3\}$, and $\{4\}$ are produced, respectively. The *UCP* of a template represents the itemset must be contained in the corresponding transactions and it is the union of all corresponding sensitive patterns which can be sanitized by this template. It means that if the *victim* is deleted from the corresponding sensitive transactions which contain the *UCP*, the support of each corresponding sensitive pattern is decreased. For instance in Table 2, TP₁ for hiding $\{3\}$ is to delete $\{3\}$ from

the transactions containing $\{3\}$; TP_3 for hiding $\{2, 3\}$ and $\{3\}$ is to delete $\{3\}$ from the transaction containing $\{2, 3\}$, etc. The *SPC*, stands for the number of the sensitive patterns which can be sanitized by this template. For example, the TP_3 in Table 2 can hide two sensitive patterns $\{3\}$ and $\{2, 3\}$ at the same time, so its *SPC* is 2. The *MC* indicates the minimal number of the support should be decreased, such that all corresponding sensitive patterns of this template are hidden. Hence the *MC* is the maximum *Count* among all corresponding sensitive patterns of this template. For instance, the *MC* of TP_3 in Table 2 is $\max\{3, 5\} = 5$.

Not only are those templates introduced above, but also we generate joint templates to cover more sensitive patterns. If any two templates have the same *victim*, and their *UCP* do not contain each other, we can join them to be a new template. The *UCP* of the new joint template is the union of the *UCPs* of all combined templates. Then the *SPC* and the *MC* are computed according all corresponding sensitive patterns. As shown in Table 2, TP_3 and TP_5 can be combined to generate TP_7 . The *UCP*, the union of $\{2, 3\}$ and $\{1, 3, 4\}$, is $\{1, 2, 3, 4\}$. Then the *SPC* of TP_7 will be 3 because removing $\{3\}$ from transaction containing $\{1, 2, 3, 4\}$ can decrease the supports of three sensitive patterns, $\{3\}$, $\{2, 3\}$, and $\{1, 3, 4\}$. The *MC* of TP_7 is $\max\{3, 5, 6\} = 6$. Consequently, TP_7 becomes a better choice than TP_3 and TP_5 because the *SPC* in it is larger than the others, thus TP_7 can hide more patterns at the same time. We use the hash table to avoid generating the same template with existing ones, and transfer the pattern index from binary to decimal to be the hash key. We can compute the *SPC* and the *MC* of all templates refer to the sensitive pattern table, and the computation algorithm is similar in [5].

3.4 Choosing Strategy and Updating Process

Based on the essence of our hiding strategies - “minimizing side effect”, we choose the template having the largest *SPC* at each round. If there exists more than one template having the same *SPC*, the template with the smallest *MC* is selected. If there still exists more than one, we choose the template which has the *victim* with the lowest support in the database. Finally, if the tie is still not solved, a random choice will be picked. After choosing the template, the corresponding sensitive transactions are found out by the transaction index. If the number of the sensitive transactions is larger than the *MC* of the chosen template, we choose the first *MC* shortest transactions to move to action table for sanitizing; otherwise all corresponding sensitive transactions will be sanitized. By the number of sanitized transactions, the *Count* and the *MC* are recomputed. If some patterns are hidden by this template, the *SPC* and the *UCP* should be changed. As the *SPC* of a template becomes zero, we remove this template from template table. Lastly, the TID of sanitized transactions are removed from the transaction index of the corresponding sensitive patterns of this template. In order to achieve better hiding performance, the number of the victim items in one transaction is not restricted.

3.5 Performance and Efficiency Improvement

In order to promote the performance and efficiency of our framework, we propose the concept of revised border itemsets for reducing redundant work. Because of the monotonic property of frequent patterns, hiding a sensitive pattern will hide all supersets of this pattern. Hence in the sanitization process, we merely need to hide the sensitive patterns which have no sensitive subsets. Such itemset is said to be a border itemset. For instance, if the $\{1, 2, 3\}$, $\{1, 3\}$ and $\{1\}$ are three sensitive patterns, then $\{1\}$ is a border itemset but $\{1, 2, 3\}$, $\{1, 3\}$ are not. As long as the border itemsets of the sensitive patterns have been hidden, we can protect all sensitive information. However this technique cannot be applied to the situation of hiding frequent patterns with multiple sensitive thresholds. While the sensitive threshold of the super-itemset is smaller than that of the itemset, focusing on hiding the border itemset cannot guarantee the protection of all sensitive frequent patterns. Therefore, we take all the itemsets which have no sensitive subsets with lower support thresholds than themselves to be the *revised border itemset*. In the above example, if the sensitive thresholds of $\{1,2,3\}$, $\{1,3\}$, and $\{1\}$ are 2, 4, and 3, respectively, then $\{1\}$ and $\{1,2,3\}$ are revised border itemset. If the revised border itemsets are hidden, all the sensitive knowledge will be protected. In addition, the techniques of pattern index [5] and inverted file are also applied in our framework for increasing efficiency.

4 Related Works

The problem of hiding frequent patterns and association rules was proposed in [9] firstly. The authors proved that finding an optimal sanitization for hiding frequent patterns is an NP-hard problem and proposed a heuristic approach by deleting items from transactions in the database to hide sensitive frequent patterns. In recent year, more and more researchers start paying attention to privacy issues. The consequent approaches can be classified into two categories: data modification, and data reconstruction.

Data Modification. The main idea of this group is to alter original database such that the sensitive information is not able to be mined in new database. So as to decrease the support or confidence of sensitive rules below the user-predefined threshold, these algorithms choose some items as victims and delete or insert them in some transactions [6]. In [17], the authors present a novel approach using a disclosure parameter instead of the support threshold to directly control the balance between privacy requirement and information preservation. Based on the disclosure parameter, the support of each sensitive pattern is decreased by the same proportion. The proposed IGA algorithm groups sensitive patterns first, and then chooses the victim items based on the minimal side effects for database. In [7], the border-based concept was proposed to evaluate the impact of any modification on the database efficiently. The quality of database and relative frequency of each frequent itemset can be well maintained by greedily selecting the modifications with minimal side effect. In [4], the authors

propose an algorithm which can be secure against forward inference attack. By multiplying the original database matrix and a sanitization one together, the method raises more efficiency. Most of recent works focus on the minimal side effects on database [5][10]. These methods minimize the number of sanitized transactions or items to limit the side effects on database respectively.

Data Reconstruction. The motivation of this group is that the previous database-based modification methods spend more time on scanning the database and we can not directly control the information on the released database. Hence the data-reconstruction group uses knowledge-based method to directly reconstruct the released dataset containing the knowledge that the database owner wants to preserve. In general, the set of frequent patterns in original dataset is regarded as knowledge. The concept of “inverse frequent set mining problem” was first proposed in [11], and was proved to be an NP-hard problem. Consequently, most researches apply this concept on privacy issues and algorithm benchmarks [12]. In [3], the authors proposed a constraint itemset lattice mining technique to automatically generate a sample dataset which can be released for sharing. It indicates that if there exists a feasible support set of all itemsets, they can generate the new database containing the same frequent itemsets by one-to-one mapping. In [13], the authors proposed the FP-tree-based method for inverse frequent set mining, and the new database exactly satisfies the whole given constraints. However, this method does not provide complete and well hiding. It only controls the support counts of the non-sensitive patterns to be the same as before, but the frequencies do not satisfy the original constraint. For this issue, the major problem at present is how to find the feasible support set which has compatible dataset.

In addition, the concept of multiple support thresholds is first proposed in [14] owing to the observed phenomenon that support thresholds of different itemsets are not always uniform in the reality. We take some of existing specification of multiple support thresholds to be the benchmark of our sanitization framework, and they will be introduced in next section.

5 Experiments

In this section, the performance, efficiency and scalability of our proposed sanitization process are shown. We use the support constraint [18] and the maximum constraint [8] to assign different sensitive threshold to each sensitive pattern. In addition, in order to compare our sanitization process with the Item Grouping Algorithm [20], IGA for short, the same disclosure threshold of IGA is used by our framework to show performance for the situation of uniform sensitive threshold. These constraints are described as follow:

Disclosure Thresholds: It is devised to compare our method with IGA under uniform support thresholds. Therefore, the sensitive thresholds are set to be the same with IGA as follows:

$$st(X) = sup(X) \times \alpha$$

where α is the same as the disclosure threshold used by IGA.

Support Constraints: It is similar in [18]. We first partition the support range in database into the bin number of intervals. Each interval has the same number of items so that each bin, B_i , contains every items in the i th interval. Next, the support constraints with the schemas which are made up of all possible combinations of bins were generated. And the support threshold of the support constraint $SC_k(B_1, \dots, B_r) \leq \theta_k$ is defined as follows:

$$\theta = \min\{\gamma^{k-1} \times S(B_i) \times \dots \times S(B_r), 1\}$$

where $S(B_i)$ denotes the smallest item support for the B_i , and γ is an integer larger than 1. A large value of γ can be used to slow down the rapid decrease of $S(B_1) \times \dots \times S(B_r)$. We can vary the value of γ to generate different support constraints.

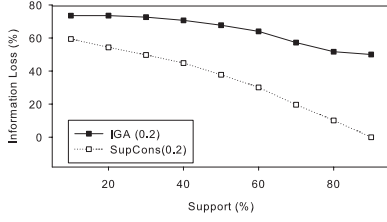
Maximal Constraints: we use the same formula in [14] to assign the support threshold of each item:

$$st(X) = \begin{cases} sup(X) \times \sigma & \text{if } sup(X) \times \sigma > minsup, \\ minsup & \text{otherwise.} \end{cases}$$

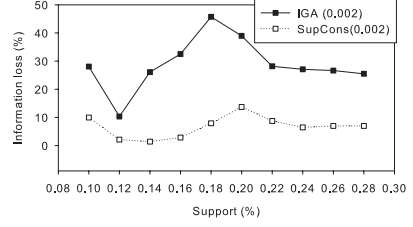
where $0 \leq \sigma \leq 1$, and $sup(i)$ denotes the support of item i in the dataset. If σ is set to be zero, the support thresholds of all items are the same, then this case becomes the same as the uniform one.

We use two real datasets, accidents [19] and kosarak with different characteristics, to compare our method with IGA when applying disclosure threshold. The accidents dataset is donated by Karolien Geurts and contains traffic accident data, and the kosarak dataset was provided by Ferenc Bodon and contains click-stream data from a Hungarian on-line news portal. On the other hand, considering the time complexity of mining with multiple sensitive thresholds, without loss of generality, two smaller real datasets chess and mushroom [20] are used to evaluate the performance of our hiding approach. These four datasets are commonly used for performance evaluation of various association rule mining algorithms. Their characteristics are shown in Table 3. For each original dataset, we first execute Apriori algorithm to mine the supports of all items and use them to establish the settings of item support thresholds and support constraints. Next, according to different applications, we apply algorithms Apriori, Apriori-like [18], and Adaptive-Apriori [8] to mine the frequent patterns under the uniform threshold, maximal constraint, and support constraints, respectively. Subsequently, the sensitive patterns to be hidden are selected randomly to simulate the application-oriented sensitive information.

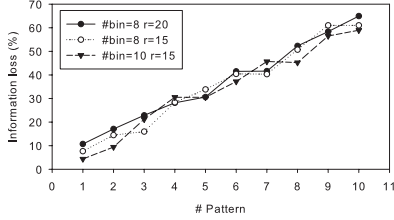
Our testing environment consists of a 3.4 GHz Intel(R) Pentium(R) D processor with 1 GB of memory running a Window XP operating system. All recorded execution times include the CPU time and the I/O time.



(a) accident dataset



(b) kosarak dataset

Fig. 2. *IL* with disclosure thresholds**Fig. 3.** *IL* with support constraints on chess dataset**Table 3.** The characteristics of each dataset

	trans.	Items
Accidents	340,184	572
Kosarak	990,002	41,270
Chess	3,196	75
mushroom	8,124	119

5.1 Metrics

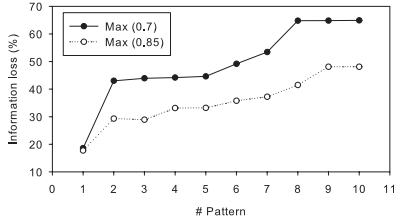
Two measures, information loss and hiding failure, are adopted to evaluate the performance of our hiding strategies. *Information loss (IL)* is the percentage of non-sensitive patterns which are hidden in the sanitization process, as shown in the following equation:

$$IL = \frac{((|\mathcal{FP}| - |\mathbf{P}_s|) - (|\mathcal{FP}'| - |\mathbf{P}'_s|))}{(|\mathcal{FP}| - |\mathbf{P}_s|)}$$

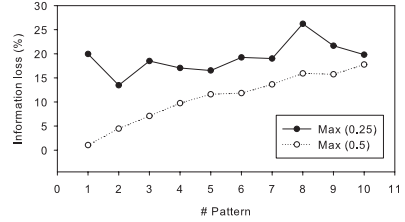
where $|\mathcal{FP}|$ and $|\mathbf{P}_s|$ are the number of frequent patterns in original database, D , and the number of sensitive patterns in D , respectively, and $|\mathcal{FP}'|$ and $|\mathbf{P}'_s|$ are the number of frequent patterns in new database, D' , and the number of sensitive patterns in D' , respectively. *Hiding failure (HF)* is the percentage of sensitive patterns remaining in D' after sanitization, represented as follows:

$$HF = \frac{|\mathbf{P}'_s|}{|\mathbf{P}_s|}$$

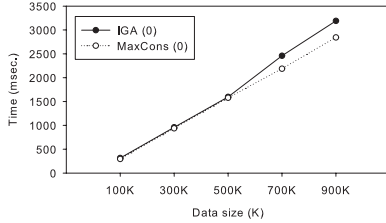
Under the uniform support threshold, $\mathbf{P}_s$ contains all the supersets of any patterns in $\mathcal{SP}$. However under the multiple support thresholds, the subset of a frequent pattern may not be frequent. Hence $\mathbf{P}_s$ should only contain the supersets which have larger sensitive thresholds than those of its sensitive subsets of any pattern in $\mathcal{SP}$.



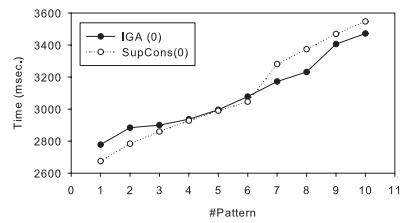
(a) chess dataset



(b) mushroom dataset

Fig. 4. *IL* under maximal constraints

(a) the efficiency



(b) the scalability

Fig. 5. The comparison with IGA on the kosarak dataset

5.2 Performance

Firstly, our framework is compared with IGA to evaluate its performance under the uniform support threshold. We use the disclosure thresholds 0.2 and 0.002 to hide sensitive patterns in accidents and kosarak datasets. The sensitive patterns are randomly selected from the frequent patterns which have the support being larger than 20% and 0.2%, respectively. Subsequently we mine the frequent patterns in new dataset by the support thresholds 10% to 90%, 0.1% to 0.28%, respectively. The result of *IL* is shown in Fig. 2. The trend of the *IL* is highly related the characteristic of the experiment dataset. We can observe that our method reaches better information preservation. Most of *HF* are zero, except for the case when the support threshold being 10% of accidents, the *HF* of IGA is 0.0057% and our method is 0.325%.

For the capability of hiding with multiple thresholds, we evaluate performance under support constraints and under the maximal constraints. Under the support constraints, we evaluate performance from hiding 1 pattern to 10 patterns. All the sensitive patterns are chosen randomly. We compare the results of $\gamma = 15, 20$ under a fixed bins number 8, and bins number = 8, 10 with a fixed γ . The result of *IL* is shown in Fig. 3. All *HF*s are zero. We can observe that the *IL* is not affected by the bins number and γ , but probably relies on the distribution of the dataset and the chosen sensitive patterns. Finally, performances under maximal constraints are measured. Under the maximum constraints, we evaluate performance from hiding 1 pattern to 10 patterns. All the sensitive patterns are chosen randomly. Different parameter settings, including $\sigma = 0.7, 0.85$ and $\sigma = 0.25, 0.5$ on the chess and mush-

room datasets are examined, respectively. The result of IL is shown in Fig. 4(a) and 4(b). Since supports of chosen sensitive patterns of chess dataset are higher and more items are deleted to hide such patterns, the IL is higher. All HF s are zero. We can observe that the IL will increase along with the decrease of γ and the increase of the number of hidden sensitive patterns.

5.3 Efficiency and Scalability

We estimate the efficiency and scalability of our method compared with IGA on the size of the database and the number of the sensitive patterns. The disclosure parameter of IGA is set to be zero, and the same is α of our method. The zero value means hiding completely.

We vary the size of dataset from 100K to 900K on hiding six mutually exclusive frequent patterns with length 2-7. The result is illustrated in Fig. 5(a). Next, the number of the hidden sensitive patterns is varied from 1 pattern to 10 patterns. All the patterns are chosen randomly. The result is illustrated in Fig. 5(b). We can observe that the execution time is linear with the size of the database and the number of sensitive patterns. Note that our method achieves good scalability as IGA while attaining better information preservation and providing additional capability of hiding with multiple sensitive thresholds.

6 Conclusions

In this paper, we introduce the concept of frequent pattern hiding under multiple sensitive thresholds. A new hiding strategy of multiple sensitive thresholds is proposed. The hiding strategy is more applicable in the practical applications. Considering the properties of the frequent patterns under multiple sensitive thresholds, we suggest the revised border-based method to reduce the redundant work on hiding, and used the inverted file and the pattern index to speed up the update in our framework.

We empirically validated the performance, efficiency and scalability of our method by using a series of experiments. In all of these experiments, we took into account the uniform support threshold, multiple support thresholds with support constraints, and multiple support thresholds under maximal support constraints. The results of our experiments reveal that our method is effective and achieves significant improvement over the IGA with the uniform support thresholds. Furthermore, we can hide sensitive knowledge of the dataset with multiple sensitive thresholds.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining Associations Rule Between Sets of Items in Massive Database. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 207–216 (1993)
2. Clifton, C., Marks, D.: Security and Privacy Implication of Data Mining. In: ACM SIGMOD Workshop on Data Mining and Knowledge Discovery, pp. 15–19 (1996)

3. Chen, X., Orlowska, M., Li, X.: A New Framework of Privacy Preserving Data Sharing. In: Proc. of IEEE 4th Int. Workshop on Privacy and Security Aspects of Data Mining, pp. 47–56 (2004)
4. Wang, E.T., Lee, G., Lin, Y.T.: A Novel Method for Protecting Sensitive Knowledge in Association Rules Mining. In: Proc. of the 29th Annual Int. COMPSAC, vol. 1, pp. 511–516 (2005)
5. Wu, Y.H., Chiang, C.M., Chen, A.L.P.: Hiding Sensitive Association Rules with Limited Side Effects. *IEEE Transactions on Knowledge and Data Engineering* 19(1), 29–42 (2007)
6. Verykios, V.S., Elmagarmid, A., Bertino, E., Saygin, Y., Dasseni, E.: Association Rule Hiding. *IEEE Transactions on Knowledge and Data Engineering* 16(4), 434–447 (2004)
7. Xingzhi, S., Yu, P.S.: A Border-Based Approach for Hiding Sensitive Frequent Itemsets. In: Proc. of 5th IEEE Int. Conf. on Data Mining, pp. 426–433 (2005)
8. Lee, Y.C., Hong, T.P., Lin, W.Y.: Mining Association Rules with Multiple Minimum Supports Using Maximum Constraints. *Int. Journal of Approximate Reasoning on Data Mining and Granular Computing* 40(1–2), 44–54 (2005)
9. Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, V.: Disclosure Limitation of Sensitive Rules. In: Proc. of the IEEE Knowledge and Data Exchange Workshop, pp. 45–52 (1999)
10. Gkoulalas-Divanis, A., Verykios, V.S.: An Integer Programming Approach for Frequent Itemset Hiding. In: Proc. of Int. Conf. on Information and Knowledge Management, pp. 748–757 (2006)
11. Mielikainen, T.: On Inverse Frequent Set Mining. In: Proc. of the 2nd IEEE ICDM Workshop on Privacy Preserving Data Mining (2003)
12. Wu, X., Wu, Y., Wang, Y., Li, Y.: Privacy-Aware Market Basket Data Set Generation: A Feasible Approach for Inverse Frequent Set Mining. In: Proc. 5th SIAM Int. Conf. on Data Mining (2005)
13. Guo, Y.: Reconstruction-Based Association Rule Hiding. In: Proc. of SIGMOD 2007 Ph.D. Workshop on Innovative Database Research (2007)
14. Liu, B., Hsu, W., Ma, Y.: Mining Association Rules with Multiple Minimum Supports. In: Proc. of the 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 337–341 (1999)
15. Wang, K., Fung, B.C.M., Yu, P.S.: Template-Based Privacy Preservation in Classification Problems. In: Proc. - IEEE Int. Conf. on Data Mining, pp. 466–473 (2005)
16. Dwork, C.: Ask a Better Question, Get a Better Answer: A New Approach to Private Data Analysis. In: Schwentick, T., Suciu, D. (eds.) *ICDT 2007*. LNCS, vol. 4353, pp. 18–27. Springer, Heidelberg (2006)
17. Oliveira, S.R.M., Zaiane, O.R.: A Unified Framework for Protecting Sensitive Association Rules in Business Collaboration. *Int. J. of Business Intelligence and Data Mining* 1(3), 247–287 (2006)
18. Wang, K., He, Y., Han, J.: Pushing Support Constraints into Association Rules Mining. *IEEE Transactions on Knowledge and Data Engineering* 15(3), 642–658 (2003)
19. Geurts, K., Wets, G., Brijs, T., Vanhoof, K.: Profiling High-Frequency Accident Locations Using Association Rules. In: Proc. of the 82th Annual Transportation Research Board, p. 18 (2003)
20. Blake, C.L., Merz, C.J.: UCIRepository of machine learning databases. University of California, Dept. of Inf. and CS, Irvine (1998), <http://www.ics.uci.edu/~mllearn/MLRepository.html>

BSGI: An Effective Algorithm towards Stronger l -Diversity

Yang Ye¹, Qiao Deng², Chi Wang³, Dapeng Lv³, Yu Liu³, and Jianhua Feng³

¹ Institute for Theoretical Computer Science, Tsinghua University
Beijing, 100084, China
yey05@mails.tsinghua.edu.cn

² Department of Mathematical Science, Tsinghua University
Beijing, 100084, China
dengxinqiao@163.com

³ Department of Computer Science, Tsinghua University
Beijing, 100084, China
{wangchi05, lvdp05, liuyu-05}@mails.tsinghua.edu.cn
fengjh@tsinghua.edu.cn

Abstract. To reduce the risk of privacy disclosure during personal data publishing, the approach of anonymization is widely employed. On this topic, current studies mainly focus on two directions: (1) developing privacy preserving models which satisfy certain constraints, such as k -anonymity, l -diversity, etc.; (2) designing algorithms for certain privacy preserving model to achieve better privacy protection as well as less information loss. This paper generally belongs to the second class. We introduce an effective algorithm “*BSGI*” for the widely accepted privacy preserving model: l -diversity. In the meantime, we propose a novel interpretation of l -diversity: Unique Distinct l -diversity, which can be properly achieved by *BSGI*. We substantiate it’s a stronger l -diversity model than other interpretations. Related to the algorithm, we conduct the first research on the optimal assignment of parameter l according to certain dataset. Extensive experimental evaluation shows that Unique Distinct l -diversity provides much better protection than conventional l -diversity models, and *BSGI* greatly outperforms the state of the art in terms of both efficiency and data quality.

Keywords: Privacy preservation, *BSGI*, k -anonymity, l -diversity, Unique-Distinct l -diversity.

1 Introduction

With the development of internet, more and more data on individuals are being collected and published for scientific and business uses. To reduce the risk of privacy disclosure during such publishing, the approach of anonymization is widely used. Removing the attributes that explicitly identify an individual, (e.g., name, social security number) from the released data table is necessary but insufficient, because a set of Quasi-identifying (QI) attributes (e.g., date of birth, zip code, gender) can be linked with public available datasets to reveal personal identity. To counter such “link attack”, *P. Samaritan* and *L. Sweeney* proposed the model of k -anonymity[1,2,3,4]. K -anonymity requires each

tuple in the published table to be indistinguishable from at least $k - 1$ other tuples on QI values. Tuples with the same QI values form an *equivalence class*. Thereby k -anonymity reduces the *identity disclosure* risk to no more than $1/k$.

However, since k -anonymity does not take into account the sensitive attribute (SA), namely, the attribute containing privacy information (e.g., disease, salary), it may be vulnerable to *sensitive attribute disclosure*[5]. [5] presents two kinds of possible attacks that k -anonymity cannot prevent: *homogenous attack* and *background knowledge attack*, then proposes a new model: l -diversity to counter such attacks. l -diversity ensures each equivalence class contains at least l “well-represented” SA values, thereby reduces the risk of sensitive attribute disclosure to no more than $1/l$.

Current algorithms for l -diversity are generally derived from algorithms for k -anonymity. As proved in [5], any algorithm for k -anonymity, like *hierarchy-base* algorithm *Incognito*[13] and *partition-based* algorithm *Mondrian*[14], can be transformed easily to algorithm for l -diversity, just by changing the condition in each checking phase from k -anonymity to l -diversity. However, since k -anonymity algorithms do not take into account the distribution of SA values at all, which is the essence of l -diversity, the derived l -diversity algorithms may generate great and unnecessary information loss. In fact, our experiments in Section 6 reveal that *Incognito* for l -diversity is almost impractical for low efficiency and data quality while *Mondrian* for l -diversity drops behind our algorithm largely in both terms.

In [8], a new model, “*Anatomy*” was proposed for privacy preserving. Although *Anatomy* fails to prevent identity disclosure because of no generalization on QI attributes, its ideas inspire us to propose an algorithm specially designed for l -diversity: *BSGI*. Since the implementation of l -diversity largely relies on the distribution of SA values, an intuitive but most effective inspiration is to firstly “bucketize” the tuples according to their SA values, then recursively “select” l tuples from l distinct buckets and “group” them into an equivalence class. As for the residual tuples, “incorporate” each of them into a proper equivalence class. The resulted table will satisfy l -diversity perfectly.

For instance, for the disease information table: Table 1, to satisfy 2-diversity, firstly, tuples are bucketized according to the “Disease” attribute and three buckets are formed: $B_1 = \{t_1, t_4\}$, $B_2 = \{t_3, t_5\}$ and $B_3 = \{t_2, t_6, t_7\}$. Here t_i denotes the i^{th} tuple in the table. Secondly, t_1 and t_2 are selected from B_1 and B_3 and grouped. An group (equivalence class) is formed as shown in Table 2.

Continuously, t_3 and t_4 , t_5 and t_6 are selected and grouped (Table 3). Finally, the residual tuple t_7 is incorporated into Group 2, the final published table is created (Table 4).

Detailed discussions about the implementation of the four steps form the mainbody of this paper, together with two natural by-products: the optimal assignment of the parameter l and the stronger l -diversity model: Unique Distinct l -diversity.

The idea of Unique Distinct l -diversity comes from the property of the transformed tables achieved by *BSGI*: without considering the incorporated tuples, each equivalence class contains exactly l distinct SA values, we call such model “Unique Distinct l -diversity” and will further discuss it in this paper.

The rest of this paper is organized as follows. Section 2 gives the basic notations and definitions, including the Unique Distinct l -diversity model. Section 3 and 4 provide the

Table 1. The Original Table

NO.	Name	Gender	Postcode	Age	Disease
1	Alice	F	10075	50	Cancer
2	Bob	M	10075	50	Obesity
3	Carl	M	10076	30	Flu
4	Diana	F	10075	40	Cancer
5	Ella	F	10077	20	Flu
6	Fiona	F	10077	25	Obesity
7	Gavin	M	10076	25	Obesity

Table 2. The First Equivalence Class

Group id.	Gender	Postcode	Age	Disease
1	*	10075	50	Cancer
1	*	10075	50	Obesity

Table 3. The Table after *Bucktizing*, *Sel – ecting* and *Grouping*

Group id	Gender	Postcode	Age	Disease
1	*	10075	50	Cancer
1	*	10075	50	Obesity
2	*	1007*	30-40	Flu
2	*	1007*	30-40	Cancer
3	F	10077	20-25	Flu
3	F	10077	20-25	Obesity

Table 4. The Final Published Table

Group id	Gender	Postcode	Age	Disease
1	*	10075	50	Cancer
1	*	10075	50	Obesity
2	*	1007*	25-40	Flu
2	*	1007*	25-40	Cancer
2	*	1007*	25-40	Obesity
3	F	10077	20-25	Flu
3	F	10077	20-25	Obesity

essential ideas of *BSGI* algorithm, together with the discussion about l 's assignment. Section 5 formally presents the *BSGI* algorithm with further discussions. Section 6 provides the experimental evaluations. Section 7 introduces related work and Section 8 concludes this paper with discussions about future work.

2 Preliminary

2.1 Basic Notations

Let $T = \{t_1, t_2, \dots, t_n\}$ be the table that need to be anonymized. Here $t_i, i = 1, 2, \dots, n$ represents the i^{th} tuple of the table. Each tuple contains a set of Quasi-identifying attribute $\{A_1, A_2, \dots, A_N\}$. Each tuple contains one sensitive attribute S (we will discuss the *single-tuple-multi-SA* case in Section 5). We use $t[A]$ to denote the value of t 's attribute A . Let $T^* = \{t_1^*, t_2^*, \dots, t_n^*\}$ be the anonymized table, where t_i^* is the i^{th} tuple after anonymization. Also $T^* = e_1 \cup e_2 \cup \dots \cup e_m$, where e_i is the i^{th} equivalence class. Let E be the set of equivalence classes. By overriding, we also use $e_i[A_j]$, etc. And $e_i[S]$ denotes the multi-set of e_i 's SA values.

2.2 The Information Loss Metric

In fact, our *BSGI* algorithm does not rely on a certain information loss metric. Any metric that captures the quality of generalization [12, 15, 18] can be adopt by the algorithm. In our experiment, we use the metric proposed by [12], denoted as *IL* metric.

IL metric defines the information loss for categorical and numerical attributes separately. The information loss of a tuple is defined by summing up the loss of all attributes (multiplied by different weights). The total information loss of the whole table is defined by summing up the loss of all tuples.

2.3 *l*-diversity and Unique Distinct *l*-diversity

Definition 1. (*The l-diversity Principle*) An anonymized table is said to satisfy *l*-diversity principle if for each of its equivalence class e , $e[S]$ contains at least l “well-represented” values[5].

According to [5,6], the so called “well-represented” has several interpretations:

1. *Distinct l-diversity*. This interpretation just requires that for each equivalence class e_i , there are at least l distinct values in $e_i[S]$.
2. *Entropy l-diversity*. The entropy of equivalence class e is defined as follows:

$$\text{Entropy}(e) = - \sum_{\text{each distinct } s \in e[S]} P(e, s) \log(P(e, s))$$

Here $P(e, s)$ denotes the proportion that value s takes in $e[S]$. Entropy *l*-diversity requires for each equivalence class e_i , $\text{Entropy}(e_i) \geq \log l$.

3. *Recursive (c,l)-diversity*. Let d be the number of distinct *SA* values in $e[S]$. r_i , $1 \leq r \leq d$, be the number of the i^{th} most frequent *SA* value in $e[S]$. Recursive (c,l) -diversity requires $r_1 < c(r_1 + r_{l+1} + \dots + r_d)$.

Here we propose our interpretation of *l*-diversity:

Definition 2. (*Unique Distinct l-diversity*) An anonymized table is said to satisfy *Unique Distinct l-diversity* if for each of its equivalence class e , $e[S]$ contains exactly l distinct *SA* values.

Observation 1. If equivalence class e satisfies *Unique Distinct l-diversity*, then it also satisfies *Distinct l-diversity*, *Entropy l-diversity* and *Recursive (c,l)-diversity* for all constant $c > 1$.

The proof is simple, we need only to check the demand of the three models one by one. According to this observation, *Unique Distinct l-diversity* is a stronger model. $\square$

Observation 2. *Unique Distinct l-diversity prevents “probability inference attack”¹ better than other three models.*

This is also apparent, in *Unique Distinct l-diversity*, the *SA* attributes are uniformly distributed. Therefore, when the attacker locates some individual in a certain equivalence class e , without further background knowledge[5], he cannot disclose the individual’s *SA* value with probability higher than $1/l$. However, in the other three models,

¹ Or “skewness attack”[6], the privacy disclosure because of non-uniform distribution of *SA* values within a group.

there may be cases when one SA value appears many more times than other SA value in $e[S]$. Then the attacker could guess the individual has such SA value with high probability. $\square$

The foregoing observations substantiate the advantages of Unique Distinct l -diversity. We shall prove its feasibility in Section 4.

3 The Implementation of the *Selecting* Step

In *BSGI*, the tuples are first bucketized according to their SA values. Let B_i denote the i^{th} greatest bucket and $B = \{B_1, B_2, \dots, B_m\}$ denote the set of buckets. We have: $n_i = |B_i|$, $n_1 \geq n_2 \geq \dots \geq n_m$, $\sum_{i=1}^m n_i = n$. Since different n_i 's may vary greatly, we shall use the following “*Max-l*” method to ensure the formed “ l -tuple groups” are as many as possible: in each iteration of selecting, one tuple is removed from each of the l largest buckets to form a new group. Note that after one iteration, the size of some buckets will be changed. So in the beginning of every iteration, the buckets are sorted according to their sizes, as shown in Figure 2.

Theorem 1. *The Max-l method creates as many groups as possible.*

Proof. We prove by induction on $m = |B|$ and $n = |T|$.

Basis. $m = n = l$. This is the basis because when $m < l$ or $n < l$, no group can be created. In this case, there is exactly one tuple in each bucket, apparently, the *Max-l* method creates as many groups as possible.

Induction. When $m > l$, $n > l$. Assume the way W creates maximal number of groups, which equals k . We denote $G_i = \{i_1, i_2, \dots, i_l\}$ ($i_1 < i_2 < \dots < i_l$) to be the i^{th} group created by W and G_i contains one tuple from each of $B_{i_1}, B_{i_2}, \dots, B_{i_l}$. From W , a new way W' can be constructed that satisfies: (1) W' creates k groups; (2) The first group created by W' is $G'_1 = \{1, 2, \dots, l\}$. The construction takes two operations: *swap* and *alter*.

1. **swap.** $((i, a), (j, b))$ ($1 \leq i, j \leq k$, $1 \leq a, b \leq m$, $a \in G_i$, $a \notin G_j$, $b \in G_j$, $b \notin G_i$) means to exchange a in G_i with b in G_j . For example, $G_1 = \{1, 2\}$, $G_2 = \{3, 4\}$, *swap* $((1, 1), (2, 3))$ leads to $G_1 = \{2, 3\}$, $G_2 = \{1, 4\}$. Since $a \notin G_j$, $b \notin G_i$, the grouping way after this operation is always valid.
2. **alter.** (a, b) ($1 \leq a < b \leq m$) means to replace each a in every G_i with b and replace each b with a . For the above example, *alter* $(2, 3)$ leads to $G_1 = \{1, 3\}$, $G_2 = \{2, 4\}$. The grouping way is valid after this operation if and only if a 's total appearing times is no more than b 's.

The construction is like this: for variable i from 1 to l , assume the i^{th} element in G_1 is b . If $i = b$, we do nothing. Otherwise, b must be greater than i . We check for other $k - 1$ groups $G_2, \dots, G_k$. There are two possible cases:

1. There is a group G_j such that $i \in G_j$ and $b \notin G_j$. In this case, we perform *swap* $((1, b), (j, i))$ to obtain a new grouping way. Since $i \notin G_1$, $b \notin G_j$, it is still a valid grouping way.

2. Every group that contains i also contains b . Therefore, the total number of i 's is no more than that of b 's. In this case, we perform $alter(i, b)$, the grouping way is still valid after this operation.

Note operation on i ensures the i^{th} element in G_1 to be i and does not change the first $i - 1$ elements. So when the whole process finishes, we obtain a valid grouping way W' with $G'_1 = \{1, 2, \dots, l\}$. Removing tuples responding to the elements in G'_1 , we obtain a new instance of the problem with $m' \leq m, n' = n - l < n$. Due to induction hypopiesis, we know our algorithm generates as many groups as possible for the new instance. In the meantime, the best solution to the new instance contains at least $k - 1$ groups, because $G'_2, G'_3, \dots, G'_k$ is such a grouping way. So for the original instance, our algorithm generates at least k groups. That is the maximal number as assumed. The proof is completed. $\square$

During selecting, in order to reduce information loss and avoid exhaustively searching the solution space, the following greedy method is adopted: in each iteration of selecting, a random tuple t_1 is selected from B_1 and it forms the original equivalence class(group) e . For variable i from 2 to l , from B_i , a tuple t_i that minimize $IL(e \cup t_i)$ is selected and merged into e , as shown in Figure 2.

4 The Property of Residual Tuples after *Selecting and Grouping*

In this section, we shall investigate the property of residual tuples after selecting and grouping steps.

Theorem 2. *When the selecting and grouping steps terminate, there will be no residual tuples if and only if the buckets formed after the “bucketizing” step satisfy the following properties (we call it l -Property):*

- (1) $\frac{n_i}{n} \leq \frac{1}{l}, i = 1, 2, \dots, m$ (Use the same notation: n_i, m, n , as in Section 3)
- (2) $n = kl$ for some integer k

Proof. First notice that $\frac{n_i}{n} \leq \frac{1}{l}$ is equivalent with $n_1 \leq k$, because n_1 is the largest among all n_i 's.

(If) We prove by induction on $m = |B|$ and $n = |T|$.

Basis. $m = n = l$, this is the basis because m cannot be smaller than l . Now there's one tuple in each bucket. Obviously the algorithm leaves none.

Induction. $m > l$ or $n > l$. Resembling the proof of Theorem 1, we assume that when the first group is created by our algorithm, the remaining buckets and tuples form a new instance of the problem with parameter (m', n') . We shall prove this new instance also has l -Property.

Apparently $m' \leq m, n' = n - l = (k - 1)l$. To prove $\frac{n'}{n'} \leq \frac{1}{l}$. We discuss two cases for different values of n_1 .

1. $n_1 = k$. Assume that $n_1 = n_2 = \dots = n_j = k, n_{j+1} < k$. We have:

$$n = kl = \sum_{i=1}^m n_i = \sum_{i=1}^j n_i + \sum_{i=j+1}^m n_i \geq kj$$

So $l \geq j$. This means the number of the buckets with k tuples does not exceed l . According to our algorithm, after the first group is removed, the bucket with most tuples has size $k - 1$ because all the buckets previously has size k contribute one to that group. That is $n'_1 = k - 1 = \frac{n'}{l}$, or $\frac{n'_1}{n'} \leq \frac{1}{l}$.

2. $n_1 \leq k - 1$. This case is simple because $n'_1 \leq n_1 \leq k - 1$, so $\frac{n'_1}{n'} \leq \frac{1}{l}$.

In both cases, we obtain that the new instance has l -Property. With the very same idea as used in the proof of Theorem 1, the outcome of the remaining execution of the algorithm equals to what we obtain by running the algorithm individually for the new instance. Due to induction hypopiesis, we know our algorithm will leave no non-empty buckets. So for the original instance, the conclusion also holds. The proof of if-part is completed.

(Only-if) It is easy to verify that n must be multiple of l to guarantee that all the tuples can be grouped. So there exists some integer k such that $n = kl$

Since there's no residual tuples, for the requirement of l -diversity, each group contains at most one tuple from the first bucket. The mapping from the tuples in B_1 to the groups is *one - to - one*, but not necessarily *onto*. Therefore, we have $n_1 \leq k = \frac{n}{l}$, or $\frac{n_1}{n} \leq \frac{1}{l}$. The proof of only-if part is completed. $\square$

When the buckets satisfy the first condition while do not satisfy the second condition of l -Property, we have following conclusion:

Corollary 1. *If the buckets satisfy following Property: $\frac{n_i}{n} \leq \frac{1}{l}$, then after the selecting and grouping steps, each non-empty bucket has only one tuple.*

Proof. Assume $n = kl + r$, $0 \leq r < l$, hypothetically change our algorithm like this: first subtract one tuple from each of $B_1, B_2, \dots, B_r$, then operate the “ $Max - l$ ” selecting method in Section 3. The new instance satisfies l -Property and k groups will be formed. Therefore the best solution creates no less than k groups. In the meantime it creates no more than k groups because $n = kl + r$.

Now we already know there are k iterations of “selecting and grouping” in total², denote them to be $I_1, I_2 \dots I_k$. Assume one bucket(denoted B_{bad}) contains at least 2 tuples after I_k . Note before I_k , there are at most $l - 1$ buckets with size at least 2, otherwise there will be at least l non-empty buckets after I_k . So a tuple from B_{bad} is selected during I_k and $|B_{bad}| \geq 3$ before I_k . Similarly, before I_{k-1} , there are at most $l - 1$ buckets with size at most 3. So a tuple from B_{bad} is selected during I_{k-1} and $|B_{bad}| \geq 4$ before I_{k-1} . Recursively, we obtain $|B_{bad}| \geq k + 2$ before I_1 , this contradicts the condition. The proof is completed. $\square$

The above result is of great merits. On one side, the number of residual tuples is limited and bounded by l , our algorithm will not suffer from large number of residual tuples. Thus the feasibility of Unique Distinct l -diversity can be assured. As proved in Section 2, Unique Distinct l -diversity is a stronger l -diversity model which provides better privacy preservation. The experiment in Section 6 will also substantiate this. In sum, we have:

² Similar theorem is proved in [8], however, we find that proof ungrounded because it assumes the number of iteration equals k , without proof.

Corollary 2. *Unique Distinct l -diversity can be exactly achieved if the original table satisfy both l -Property (1) and (2). If the table just satisfy l -Property (1), Unique Distinct l -diversity can be achieved with less than l residual tuples.*

On the other side, we can choose a proper l according to the distribution of SA values. Consider, assigning a large number to l provides better privacy preservation but greater information loss, while a small number leads to less data distortion but higher privacy disclosure risk. Current studies ignore to investigate the optimal assignment of l to balance such trade-off. However, from previous discussion we can reach the following conclusion:

Corollary 3. *The optimal assignment to parameter l in l -diversity is $\max\{2, \lfloor \frac{n}{n_1} \rfloor\}$.*

If $\lfloor \frac{n}{n_1} \rfloor = 1$, this reflects the most frequent SA value takes a proportion more than 50%. This is a greatly “skew” distribution and the privacy disclosure risk cannot be reduced to below $1/2$.

As for the residual tuples, the simplest way is to *suppress* them. Here we perform *incorporating*: for each of them, find a proper equivalence class to incorporate it. The so called “proper” has two requirements: (1)The chosen equivalence class had better not contain the new SA value, thus it will satisfy Unique Distinct $(l + 1)$ -diversity after incorporation. (2)The incorporation leads to minimal information loss. The detailed implementation is in Figure 3.

5 The BSGI Algorithm

5.1 The Algorithm

Summing up the previous discussions, we formally present the *BSGI* algorithm in this section.

The “*Select*” procedure in Figure 2 implements the “*Max- l* ” selecting method in Section 3 and the “*Incorporate*” procedure implements the incorporating method in Section 4. Say, if there exists some equivalence class e that $t[S] \notin e[S]$, t is incorporated into one of such classes that minimize the information loss. Otherwise, for each e , $t[S] \in e[S]$, the choosing of e to incorporate t is only based on minimal information loss.

5.2 Further Discussion about the Algorithm

In this section, we shall discuss some special cases with regard to *BSGI*.

1. The *single-Individual-Multi-Class* Case

Note our algorithm can be categorized into “local-recoding”[13] that the created equivalence classes may overlap each other. Thus one individual may be associated with more than one equivalence classes. For instance, in Table 4, the individual *George* can be associated with both Group 2 and Group 3. With regard to its influence on privacy disclosure risk, we shall prove:

Input: Original table T
Output: Anonymized table T^* which satisfies l -diversity
Data: $E = \emptyset$, E is the set of equivalence classes

```

1 begin
  /* The bucketizing step */
2   Bucketize tuples of  $T$  according to their  $SA$  values;
3    $B = \{B_i\}$                                      /*  $B$  is the set of buckets */
  /* The selecting and grouping steps */
4   while  $|B| \geq l$  do
5      $E = E \cup \text{Select}()$ ;
  /* The incorporating step */
6   foreach residual tuple  $t$  do
7      $\text{Incorporate}(t)$ ;
8   return  $T^*$ ;
9 end

```

Fig. 1. The BSGI Algorithm

Data: B = the set of buckets; $e = \emptyset$, the equivalence class to be created

```

1 begin
2   Sort buckets in  $B$  according to their size;
3    $B = \{B_1, B_2, \dots, B_m\}$  where  $B_i$  is the  $i^{th}$  greatest bucket in  $B$ ;
4   Randomly remove one tuple  $t_1$  from  $B_1$ ;
5    $e = \{t_1\}$ ;
6   for  $i \leftarrow 2$  to  $l$  do
7     Remove one tuple  $t_i$  from  $B_i$  that minimize  $IL(e \cup t_i)$ ;
8      $e = e \cup t_i$ ;
9   return  $e$ ;
10 end

```

Fig. 2. The Select Procedure

Data: E = the set of equivalence classes; t = the tuple to be incorporated

```

1 begin
2    $E' = \{e | e \in E \text{ and } t[S] \notin e[S]\}$ ;
3   if  $|E'| \neq 0$  then
4     Find  $e$  in  $E'$  that minimize  $IL(e \cup t)$ ;
5   else
6     Find  $e$  in  $E$  that minimize  $IL(e \cup t)$ ;
7    $e = e \cup t$ ;
8 end

```

Fig. 3. The Incorporate Procedure

Theorem 3. *The case of single-individual-multi-class does not increase sensitive attribute disclosure risk to more than $1/l$.*

Proof. Assume one individual I , with SA value $I[s]$, can be associated with equivalence classes $e_{i_1}, e_{i_2}, \dots, e_{i_j}$. According to probability's Bayes Model, the risk of sensitive attribute disclosure is

$$\sum_{k=1}^j Pr(I \in e_{i_k}) \cdot Pr(\text{privacy disclosure} | I \in e_{i_k})$$

Consider

$$\forall k, Pr(\text{privacy disclosure} | I \in e_{i_k}) \leq 1/l$$

and

$$\sum_{k=1}^j Pr(I \in e_{i_k}) = 1$$

We have, the total risk of sensitive attribute disclosure:

$$Pr(\text{privacy disclosure}) \leq 1/l$$

□

2. The Single-Individual-Multi-Tuple Case

Traditionally, we assume one single individual corresponds to a single tuple in the table. However, there are cases where one single individual corresponds to multiple tuples. (e.g., one person's multiple disease records for different diseases). In this case, if multiple tuples of a same individual is grouped together, the proportion of tuples containing the individual's SA values within that group will be larger than $1/l$, thus leads to higher privacy disclosure risk.

To counter such case, we need only to add a “*check*” procedure during the selecting step. If a candidate tuple belongs to a already-selected individual, that tuple will not be selected.

3. The Single-Tuple-Multi-SA Case

Traditionally, we deal with the case where a single tuple contains only one sensitive attribute. For the *single-tuple-multi-SA* case, an intuitive thinking is to consider the SA value as one multi-dimensional vector. However, this may lead to privacy disclosure. Consider the case of two sensitive attributes: (*Disease*, *Salary*). The values (*flu*, \$10000), (*cancer*, \$10000), (*obesity*, \$10000) do not equal to each other. But if tuples with these SA values are grouped, the disclosure risk for attribute *Salary* is 100%.

To counter such case, in the *selecting* step, the new tuple should be unequal to each of the already-selected tuples on all sensitive attributes. However, this is quite a preliminary approach, it's performance deserves extensive study.

6 Experiments

In this section, we conducted several experiments using the real world database *Adult*, from the *UCI* Machine Learning Repository[20] to verify the performance of *BSGI* in

both efficiency and data quality by comparing with full-domain generalization algorithm “*Incognito*” and multi-dimensional partition algorithm “*Mondrian*” respectively.

6.1 Experimental Data and Setup

Adults database is comprised of data from the US Census. There are 6 numerical attributes and 8 categorical attributes in Adult. It leaves 30162 records after removing the records with missing value. In our experiments, we retain only eight attributes. {*Age*, *Final-Weight*, *Education*, *Hours per Week*, *Marital Status*, *Race*, *Gender*} are considered as Quasi-identifying attributes. The former four attributes are treated as numeric attributes while the latter three are treated as categorical attributes. *WorkClass* is the sensitive attribute. According to Corollary 1, the upper bound of l is determined to be 7 because the most frequent *SA* value “*Prof-specialty*” takes a proportion greater than $1/8$ while less than $1/7$.

We modify LeFevre’s *Incognito*[13] and *Mondrian*[14] into the l -diversity versions. These two algorithms and our *BSGI* are all built in Eclipse 3.1.2, JDK 5.0, and executed on a dual-processor Intel Pentium D 2.8 GHz machine with 1 GB physical memory. The operating system is Microsoft Windows Server 2003.

6.2 Efficiency

The running time of *Incognito* is not in Figure 4 because such exhaustive algorithm takes nearly exponential time in the worst case. In our experiment, its execution time is more than half an hour, exceed the other two by several orders of magnitude. We execute both *BSGI* and *Mondrian* three times, and calculate the average. Figure 4 reports the average time of both algorithms. As is shown, the running time of *Mondrian* decreases from about 90s to 75s, because when l increases, the recursive depth of the algorithm reduces. However, as is shown, *BSGI* performs much better than *Mondrian* and almost does not increase with l . In fact, it is easy to conclude the time complexity of *BSGI* is $O(n^2)$, highly efficient and independent of l .

6.3 Data Quality

Figure 6 depicts the widely adopted metric: Discernibility Metric cost(DM)[16] of the three algorithms and Table 5 shows the average group size resulted from them. These two metrics are mutually related, because without suppression, DM is defined as

$$DM = \sum_{\text{each equivalence class } e} (|e|)^2$$

In both metrics, the cost of *Incognito* exceeds the other two by orders of magnitude. Since *Incognito* always maps all the QI values within the same level of its generalization hierarchy into a same generalized value, as a result, it tends to over-generalize the original table. In fact, over-generalization is the fatal shortcoming of the class of full-domain generalization algorithms. Secondly, *BSGI* does a much better job than *Mondrian*. Actually, *BSGI* always achieves the best result with regard to these two

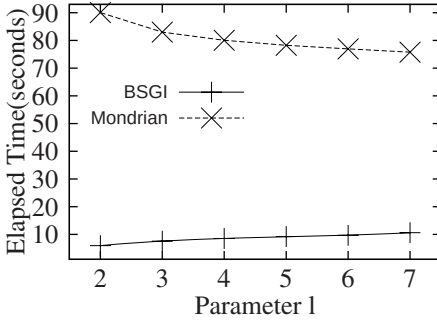


Fig. 4. Elapsed Time

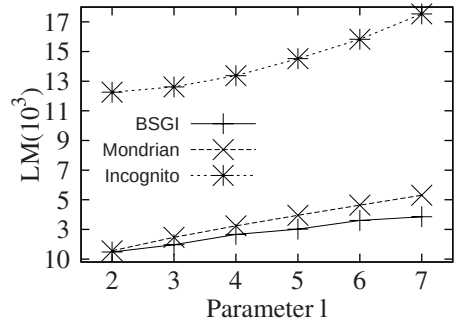


Fig. 5. Information Loss Metric

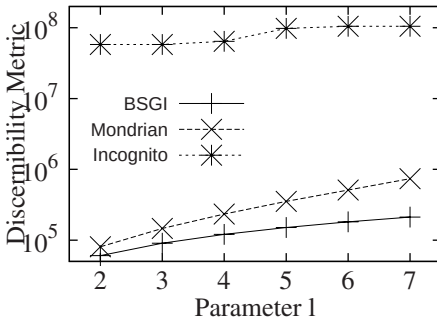


Fig. 6. Discernibility Metric

Table 5. Average Group Size

l	Average Group Size		
	BSGI	Mondrian	Incognito
2	2.00	2.47	471
3	3.00	4.32	471
4	4.00	6.71	628
5	5.00	9.81	942
6	6.00	13.79	1005
7	7.00	18.73	1005

metrics, because it implements the Unique Distinct l -diversity model and every equivalence class is of the minimal size l . We can learn that there are almost exactly l tuples in each equivalent class generated by *BSGI*.

Besides *DM* and average group size metrics, we adopt the *IL* metric in Section 2.2, which gives more information about how much the tuples are generalized. Figure 5 demonstrates the *IL* as a function of l . Again, *Incognito* causes more loss by orders of magnitude. *BSGI* is the best but the advantage seems not so significant in comparison with *Mondrian*. When $l = 7$, the *IL* of *BSGI* is 70% of *Mondrian*'s. This can be explained by the implementation of *selecting* step: the new selected tuple that minimize *IL* is not from the whole table, but from an appointed bucket. As proved in Section 3, such selecting method ensures the maximum number of created groups, however may be unable to achieve minimal information loss. This cost is worthwhile, because Unique Distinct l -diversity largely enhances privacy preservation.

In sum, the excessively long execution time and high information loss render *Incognito* almost impractical. *BSGI* achieves the optimal *DM* and *AverageSize* metric. With regard to the *IL* metric, *BSGI* still outperforms *Mondrian* apparently. The *BSGI* is an highly efficient algorithm with low information loss. In the meantime, it achieves the stronger Unique Distinct l -diversity model, which preserves privacy excellently.

7 Related Work

As introduced in the abstract, the work dealing with developing privacy models includes [5,6,7,8,9,10,11] and etc. [6] proposes the model of t -closeness, which requires the distribution of SA values in each equivalence class to be close to the entire table. [7] enable personal specified degree of privacy preserving. Instead of generalizing original QI values, [8] anatomize the original table into a quasi-identifier table (QIT) and a sensitive table (ST). [9] propose the model of δ -presence to the case of individual presence should be hidden. Unlike previous work on static datasets, [10,11] deal with privacy preserving for dynamic, or incremental datasets. The work on designing algorithms for privacy models includes [13,14,15,16,17] and etc. [13], [14] and [15] represent three main classes of algorithms: *hierarchy-based*, *partition-based* and *clustering-based*. In fact, our work can be categorized into *clustering-based* algorithms. There are still other related works. The information loss metric proposed by [12] is adopted by this paper. [19] investigates the large information loss that privacy preservation techniques encounter in high-dimension cases.

8 Conclusion and Future Work

In this paper, we propose a specially designed algorithm: *BSGI* for l -diversity. Through such algorithm, a stronger l -diversity model, Unique Distinct l -diversity can be achieved with less information loss. We also investigate the optimal assignment to parameter l in the model.

For the future work, although we have dealt with the *single-tuple-multi-SA* case, further analysis on the influence of multiple sensitive attributes and designing specific algorithm are of great merits. In the meantime, it may be worthwhile to extend *BSGI* to work on dynamic growing datasets.

Acknowledgments. This work was Supported in part by the National Natural Science Foundation of China Grant 60553001, 60573094, the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, the National High Technology Development 863 Program of China under Grant No.2007AA01Z152 and 2006AA01A101, the National Grand Fundamental Research 973 Program of China under Grant No. 2006CB303103, and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

References

1. Samarati, P.: Protecting respondents identities in microdata release. *TKDE* 13(6), 1010–1027 (2001)
2. Sweeney, L.: Achieving k -anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 10(5), 571–588 (2002)
3. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information. In: *PODS*, p. 188 (1998)

4. Sweeney, L.: k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems* 10(5), 557–570 (2002)
5. Machanavajjhala, A., Gehrke, J., Kifer, D.: l-diversity: Privacy beyond k-anonymity. In: *ICDE*, p. 24 (2006)
6. Li, N., Li, T.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: *ICDE*, pp. 106–115 (2007)
7. Xiao, X., Tao, Y.: Personalized privacy preservation. In: *SIGMOD*, pp. 229–240 (2006)
8. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: *VLDB*, pp. 139–150 (2006)
9. Ercan Nergiz, M., Atzori, M., Clifton, C.W.: Hiding the Presence of Individuals from Shared Databases. In: *SIGMOD*, pp. 665–676 (2007)
10. Xiao, X., Tao, Y.: m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets. In: *SIGMOD*, pp. 689–700 (2007)
11. Byun, J.-W., Li, T., Bertino, E., Li, N., Sohn, Y.: Privacy-Preserving Incremental Data Dissemination. *CERIAS Tech Report*, Purdue University (2007-07)
12. Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, A.: Utility-Based Anonymization Using Local Recoding. In: *SIGKDD*, pp. 785–790 (2006)
13. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: *SIGMOD*, pp. 49–60 (2005)
14. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: *ICDE*, p. 25 (2006)
15. Byun, J.-W., Kamra, A., Bertino, E., Li, N.: Efficient k-Anonymization Using Clustering Techniques. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) *DASFAA 2006*. LNCS, vol. 3882. Springer, Heidelberg (2006)
16. Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymization. In: *ICDE*, pp. 217–228 (2005)
17. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k-anonymity. In: *JOPT* (2005)
18. Iyengar, V.: Transforming data to satisfy privacy constraints. In: *SIGKDD*, pp. 279–288 (2002)
19. Aggarwal, C.C.: On k-anonymity and the curse of dimensionality. In: *VLDB*, pp. 901–909 (2005)
20. U.C. Irvin Machine Learning Repository,
<http://archive.ics.uci.edu/ml/>

The Truncated Tornado in TMBB: A Spatiotemporal Uncertainty Model for Moving Objects

Shayma Alkobaisi¹, Petr Vojtěchovský², Wan D. Bae³,
Seon Ho Kim⁴, and Scott T. Leutenegger⁴

¹ College of Information Technology, UAE University, UAE
`shayma.alkobaisi@uaeu.ac.ae`

² Department of Mathematics, University of Denver, USA
`petr@math.du.edu`

³ Department of Mathematics, Statistics and Computer Science,
University of Wisconsin-Stout, USA
`baew@uwstout.edu`

⁴ Department of Computer Science, University of Denver, USA
`{seonkim,leut}@cs.du.edu`

Abstract. The uncertainty management problem is one of the key issues associated with moving objects (*MOs*). Minimizing the uncertainty region size can increase both query accuracy and system performance. In this paper, we propose an uncertainty model called the *Truncated Tornado* model as a significant advance in minimizing uncertainty region sizes. The *Truncated Tornado* model removes uncertainty region sub-areas that are unreachable due to the maximum velocity and acceleration of the *MOs*. To make indexing of the uncertainty regions more tractable we utilize an approximation technique called *Tilted Minimum Bounding Box (TMBB)* approximation. Through experimental evaluations we show that *Truncated Tornado* in *TMBB* results in orders of magnitude reduction in volume compared to a recently proposed model called the *Tornado* model and to the standard “Cone” model when approximated by axis-parallel *MBB*.

1 Introduction

In recent years, there is an increasing number of location-aware spatiotemporal applications that manage continuously changing data. Tracking systems, mobile services and sensor-based systems now track millions of GPS and RFIDs that can report the positions of the moving objects. These applications require new strategies for modeling, updating and querying spatiotemporal databases.

To be able to answer location-based queries, it is necessary to maintain the locations of a large number of moving objects over time. It is infeasible to store the object’s exact continuously changing location since this would require more updates than can be managed by the *MO* database. This is a first cause of *MO* location inaccuracy. A second cause is that devices are limited in ability to

report accurate locations. As a result of these inaccuracies, *MO* spatiotemporal data often require uncertainty management algorithms.

A common model of spatiotemporal query processing is to divide the query into a filtering step and a refinement step [3]. The performance of the filtering step is improved when there is a lower rate of false-hits, i.e., objects that are returned in the filtering step but subsequently are discarded by the refinement step. In spatiotemporal queries, minimizing the size of uncertainty regions and any used region approximations will result in improving the filtering step efficiency. Minimizing these regions and their approximations is the main goal of this work.

2 Related Work

Many uncertainty models for moving objects have been proposed based on the underlying applications. Uncertainty regions of moving objects in [10] are presented in 3D as cylindrical bodies which represent all the possible positions between two reported past locations. The authors in [7] proposed one of the most common uncertainty models showing that when the maximum velocity of an object is known, the uncertainty region between any two reported locations can be represented as an error ellipse. Another popular model is found in [5]. It represents the uncertainty region as an intersection of two half cones. Each cone constrains the maximum deviation from two known locations in one movement direction. Recently in [12], a non-linear extension of the funnel model [11], named *Tornado* was presented. This higher degree model reduces the size of the uncertainty region by taking into account higher order derivatives, such as velocity and acceleration.

There is a lack of research in investigating the effect of different object approximations on the false-hit rate. In [3], the authors investigated six different types of static spatial objects approximations. Their results indicated that depending on the complexity of the objects and the type of queries, the approximations five-corner, ellipse and rotated bounding box outperform the axis-parallel bounding box. The authors in [1] presented *MBR* approximations for three uncertainty region models, namely, the *Cylinder Model*, the *Funnel Model of Degree 1* which is the *Cone* model proposed in [5] and the *Funnel Model of Degree 2* which is the *Tornado* model presented in [12].

Research in the field of computational geometry has resulted in several object approximation solutions. In [9], the author was able to use the fact that a minimal area rectangle circumscribing a convex polygon has at least one side flush with an edge of the polygon to use the “rotating calipers” algorithm to find all minimal rectangles in linear time. In [6], O’Rourke presented the only algorithm for computing the exact arbitrarily-oriented minimum volume bounding box of a set of points in R^3 which runs in $\mathcal{O}(n^3)$. The authors in [2], proposed an efficient solution of calculating a $(1 + \epsilon)$ -approximation of the non axis-parallel minimum-volume bounding box of n points in R^3 . The running time of their algorithm is $\mathcal{O}(n \log n + n/\epsilon^3)$.

3 The Truncated Tornado

Assuming maximum velocity M_v and maximum acceleration M_a of the moving object, the *Tornado* model [12] calculated the uncertainty region defining functions as shown in Fig. 1 (a). The dotted region is the uncertainty region defined by the *Tornado* model and the top and bottom intervals represent the instrument and measurement error associated with each reported position.

Let $displ_1$ and $displ_2$ be, respectively, a first-degree and second-degree displacement functions defined as follows:

$displ_1(V, t) = V \cdot t$ and $displ_2(V, a, t) = \int_0^t (V + a \cdot x) dx \approx V \cdot t + (a/2) \cdot t^2$, where V is the current velocity of the moving object, a is acceleration and t is time. The future (past) position f_{pos} (p_{pos}) of a moving object after (before) some time, t , can be calculated as follows:

$$f\_pos(P_1, V_1, M_v, M_a, t) = \begin{cases} P_1 + D_2 + D_1 & \text{if } t_{Mv} < t \\ P_1 + D_2 & \text{otherwise} \end{cases}$$

$$p\_pos(P_2, V_2, M_v, M_a, t) = \begin{cases} P_2 - D_2 - D_1 & \text{if } t_{Mv} < t \\ P_2 - D_2 & \text{otherwise} \end{cases}$$

$D_1 = displ_1(M_v, t - t_{Mv})$ and $D_2 = displ_2(V, M_a, t_{Mv})$, where t_{Mv} is the time the moving object needs to reach M_v . Notice that the above functions define the dotted uncertainty region in Fig. 1 (a). $g_{min}(t)$ and $g_{max}(t)$ are produced by the f_{pos} function, and $f_{min}(t)$ and $f_{max}(t)$ are produced by the p_{pos} function.

Objects moving with momentum cannot make extreme changes in their velocity. Hence they need some time to change their velocities from one direction to the opposite direction, thus, the right and left corners P_{max} and P_{min} shown in Fig. 1 (a) are impossible to be reached by the moving object unless we assume

Table 1. Notations used in this paper

Notation	Meaning
P_1	a reported position of a moving object
P_2	a following reported position
t_1	time instance when P_1 was reported
t_2	time instance when P_2 was reported
t	any time instance between t_1 and t_2 inclusively
T	time interval between t_2 and t_1 , $T = t_2 - t_1$
V_1	velocity vector at P_1
V_2	velocity vector at P_2
e	instrument and measurement error
M_v	maximum velocity of an object
M_a	maximum acceleration of an object
t_{Mv}	time to reach maximum velocity

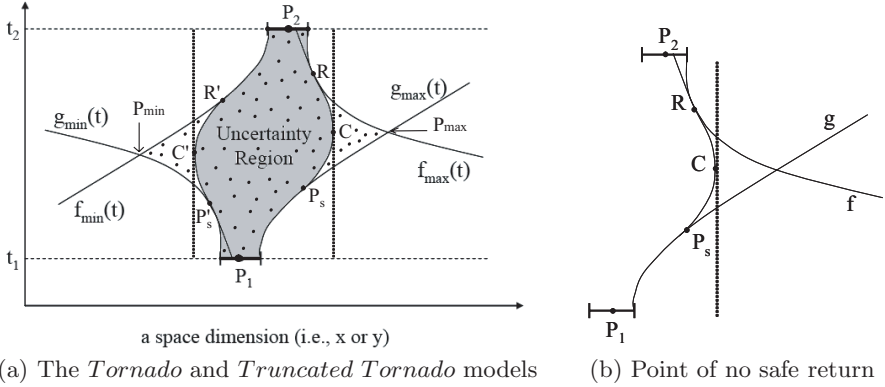


Fig. 1. Calculating uncertainty regions of *Truncated Tornado*

infinite acceleration. Our proposed *Truncated Tornado* model removes unreachable sub-areas of the uncertainty regions by calculating the furthest point an object can reach given its maximum acceleration.

Assuming that the two intervals and trajectories are as shown in Fig. 1 (b), we define the *Truncated Tornado* model as follows:

We say that P_s is the *point of no safe return* for g if P_s is the rightmost point on the trajectory g such that when the object (car) starts changing direction at P_s then it will touch the trajectory f (at point R), i.e., the object is within the boundary of the maximum possible deviation.

Since any realizable trajectory between the two intervals must remain within the boundary defined by f and g , it is clear that the point C (which is the rightmost point on the decelerating trajectory started at P_s) can be used as a cut point for the right boundary of the *MBR* encompassing the uncertainty region. This boundary is indicated in the figure by the dotted line.

The question is how to calculate the points P_s and C . We show the case when both f, g are parabolas. Upon turning the situation by 90 degrees counterclockwise, f, g are parabolas given by $f(x) = ax^2 + b_1x + c_1$, $g(x) = ax^2 + b_2x + c_2$. (We use the same quadratic coefficient a since the maximal acceleration M_a is the same for f and g .) Note that $b_1 \neq b_2$ since the parabolas f, g are not nested.

Let x_0 be the x -coordinate of the point P_s . The deceleration trajectory started at P_s is a parabola, and it can be given by $h(x) = -ax^2 + ux + v$. To determine u, v and x_0 , we want h to stay below f at all times and hence $-ax^2 + ux + v \leq ax^2 + b_1x + c_1$ for every x . Equivalently, $k(x) = 2ax^2 + (b_1 - u)x + (c_1 - v) \geq 0$ for every x . Since h needs to touch f , we want k to be a parabola that touches the x -axis. Equivalently, the discriminant $(b_1 - u)^2 - 4(2a)(c_1 - v)$ needs to be equal to 0. This yields

$$v = c_1 - (b_1 - u)^2 / (8a) \quad (1)$$

Analogously, we want h to stay below g at all times and hence $-ax^2 + ux + v \leq ax^2 + b_2x + c_2$ for every x . Equivalently, $k(x) = 2ax^2 + (b_2 - u)x + (c_2 - v) \geq 0$ for every x . Since h needs to touch g , we want k to be a parabola that touches

the x -axis. Equivalently, the discriminant $(b_2 - u)^2 - 4(2a)(c_2 - v)$ needs to be equal to 0. This yields

$$v = c_2 - (b_2 - u)^2 / (8a) \quad (2)$$

The parabola h must satisfy both (1) and (2), therefore, we can set (1) = (2), eliminate v from the equation, solve for u and then substitute to find v . Solving for u with the observation that $b_1 \neq b_2$ we get

$$u = 4a \frac{c_2 - c_1}{b_1 - b_2} + \frac{1}{2}(b_1 + b_2) \quad (3)$$

It is now easy to find the cut point C , as this is the vertex of the parabola h . Notice that we only need to calculate u to find C since $C = \frac{u}{2a}$.

Finally, the reverse time problem (going from P_2 to P_1) is precisely the forward time problem: we are looking for a parabola that stays below and touches both f and g , hence the reverse time parabola coincides with the forward time parabola.

The same technique needs to be applied to find the cut point C' on the left boundary of the calculated MBR , i.e., the minimum extreme point of the uncertainty region (see Fig. 1 (a)). In this case, upon turning the situation by 90 degrees counterclockwise, we see that f, g are parabolas given by $f(x) = -ax^2 + b_1x + c_1$, $g(x) = -ax^2 + b_2x + c_2$ and h is a parabola given by $h(x) = ax^2 + ux + v$ and we need h to stay above f and g at all times and touches them.

The uncertainty region example shown in Fig. 1 (a) is generated by this model when both P_s and R for the minimum and maximum calculations lie on curved part of g and f , respectively. Obviously, there are three other cases that need to be considered when calculating C and C' , depending on the locations of P_s, R and P'_s, R' , respectively. We leave these cases to the reader.

4 The TMBB Approximation

The uncertainty regions of MOs are rather “tilted” in shape in which traditional (axis-parallel) Minimum Bounding Boxes $MBBs$ are most likely not close to the optimal approximations of the regions. The advantage of *Truncated Tornado* can be strengthened by a more accurate approximation that takes the tilted shape of the regions into account and not only the extreme points of the uncertainty region. We investigate *Tilted Minimum Bounding Boxes (TMBBs)* as approximations of the uncertainty regions generated by *Truncated Tornado*. When compared with axis-parallel $MBBs$ in 3D, $TMBBs$, which are minimum volume bounding boxes that relax the axis-aligned property of $MBBs$, generally allow geometries to be bounded more tightly with a fewer number of boxes [4].

To calculate $TMBB$ of the uncertainty region of Fig. 1 (a), we identify all the extreme points that need to be considered as follows:

R, C and P_s of the maximum direction (upper boundary) in the x -dimension need to be calculated and the corresponding y -values (at specific time instance when the x -values are calculated) are assigned to these points. Similarly, R, C and P_s in the y -dimension need to be calculated and the corresponding x -values

(at specific time instance when the y -values are calculated) are assigned to these points. This results in 6 points calculated in 3D. The same calculation set needs to be done for the minimum direction (lower boundary) by calculating R' , C' and P'_s in both the x and y dimensions which results in 6 other points. The other extreme points are $P_1 - e$, $P_1 + e$, $P_2 - e$ and $P_2 + e$.

Given 6 points in 3D calculated for the upper boundary, 6 points in 3D calculated for the lower boundary and finally 4 points in 3D (2 top and 2 bottom), we calculate $TMBB$ enclosing the uncertainty region of *Truncated Tornado* using the approximation method of [2].

5 Experiments

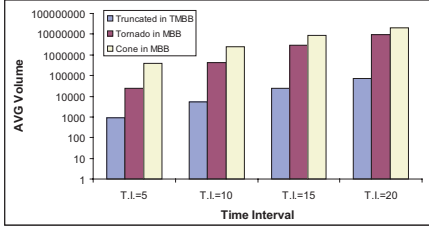
All velocities in this section are in *meters/second* (m/s), all accelerations are in *meters/second²* (m/s^2) and all volumes are in *meters² · second* (volumes in 3D are generated by moving objects in 2D with time being the 3rd dimension). Our synthetic datasets were generated using the “Generate_Spatio_Temporal_Data” (*GSTD*) algorithm [8].

Our synthetic dataset was generated by 200 objects moving with the velocity in the x direction greater than the velocity in the y direction with an average velocity of 17.76 m/s. Each object in the synthetic dataset reported its position and velocity every second for an hour. The real data set was collected using a GPS device while driving a car in the city of San Diego in California, U.S.A. The actual positions and velocities were reported every one second and the average velocity was 11.44 m/s. Although each moving object in both datasets reported its position every second, we used different time interval (T.I) values (e.g., T.I=10) to simulate various update frequencies. The range queries’ sizes in our experiments are calculated using certain percentages of the universe area combined with specific time extents. Table 2 shows the datasets and system parameters used in our experiments.

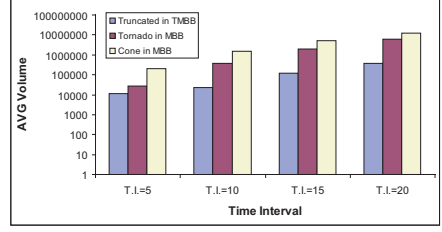
We first compared the volume of $TMBBs$ approximating the *Truncated Tornado* uncertainty regions to the volume of the axis-parallel $MBBs$ of *Tornado* and *Cone* using the real and synthetic datasets. Fig. 2 (a) and (b) show the average volume of $TMBBs$ generated by *Truncated Tornado* and the average volume of $MBBs$ generated by the other two models using the synthetic and real datasets, respectively. *Truncated Tornado* combined with $TMBB$ resulted in an average reduction of 93% and 97% over the axis-parallel MBB of

Table 2. Synthetic and real datasets and system parameters

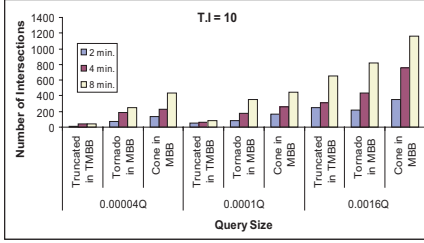
datasets		reported records			parameters	
		AVG Vel.	MAX Vel.	MAX Acc.	M_v	M_a
synthetic	Dataset	17.76	20.61	6.41	55	8
real	San Diego	11.44	36.25	6.09	38.89	6.5



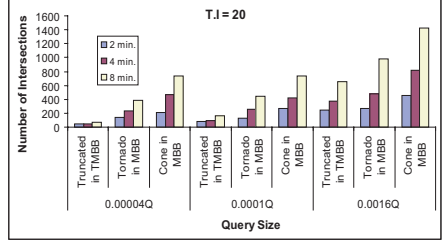
(a) Synthetic dataset



(b) Real dataset



(c) T.I = 10 (Real dataset)



(d) T.I = 20 (Real dataset)

Fig. 2. *Truncated in TMBB Vs. Tornado and Cone in MBB*

Tornado and *Cone*, respectively, using the real dataset. The reduction when using the synthetic dataset was 99% over both *Tornado* and *Cone*.

Next, we generated and evaluated 5000 random queries to *TMBBs* and *MBBs* calculated in the previous result for the real dataset. Fig. 2 (c) and (d) show the number of intersecting *TMBBs* of *Truncated Tornado* and the number of intersecting *MBBs* of *Tornado* and *Cone*. *TMBBs* of *Truncated* resulted in much less number of intersections compared to *MBBs* of the other models since *Truncated Tornado* results in much smaller uncertainty regions compared to *Tornado* and *Cone*. Also, *TMBBs* result in significantly smaller average volumes compared to *MBBs* as they more accurately approximate the uncertainty regions. The reduction in the number of intersections of *Truncated Tornado TMBBs* was 42% over *Tornado MBBs* and 62% over *Cone MBBs* when T.I=10. When T.I=20, the reduction over *Tornado MBBs* was 47% and was 68% over *Cone MBBs*.

6 Conclusions

In this paper we proposed the *Truncated Tornado* model that minimizes the moving object uncertainty regions. The model takes advantage of the fact that changes in the velocities of moving objects that move with momentum are limited by maximum acceleration values. This fact is used to identify and eliminate unreachable object locations, thus significantly reducing uncertainty region size. We then showed how to combine this model with the *Tilted Minimum Bounding*

Box (*TMBB*), in order to achieve another order of magnitude reduction in uncertainty region size when compared to approximation bounding via traditional *MBBs*. Experiments on both synthetic and real datasets showed an order of magnitude improvement over previously proposed uncertainty models in terms of I/O accesses.

References

1. Alkobaisi, S., Bae, W.D., Kim, S.H., Yu, B.: MBR models for uncertainty regions of moving objects. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DASFAA 2008. LNCS, vol. 4947, pp. 126–140. Springer, Heidelberg (2008)
2. Barequet, G., Har-Peled, S.: Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms* 38(1), 91–109 (2001)
3. Brinkoff, T., Kriegel, H.-P., Schneider, R.: Comparison of approximations of complex objects used for approximation-based query processing in spatial database systems. In: *Proceedings of Int. Conf. on Data Engineering*, pp. 40–49 (1993)
4. Gottschalk, S., Lin, M.C., Manocha, D.: OBB-tree: A hierarchical structure for rapid interference detection. In: *Proceedings of ACM Siggraph*, pp. 171–180 (1996)
5. Hornsby, K., Egenhofer, M.J.: Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence* 36(1-2), 177–194 (2002)
6. O'Rourke, J.: Finding minimal enclosing boxes. *International Journal of Parallel Programming* 14(3), 183–199 (1985)
7. Pfoser, D., Jensen, C.S.: Capturing the uncertainty of moving-objects representations. In: *Proceedings of Int. Symposium on Advances in Spatial Databases*, pp. 111–132 (1999)
8. Theodoridis, Y., Silva, J.R.O., Nascimento, M.A.: On the generation of spatiotemporal datasets. In: *Proceedings of Int. Symposium on Advances in Spatial Databases*, pp. 147–164 (1999)
9. Toussaint, G.T.: Solving geometric problems with the rotating calipers. In: *Proceedings of IEEE MELECON*, pp. A10.02/1–4(1983)
10. Trajcevski, G., Wolfson, O., Hinrichs, K., Chamberlain, S.: Managing uncertainty in moving objects databases. *ACM Trans. on Databases Systems* 29(3), 463–507 (2004)
11. Yu, B.: A spatiotemporal uncertainty model of degree 1.5 for continuously changing data objects. In: *Proceedings of ACM Int. Symposium on Applied Computing, Mobile Computing and Applications*, pp. 1150–1155 (2006)
12. Yu, B., Kim, S.H., Alkobaisi, S., Bae, W.D., Bailey, T.: The Tornado model: Uncertainty model for continuously changing data. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 624–636. Springer, Heidelberg (2007)

Reordering of Location Identifiers for Indexing an RFID Tag Object Database^{*}

Sungwoo Ahn and Bonghee Hong

Department of Computer Engineering, Pusan National University,
San 30, Jangjeon-dong, Geumjeong-gu, Busan 609-735, Republic of Korea
{swan, bhhong}@pusan.ac.kr

Abstract. The query performance for tracing tags depends upon the distribution of tag trajectories in the data space. We examine a more efficient representation of tag trajectories by means of ordering the set of domain values. Our analysis shows that the order of Location IDentifiers (LIDs) can give a great impact on the efficiency of query processing. To find out the optimal ordering of LIDs, we propose a new *LID proximity function* to rearrange an arbitrary order of LIDs. This function enables logically adjacent tag trajectories, which are frequently accessed together, to be stored in close proximity on the disk. To determine the optimal sequence of LIDs in the domain, we also propose a *reordering scheme of LIDs*. The experimental results show that the proposed reordering scheme achieves better query processing than the previous methods of assigning LIDs.

1 Introduction

An efficient query processing of tracing tags can be achieved by providing the repository for tag data [2][5][9]. EPCglobal, a leader in standards management and development for RFID related technologies, proposes EPC Information Service (EPCIS) as the repository for tag events [5][9]. Tag data stored and indexed in the EPCIS consists of the static attribute data and the timestamped historical data.

Historical information is collected and updated whenever each tag is identified by an RFID reader. Among historical information, an RFID application uses Location Identifier (LID), Tag Identifier (TID), and the identified time (TIME) as predicates for tracking and tracing tags [1][9]. To index these values efficiently, we can define the *tag interval* by means of two tag events generated when the tag enters and leaves a specific location. The tag interval is represented and indexed as a time-parameterized line segment in a three-dimensional domain defined by LID, TID, and TIME axes [1].

Logical closeness between tag intervals is very important for simultaneous accesses during a query. It gives a great influence on the performance of query processing because the cost of disk accesses depends on the sequence of storing tag intervals on the disk. For example, assume that a query, Q_i , searches tag intervals using the

^{*} “This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)” (The Regional Research Universities Program/Research Center for Logistics Information Technology).

index structure. If all tag intervals accessed by Q_i are stored in P3 as shown in Fig. 1-(a), a query processor only needs to access one disk page, P3. If these tag intervals are dispersed across disk pages, P2, P3, and P5 as shown in Fig. 1-(b), a query processor usually incurs the additional cost of accessing two pages, P2 and P5. To minimize the cost of disk accesses, the logical closeness between tag intervals in the same disk page must be higher than the logical closeness to others.



Fig. 1. An example of different access cost of the disk

Most work for clustering spatial objects have used the spatial distance in the spatial domain as the measure of the logical closeness [6][7]. To diminish the number of disk accesses at answering spatial queries, they stored adjacent objects sequentially based on the spatial proximity. In addition to the spatial proximity, moving object databases [3][8] have applied the temporal proximity to the characteristic for the distance measure in the time domain. Previous works assumed that all domains on the data space provide the proper proximity about measuring the distance between domain values. Similarly, all domains for tag intervals should also provide adequate proximity to keep to the correlation between the distance and the logical closeness.

The problem is that there is no rule for assigning LIDs to RFID locations in order to ensure this property. If LIDs are arbitrarily ordered in the domain without considering proximity, tag intervals are scattered across the data space irrespective of their logical closeness. Because this situation causes random disk accesses for searching logically adjacent tag intervals, the cost of query processing is increased.

To solve this problem, we propose a reordering method for arranging LIDs in the domain. The basic idea is to compute the distance between two LIDs to fix the logical closeness between tag intervals. To do this, we define a proximity function based on a new *LID proximity* between two LIDs. To determine a sequence of LIDs based on LID proximity, we construct a weighted graph and generate the ordered LID set.

In the next section, we examine the path of tag flows based on characteristics of RFID locations and tag movements, and define the LID proximity function. In Section 3, we propose a reordering scheme of LIDs, using a weighted graph. Section 4 presents experimental results of performance evaluation for the proposed reordering scheme. A summary is presented in Section 5.

2 Proximity between Location Identifiers

Queries for tracing tags are classified into two types according to a kind of restricted predicate as shown in Table 1. An *observation query* (OQ) is used to retrieve tags that are identified by the specified locations within the specified time period. A *trajectory query* (TQ) is used to retrieve the locations that the specific tags enters and leaves

Table 1. Query classification for tracing tag locations

Predicate			Query results	Query types
LID	TID	TIME		
point/set/range	*	point/range	TID(s)	Observation Query (OQ)
*	point/set/range	point/range	LID(s)	Trajectory Query (TQ)

within the specified time period. Queries in Table 1 can be extended as a combined query by performing two queries in the order OQ and TQ.

There are two types of location related to tag events based on the business perspective for an RFID location. Tagged items always move between *business locations* (BizLoc) by traversing *read points* (RP) placed at the entrance of each BizLoc [5]. If there are no RPs connecting specified BizLocs, however, the tagged item cannot move directly between two BizLocs. Although RPs exist between two particular BizLocs, the tag movement can be restricted because of a business process of an applied system.

Based on these restrictions, there is a predefined path which a tag can cross. We designate this path as the *path of tag flows* (FlowPath). Tags attached to items generate a flow of tags traversing the path. FlowPath is a simple method for representing the connection property between two BizLocs. It is possible to generate FlowPath with a connected graph of BizLocs and RPs. If one or more RPs connect two particular BizLocs, they are represented as a single line connecting two LIDs. The FlowPath from $BizLoc_i$ to $BizLoc_j$ is denoted as $FlowPath_{i \text{ to } j}$.

Since most RFID applications are concerned with the historical change of locations for the specific tag, the repository uses the BizLoc as the LID predicate for tracing tags [5][9]. This implies that tag intervals generated by BizLocs along the specific FlowPath have a higher probability of simultaneous access than others. Therefore, it is necessary to reorder LIDs based on the properties of FlowPath. We first define the proximity between LIDs for application to the LID reordering as follows.

Definition 1. LID proximity (LIDProx) is the closeness value between two LIDs in the LID domain of an index. We denote the LID proximity between LID_i and LID_j as $LIDProx_{ij}$ or $LIDProx_{ji}$.

The LID proximity between two LIDs has following properties.

- (1) Any LID_i in the LID domain must have an LID proximity value for any LID_j , where $i \neq j$.
- (2) $LIDProx_{ij}$ is equal to $LIDProx_{ji}$, for all LIDs.
- (3) If there is no LID_k , for which $LIDProx_{ij} < LIDProx_{ik}$, the closest LID to LID_i is LID_j .

For applying dynamic properties based on FlowPath to the LID proximity, we define the *LID proximity function* as shown in Eq. 1; we denote T as the time to compute the LID proximity, $LIDProx_T(i, j)$ as the LID proximity function at time T , and $LIDProx\_OQ(i, j)$ and $LIDProx\_TQ(i, j)$ as proximity functions by properties of an observation query and trajectory query, respectively.

$$\text{LIDProx}_T(i, j) = \alpha \times \text{LIDProx\_OQ}(i, j) + (1 - \alpha) \times \text{LIDProx\_TQ}(i, j) \quad (1)$$

$\text{LIDProx}(i, j)$ is a time-parameterized function; the closeness value between LID_i and LID_j changes over time. To consider the closeness value for an observation query and a trajectory query simultaneously, the function calculates the sum of $\text{LIDProx\_OQ}(i, j)$ and $\text{LIDProx\_TQ}(i, j)$ with the weight value. The weight α determines the applied ratio between two proximity functions as shown in Eq. 2; we denote $\text{OQ}_{ij,t}$ as the number of observation queries for LID_i and LID_j at time t and $\text{TQ}_{ij,t}$ as the number of trajectory queries for LID_i and LID_j at time t .

$$\alpha = \begin{cases} 0 \text{ or } 1 & \text{if no queries are processed} \\ & \text{for } \text{LID}_i \text{ and } \text{LID}_j \\ \sum_{t=1}^T \text{OQ}_{ij,t} / \sum_{t=1}^T (\text{OQ}_{ij,t} + \text{TQ}_{ij,t}) & \text{otherwise} \end{cases} \quad (2)$$

$\text{LIDProx\_OQ}(i, j)$ computes the LID proximity for an observation query with the ratio of tag intervals generated by LID_i and LID_j to all tag intervals as shown in Eq. 3; we denote $\text{TI}_{i,t}$ as the number of tag intervals by LID_i at t , and σ_{OQ} and δ_{OQ} as weight values for $\text{LIDProx\_OQ}(i, j)$.

$$\text{LIDProx\_OQ}_T(i, j) = \frac{\delta_{\text{OQ}}}{\sigma_{\text{OQ}}} \times \left(\sum_{t=1}^T (\text{TI}_{i,t} + \text{TI}_{j,t}) / \sum_{t=1}^T \sum_{a=1}^n \text{TI}_{a,t} \right) \quad (3)$$

Because of the influence of the tag's flow on the LID proximity, we should consider the distribution of tag intervals over time. Equation 4 represents dynamic properties of the tag interval distribution. The difference in the distribution of tag intervals in time domain can be represented by the standard deviation of tag intervals. To apply this difference to the LID proximity, the variable σ_{OQ} in Eq. 4 is used as a weight which is inversely proportional to the number of tag intervals; we denote σ_{OQ} as the standard deviation of tag intervals by LID_i and LID_j and $\overline{\text{TI}}_i$ as the average number of tag intervals by LID_i until T .

$$\sigma_{\text{OQ}} = \sqrt{\frac{1}{T} \times \sum_{t=1}^T \{ (\text{TI}_{i,t} + \text{TI}_{j,t}) - (\overline{\text{TI}}_i + \overline{\text{TI}}_j) \}^2} \quad (4)$$

$$\delta_{\text{OQ}} = \left(\sum_{t=1}^T (\text{STI}_{i,t} + \text{STI}_{j,t}) / \sum_{t=1}^T (\text{TI}_{i,t} + \text{TI}_{j,t}) \right) \times \left(1 / \sum_{t=1}^T \text{OQ}_{ij,t} \right)$$

The hit ratio of tag intervals for an observation query is also a factor determining $\text{LIDProx\_OQ}(i, j)$. The variable δ_{OQ} in Eq. 4 computes the proportional weight – the hit ratio of tag intervals for OQ_{ij} ; we denote $\text{OQ}_{ij,t}$ as the number of observation queries for LID_i and LID_j at t and $\text{STI}_{i,t}$ as the number of results by LID_i for $\text{OQ}_{ij,t}$.

The LID proximity for a trajectory query must consider the pattern of tag movements along FlowPath because a trajectory query is concerned with LIDs traversed by a tag in the specified time period. Equation 5 shows the LID proximity function for a trajectory query retrieving tag intervals by LID_i and LID_j . This function, denoted by $\text{LIDProx\_TQ}(i, j)$, obtains the simultaneous access probability of LID_i and LID_j through the ratio of tag movements between LID_i and LID_j to the total number of tag

movements for all LIDs; we denote $TM_{i \text{ to } j, t}$ as the number of tag movements from LID_i to LID_j , and σ_{TQ} and δ_{TQ} as weight values for $LIDProx\_TQ(i, j)$.

$$LIDProx\_TQ_T(i, j) = \frac{\delta_{TQ}}{\sigma_{TQ}} \times \left(\sum_{t=1}^T (TM_{i \text{ to } j, t} + TM_{j \text{ to } i, t}) \right) / \left(\sum_{t=1}^T \left(\sum_{a=1}^n \sum_{b=1}^n TM_{a \text{ to } b, t} - \sum_{c=1}^n TM_{c \text{ to } c, t} \right) \right) \quad (5)$$

By contrast with an observation query, a trajectory query must not consider the distribution of tag intervals per individual LID but that of tag intervals between LIDs – the movements of the specified tag. To do this, we define the standard deviation, σ_{TQ} , for computing the degree of difference in the distribution of tag movements between LID_i and LID_j . We also define the hit ratio of tag intervals by LID_i and LID_j for a trajectory query as δ_{TQ} .

3 Reordering Scheme of Location Identifiers

To define the reordering problem, let us assume that there is a set of LIDs, $LIDSet = \{LID_1, LID_2, \dots, LID_{n-1}, LID_n\}$. To use the $LIDSet$ for the coordinates in the LID domain, an ordered list of LIDs, $OLIDList_i = (OLID_{i,1}, OLID_{i,2}, \dots, OLID_{i,n-1}, OLID_{i,n})$ must initially be determined. To discover the optimal $OLIDList$ for which the LID proximity for all LIDs is a maximum, we first define the linear proximity as follows.

Definition 2. Linear proximity of $OLIDList_a$ ($LinearProx_a$) is the sum of $LIDProx$ s between adjacent OLIDs for all OLIDs in $OLIDList_a$ such that

$$LinearProx_a = \sum_{i=1}^{n-1} LIDProx(i, i+1) \quad (6)$$

With Definition 2, we can define the problem for reordering LIDs in order to retrieve the $OLIDList$ with the maximum access probability as follows.

Definition 3. LID reordering problem (LOP) is to determine an $OLIDList_o = (OLID_{o,1}, OLID_{o,2}, \dots, OLID_{o,n-1}, OLID_{o,n})$ for which $LinearProx_o$ is a maximum, where $LIDSet = \{LID_1, LID_2, \dots, LID_{n-1}, LID_n\}$ and the LID proximity for all LIDs.

LOP is very similar to the well-known Minimal Weighted Hamiltonian Path Problem (MWHP) without specifying the start and termination points. MWHP involves the Hamiltonian cycle with a minimal weight in the graph. To apply LOP to MWHP, it is necessary to convert LOP into a minimization problem because LOP is a maximization problem for finding the order with maximum LID proximity values for all LIDs. Therefore, the weight value for LID_i and LID_j must be $1 - LIDProx(i, j)$. LOP can be treated as a standard Traveling Salesman Problem (TSP) by Lemma 1.

Lemma 1. LOP is equivalent to TSP for a weighted graph $G = (V, E, w)$ such that

- $V = LIDSet \cup \{v_0\}$ where v_0 is an artificial vertex for solving MWHP by TSP
- $E = \{(LID_i, LID_j) \mid LID_i, LID_j \in LIDSet, i \neq j\} \cup \{(LID_i, v_0) \mid LID_i \in LIDSet\}$
- $w : E \rightarrow R, w(i, j) = 1 - LIDProx(i, j) = 1 - LIDProx(j, i) = w(j, i), w(i, v_0) = w(v_0, i) = 0$

Proof. The graph G contains Hamiltonian cycles because G is a complete and weighted graph. Assume that a minimal weighted Hamiltonian cycle produced in G is HC where $HC = ((v_0, OLID_{a,1}), (OLID_{a,1}, OLID_{a,2}), \dots, (OLID_{a,n-1}, OLID_{a,n}), (OLID_{a,n}, v_0))$ and $OLID_{a,i} \in LIDSet$. If two edges, $(v_0, OLID_{a,1})$ and $(OLID_{a,n}, v_0)$, containing the vertex v_0 are eliminated from HC , we can get a minimal weighted Hamiltonian path L in G from $OLID_{a,1}$ to $OLID_{a,n}$. The weight of HC is identical to a path L because all of edges eliminated in order to produce the path L contain the vertex v_0 , and weights of these edges are zero. The produced path L is translated as an ordered LID list, $OLIDList_a$ where $OLIDList_a = (OLID_{a,1}, OLID_{a,2}, \dots, OLID_{a,n-1}, OLID_{a,n})$. Because of this, the reordering of LIDs is defined as a solution of the corresponding TSP for obtaining HC in the weighted graph G . ■

Because TSP is a NP-complete problem, an exhaustive exploration of all cases is impractical [11]. To solve TSP, we used GA [4] among several heuristic methods to determine the ordered LIDSet using the weighted graph G . Heuristic approaches can be used to find solutions to NP-complete problems in much less time. Although it might not find the best solution, it can find a nearly perfect solution – the local optima.

4 Experimental Evaluation

We evaluated the performance of our reordering scheme by applying LIDs as domain values of an index. We also compared it with the numerical ordering of LIDs using a lexicographic scheme. To evaluate the performance of queries, TPIR-tree [1], R*-tree [10], and TB-tree [3] are constructed with the axes TID, LID, and TIME. Since each index uses original insert and/or split algorithms, their essential properties are fixed.

Since well-known and widely accepted RFID data sets for experimental purpose do not exist, we conducted our experiments with synthetic data sets generated by Tag Data Generator (TDG). TDG generates tag events which can be represented as a tag interval based on the data model of [1]. To reflect the real RFID environment, TDG allows the user to configure its specific variables. All variables of TDG are based on properties of FlowPath, and tag movements along FlowPaths. According to user-defined variables, tags are created and move between BizLocs through FlowPaths. TDG generates a tag interval based on a tag event occurring whenever a tag enters or leaves a BizLoc.

We assigned an LID to each BizLoc by a lexicographic scheme of TDG based on the spatial distance. To store trajectories of tags in the index, TDG produces tag intervals from 100,000 to 500,000. Since the LID proximity function uses the quantity per query, OQ and TQ, as the variable, we must process queries through the index structure during TDG produces tag intervals. To do this, we processed 10,000 queries for tracing tags continuously, and estimated query specific variables over all periods. Finally, the sequence of LIDs based on the LID proximity is determined by computing the proximity value between LIDs until all tag events are produced.

All experiments presented below are performed using the TDG data set, with 200 BizLocs. To measure the average cost, all experiments are iteratively performed 10 times per data set. In figures for results, we renamed the index by attaching an

additional word with a parenthesis in order to distinguish each index according to the arrangement of LIDs. Original means the index using the initial arrangement of LIDs in the LID domain. Reorder means the index based on the LID proximity.

Experiment 1. Measuring the performance for only one query type

To measure the performance of each query type, we evaluated the performance of queries for which only one query type is processed. To obtain an optimized order of LIDs per query type, we processed 10,000 OQs in Fig. 2-(a) and 10,000 TQs in Fig. 2-(b) before the LID reordering is processed.

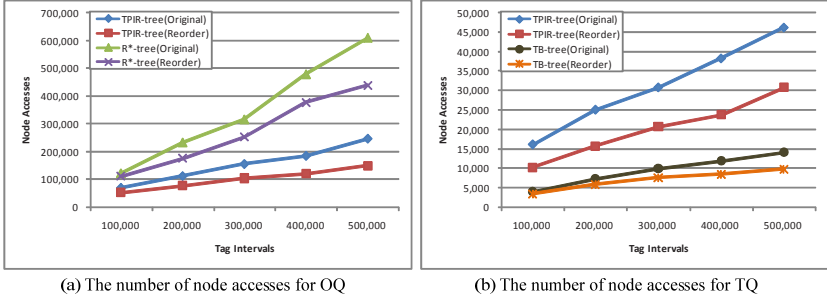


Fig. 2. Performance evaluation for indexes where only one type of query is used

Figure 2 shows the performance comparison between Original and Reorder. Figure 2-(a) and 2-(b) are related to the performance of OQ and TQ, respectively. Each query set includes 1,000 OQs or TQs. We discovered that Reorder can retrieve results with a lower cost of node accesses than Original for all cases. In general, the performance of Reorder is slightly better than the performance of Original, for the data set of 100,000 tag intervals. Nevertheless, Reorder still outperforms Original during tag intervals which are continuously generated and inserted in the index. The search performance of OQ and TQ is improved by a maximum of 39% and 33%, respectively. This experiment tells us that the LID proximity can measure the closeness between BizLocs more precisely when tag movements and queries for these tags continuously occur.

Experiment 2. Performance comparison in case of processing OQ and TQ altogether. Regardless of whether better performance was achieved than the initial arrangement of LIDs in Experiment 1, we need to measure the performance in the case where OQ and TQ are processed simultaneously. To do this, we performed the experimental evaluation as shown in Fig. 3. Since the LID proximity must reflect properties of all query types simultaneously unlike the experiments in Fig. 2, we processed both of 5,000 OQs and 5,000 TQs before the proximity is measured. For evaluating the query performance, 1,000 OQs or TQs are also processed.

Figure 3 shows that the number of node accesses of Reorder is increased, as compared with that in Fig. 2. This is because $LIDProx\_OQ_T(i, j)$ and $LIDProx\_TQ_T(i, j)$ in Eq. 1 are detrimental to the performance of a query unrelated to each proximity when OQ and TQ are processed simultaneously. The performance of Reorder is nevertheless better than the performance of Original at processing all of OQ and TQ.

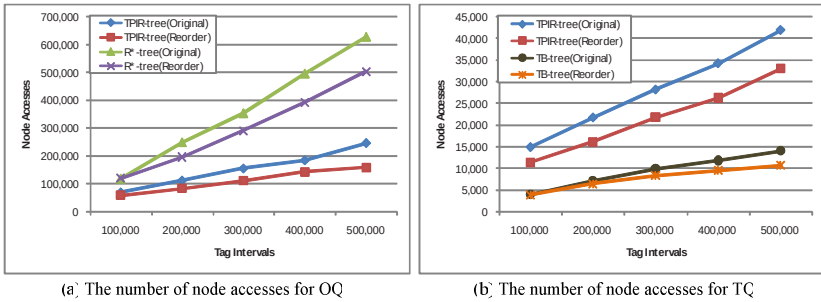


Fig. 3. Performance evaluation for indexes when processing both queries simultaneously

5 Conclusions

We addressed the problem of using the Location Identifier (LID) as the domain value of the index for tag intervals, and proposed a solution to this problem. The basic idea for solving this problem is to reorder LIDs by a new LID proximity function. By using the LID proximity function, we can discover the distance of two LIDs in the domain, to ensure the logical closeness between tag intervals. Our experiments show that the newly proposed reordering scheme outperforms the previous scheme of assigning LIDs. Future work will explore the issues of the dynamic updating of the tag interval index according to the changing LID proximity.

References

1. Ban, C.H., Hong, B.H., Kim, D.H.: Time Parameterized Interval R-tree for Tracing Tags in RFID Systems. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 503–513. Springer, Heidelberg (2005)
2. Lin, D., Elmongui, H.G., Bertino, E., Ooi, B.C.: Data Management in RFID Applications. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 434–444. Springer, Heidelberg (2007)
3. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel Approaches to the Indexing of Moving Object Trajectories. In: International Conference on VLDB, pp. 395–406 (2000)
4. Whitley, D.: A Genetic Algorithm Tutorial. *Statistics and Computing* 4, 65–85 (1994)
5. EPC global: EPC Information Services (EPCIS) Specification, Ver. 1.0, EPC global Inc., (2006)
6. Jagadish, H.V.: Linear Clustering of Objects with Multiple Attributes. *ACM SIGMOD* 19(2), 332–342 (1990)
7. Kamel, I., Faloutsos, C.: On Packing R-trees. *CIKM*, 490–499 (1993)
8. Mokbel, M.F., Ghanem, T.M., Aref, W.G.: Spatio-temporal Access Methods. *IEEE Data Engineering Bulletin* 26(2), 40–49 (2003)
9. Harrison, M.: EPC Information Service – Data Model and Queries, Technical Report, Auto-ID Center (2003)
10. Beckmann, N., Kriegel, H.P.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *ACM SIGMOD* 19(2), 322–331 (1990)
11. Skiena, S.S.: *The Algorithm Design Manual*. Springer, Heidelberg (1998)

A Free Terrain Model for Trajectory K -Anonymity

Aris Gkoulalas-Divanis^{1,2} and Vassilios S. Verykios^{1,2}

¹ Department of Computer & Communication Engineering
University of Thessaly, Volos, Greece
{arisd, verykios}@inf.uth.gr

² Research and Academic Computer Technology Institute,
Patras University Campus, GR-26500 Rio, Greece

Abstract. This paper introduces a privacy model for location based services that utilizes collected movement data to identify parts of the user trajectories, where user privacy is at an elevated risk. To protect the privacy of the user, the proposed methodology transforms the original requests into anonymous counterparts by offering trajectory K -anonymity. As a proof of concept, we build a working prototype that implements our solution approach and is used for experimentation and evaluation purposes. Our implementation relies on a spatial DBMS that carries out part of the necessary analysis. Through experiments we demonstrate the effectiveness of our approach to preserve the K -anonymity of the users for as long as the requested services are in progress.

1 Introduction

Technological advances in sensors, wireless communications and GPS receivers gave rise to a series of applications – the so called Location Based Services (LBSs) – that exploit positional data to offer services to their subscribers. The benefit of LBSs both to the individuals and to the community is undeniable. However, without strict safeguards, the deployment of these technologies poses a severe threat to user privacy. In this paper, we consider a population of users who access LBSs to conduct their everyday business. When a user moves, her mobile device periodically submits location updates to a traffic monitoring station. The collection of the location updates results to the construction of user trajectories, upon which the user travelled. The observed regularities in the user trajectories can be used by untrusted entities to breach user privacy, even when no other user identification information is in place (e.g., social security number, family name). For example, think of a scenario where a user goes by certain city areas when she commutes to work in more or less the same day times. This frequent behavior of the user can lead to a possible identification simply by matching (i) the starting point of her travel to some public domain geocoded information for her house and (ii) the ending point to the location of the business facility that she works. Furthermore, a possible matching of a series of user requests with a frequently travelled trajectory part of the user may also easily lead to her personal identification.

Many techniques have been proposed to preserve user privacy in LBSs (c.f. [4,5,6,7]). In this paper, we follow the widely adopted paradigm of a trusted server (see Section 5) and propose a model that builds on the anonymity direction to protect the

privacy of the users when requesting LBSs. Our proposed model makes use of a movement database which stores the time-series of locations visited by the users, along with user specified privacy parameters. The collected information allows the spatial engine of the trusted server to build representative user movement patterns and to provide users with customized anonymity guarantees. Through the use of both the present (captured as the requests that the user sends for LBSs) and the past (captured as the history of user movement), the enforced K -anonymity algorithm, either successfully anonymizes the user request or delays/denies the offering of the requested service.

The remainder of this paper is structured as follows. Section 2 presents the terminology. In Section 3, we present the algorithmic techniques that support the proposed privacy methodology and in Section 4 we experimentally evaluate our work. Section 5 presents the related work and Section 6 concludes this paper.

2 Terminology

Let u denote a user on the move. A *location update* is a tuple $\langle u, x, y, t \rangle$ stating that user u was located at point (x, y) at time t , where t is a detailed recording of time including the current date. By using the transmitted location updates of a user, the trusted server reconstructs her *movement history*, represented as a 3D polyline (we consider two dimensions for space and one dimension for time). The user movement history cannot be directly used to draw any significant inferences regarding the movement patterns of the user. For this reason, it is decomposed into semantically meaningful parts, the *trajectories*, each consisting of a sequence of location updates. The decomposition strategy depends on application specific parameters that signify the ending of the current user trajectory and the beginning of a new trajectory. Any part of a user trajectory, independently of its start point and end point, is called a *route*. A *frequent route* is defined as follows:

Definition 1. A route of a user u is frequent if it appears among the trajectories of u a number of times that is larger than a minimum frequency threshold $freq$.

The number of times that a route appears in the trajectories of u is called the *frequency* of the route. Any route that is not frequent is called *infrequent*. Directly related to the notion of a frequent route is the notion of an “unsafe” (equiv. “safe”) route.

Definition 2. A route of a user u is unsafe (equiv. safe) if it is frequent for u and infrequent (equiv. frequent) for a system-defined number of other users in the system.

A request is a tuple of the form $R = \langle u, sid, x, y, t \rangle$, stating that user u requests service sid from location (x, y) at time t . The system makes the best effort to guarantee that a user remains K -anonymous when requesting LBSs, defined as follows:

Definition 3. A user u , when submitting a request is K -anonymous if each of the $K - 1$ participants of his or her anonymity set (c.f. [6]) could be a potential issuer of the request, as seen by the trusted server, from within an area that is near to the requester and within a time period that is close to the time of request.

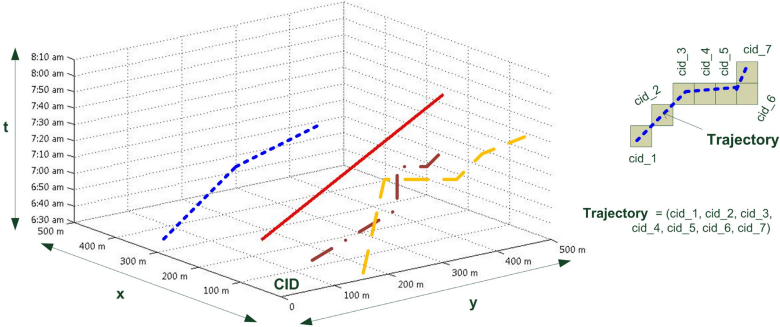


Fig. 1. A 3D grid for the spatio-temporal generalization and the derivation of the “unsafe” routes

3 The Free Terrain Model

In this section, we present our proposed privacy model that consists of two phases. The first phase is responsible for the computation of the unsafe routes for each user in the system. The second phase, uses the computed unsafe routes to offer customized trajectory K -anonymity to the requesters of LBSs.

3.1 Phase I: Derivation of the Unsafe Routes

To derive the “unsafe” routes, we overlay a 3D grid over the total area covered by the trusted server and discretize the user movement into a set of spatio-temporal regions or *cells*. In the partitioned 3D space (see Fig. 1), a trajectory of a user u is depicted as a tuple $\langle u, s \rangle$, where s is the sequence of cell identifiers of the cells that contain a part of the user trajectory. This is shown in Fig. 1 (right), where a trajectory is decomposed into a set of *CIDs*. Since each *CID* is referenced in time, without any loss of information, we consider s to be an unordered set instead of a sequence of elements [1]. Thus, the user movement history database collects all users’ trajectories as a series of transactions $T = \langle u, tid, s \rangle$ (tid is the unique identifier of the trajectory). Through this process, a set of spatio-temporal data (3D polylines) is transformed into market-basket data (cells referenced in space and time), where each trajectory is mapped to a transaction and each point within the trajectory is mapped to an item of the transaction. To identify the frequent routes of a user, we apply frequent itemset mining [2] to the transactions produced from the transformation of all the user’s trajectories in the system, as collected from the history of user movement. For the purposes of this process, the time recordings in the user trajectories are transformed into unanchored time intervals in the 3D grid to depict time spans in a *generic* day.

Given the user movement history database $\mathcal{D}$, let $\mathcal{D}_u$ be the portion of $\mathcal{D}$ that collects the trajectories of user u . Accordingly, let $\mathcal{D}_{\bar{u}}$ contain the transactions T in $\mathcal{D} - \mathcal{D}_u$. Algorithm 1 can be applied as a patch to any frequent itemset mining approach to enable the discovery of the “unsafe” routes from the frequent routes of a user. The algorithm considers the frequent itemsets in $\mathcal{D}_u$ by examining if they are also frequent in $\mathcal{D}_{\bar{u}}$.

Algorithm 1. Derivation of unsafe routes in the free terrain model

```

1: procedure UNSAFEPATHS( $\mathcal{D}_u, \mathcal{D}_{\bar{u}}, freq, MSO$ )
2:    $\mathcal{D}_s \leftarrow \mathcal{D}_{\bar{u}}$  sorted based on the id of each user  $u$ 
3:   foreach itemset  $I \in \mathcal{D}_u, frequency(I) \geq freq$  do                                 $\triangleright$  route is frequent
4:     if  $RSUP(I, \mathcal{D}_s, freq) \geq MSO$  then                                        $\triangleright$  route is "safe" so discard it
5:       treat  $I$  as infrequent in  $\mathcal{D}_u$ 

6: function  $RSUP(I, \mathcal{D}_s, freq)$ 
7:   count  $\leftarrow 0$ 
8:   foreach distinct user  $u \in \mathcal{D}_s$  do
9:      $T = \{ \langle u, tid, s \rangle \mid I \in s \}$ 
10:    if  $|T| \geq freq$  then
11:      count  $\leftarrow$  count + 1
12:   return count

```

Algorithm 2. Match of requests in the free terrain model

```

1: function MATCHREQTOUPATH( $\mathcal{C}, \mathcal{O}, R_c, frun, R_p$ )
2:    $CID\ L_c \leftarrow \mathcal{H}(R_c)$ 
3:   if  $frun = \text{true}$  then                                                          $\triangleright$  seek for an initial match
4:     foreach  $l \in \mathcal{C}$  do
5:       if  $l.cell = L_c$  then
6:         return true                                                          $\triangleright$  match was found
7:   else                                                                        $\triangleright$  seek for a subsequent match
8:      $CID\ L_p \leftarrow \mathcal{H}(R_p)$ 
9:     if  $\mathcal{O}[L_p][L_c] = 1$  then
10:      return true                                                          $\triangleright$  match was found
11:   return false                                                          $\triangleright$  request was made from a "safe" location

```

3.2 Phase II: Trajectory K-Anonymity

The offering of trajectory K -anonymity involves two steps. The first step (Alg. 2) examines the potential matching of the location of the requester to one of her "unsafe" routes. Provided that such a matching exists, the second step (Alg. 3) offers privacy to the user by means of trajectory K -anonymity.

Matching of a User Request to an Unsafe Route. Algorithm 2 presents a methodology for the determination of (i) a matching of the initial user request R_c to one of the user's "unsafe" routes, and (ii) the subsequent matches until the completion of the service. The algorithm uses a function $\mathcal{H} : \langle x, y, t \rangle \rightarrow CID$ (implemented on the spatial DBMS) that maps the user location to the appropriate CID . Variable $frun$ enables the algorithm to distinguish between the search for an initial and a subsequent match.

In an initial match, R_c corresponds to the current request. The algorithm scans the list of locations $\mathcal{C}$, derived as the union of all the $CIDs$ that appear in all the "unsafe" routes of the user, to identify if the current CID of the user is part of an "unsafe" route. If the CID matches an element of $\mathcal{C}$, then trajectory K -anonymity should be offered to the user. Otherwise, the request is anonymized just by removing the obvious identifiers. In a subsequent match, R_p corresponds to the previous location update, while R_c corresponds to the current location and time of the user. Supposing that the user has moved to a new cell, a transition table $\mathcal{O}$ identifies if the user still moves within an "unsafe" route. The transition table has $\mathcal{O}[i][j] = 1$ if the $CIDs$ i, j are consecutive elements in any of the user's "unsafe" routes. In a successful match of the user's request to one of her "unsafe" routes, the request has to become K -anonymous.

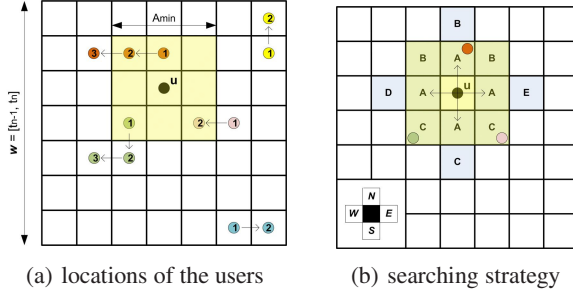


Fig. 2. Trajectory K -anonymity in the free terrain model

Offering of Trajectory K -Anonymity. To provide K -anonymity, we consider a time interval w that extends from the time of request t_n to some moments back t_{n-1} and identify the routes of all users in the system with respect to w . Two parameters regulate the spatial extend of the generalization area. The first parameter, A_{max} , defines the maximum generalization that guarantees the reliable operation of the requested service. The second parameter, A_{min} , defines the minimum area where the $K - 1$ users of the anonymity set should be found so that the user is adequately covered up.

Fig. 2(a) shows the trajectories of five users during a time interval w and the cell of request for a user u . Based on the location accuracy that is necessary for the requested service, the system decides on the appropriate size for w and creates the corresponding (x, y) -projection. The numbers in the colored dots depict the sequence of cells that were visited by each other user in the system. Fig. 2(a) provides the 2D grid that constitutes the search space for the provision of K -anonymity, while Fig. 2(b) demonstrates the applied search strategy. In Fig. 2(b), each cell has four neighbors; North, South, West, and East. First, the cell where u is located is checked against the requirements of K and A_{min} . If the requirements are not met, the system identifies its neighboring cells (denoted as ‘A’ in Fig. 2(b)) and checks for each of them if their region, combined to the already searched region, satisfies these requirements. This process iterates in a BFS manner, until either the requirements are met or the explored area exceeds A_{max} .

Algorithm 3 has the details for the provision of K -anonymity in the free terrain model. In a service that requires multiple location updates for its completion, the following strategy is applied. At each location update, the trusted server checks if the previously computed neighbors of the user have changed cell and for those who have, if their new cell lies within the previous anonymity region. If this holds, it adjusts the temporal information of the request and maintains the same anonymity area. If some of the $K - 1$ subjects have left the region, the same number of subjects are sought among the other users in the system. If the requested number of missing subjects cannot be found in this region, Alg. 3 is applied to compute a new region of K -anonymity.

4 Experimental Evaluation

The proposed algorithms were implemented in Java and Oracle 10g. The methodology was tested on the publicly available INFATI dataset that captures the movement of 20 cars, recorded at a sampling rate of one second. More information can be found at [3].

Algorithm 3. K -anonymity in the free terrain model

```

1: function GENERALIZEREQUEST( $R_c, K, A_{min}, A_{max}$ )
2:   declare  $CID\ L_c \leftarrow \mathcal{H}(R_c)$ 
3:   declare  $CID\_list\ N[4] \leftarrow \emptyset$ 
4:   declare Hash  $CS \leftarrow \emptyset$ 
5:   declare Hash  $Users \leftarrow \emptyset$ 
6:   declare Queue  $Q \leftarrow \emptyset$ 
7:   declare Integer  $Vcells \leftarrow 0$ 
8:   ENQUEUE( $Q, L_c$ )
9:    $Users\{u\} \leftarrow 1$ 
10:  while  $Q \neq \emptyset \wedge (Vcells \times area\_of\_cell \leq A_{max})$  do
11:     $CID\ l \leftarrow DEQUEUE(Q)$ 
12:     $CS\{l\} \leftarrow 1$ 
13:    foreach user  $\alpha \in cell\ l$  do
14:       $Users\{\alpha\} \leftarrow 1$ 
15:       $Vcells \leftarrow Vcells + 1$ 
16:      if  $(Vcells \times area\_of\_cell \geq A_{min}) \wedge (\%keys(Users) \geq K)$  then
17:        return  $CS$  ▷ alternatively, return MBR( $CS$ )
18:       $CID\_list\ N \leftarrow GETNEIGHBORCELLS(l)$ 
19:      for  $int\ i = 0; i < 4; i++$  do
20:        if  $N[i] \notin keys(CS)$  then
21:          ENQUEUE( $Q, N[i]$ )
22:  return null ▷  $K$ -anonymity cannot be provided

```

To decompose the history of users' movement into trajectories, we considered a trajectory to be a set of GPS measurements such that no two consecutive readings have a time difference of more than five minutes. To capture the "unsafe" routes, we considered $100meters \times 100meters \times 5minutes$ cells in a grid where the actual GPS readings were replaced by the ID of the cell they fell into. Each trajectory was mapped to a transaction in the universe of the cell identifiers and Alg. 1 was used to derive the "unsafe" routes for each driver. Fig. 3 presents the "unsafe" routes of two cars, when $freq = 2$ and $MSO = 5$. To enhance the visibility of these graphs, we removed the most distant outliers.

To create the terrain for the provision of K -anonymity, we considered the (x, y) -projection of all the trajectories for all the cars of team 1, excluding any distant outliers. The MBR of these projections was split into a grid of $100meters \times 100meters \times 5minutes$ cells. In our experiments, we considered A_{min} to equal the area of a cell and a maximum generalization threshold such that all the $K - 1$ subjects of the anonymity set are at most 1 Km far from the requester. Since the drivers in the INFATI data are few for the purposes of our experiments, we randomly assign a set of drivers U in each car, with each driver being located at a different position in the movement history of the car. In what follows, we are only interested in time periods (rather than actual times) in the users' movement history. Thus, we assume that all users move synchronously based on the (x, y) locations in the location updates of their assigned trajectories and disregard the time t of their location updates. Although the actual time t is omitted, the time difference between the consecutive location updates in a trajectory is maintained. We use this information to compute the locations of the corresponding user within the last 10 minutes. In the case of team 1, this process allows us to compute the location of all $11U$ drivers. To create a user request for an LBS, we randomly choose U location updates, each from one of the 10 trajectories and $U - 1$ location updates from the last trajectory, and assign each user to one of these points. The last user is positioned in

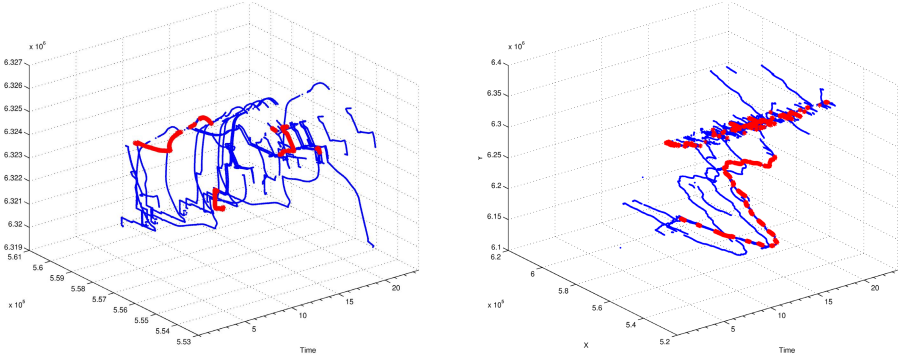


Fig. 3. The “unsafe” routes (red) of the trajectories (blue), using $msup = 2$, $MSO = 5$

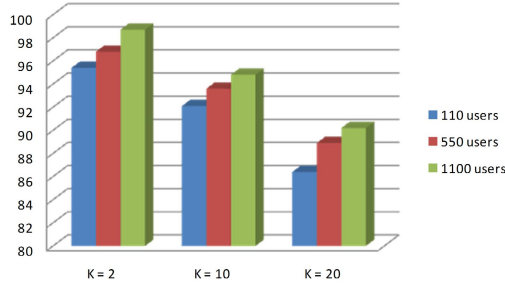


Fig. 4. Success ratios for 2, 10, 20-anonymity and 110, 550, 1100 users

the final trajectory such that she is located within one of her “unsafe” routes. This is also the point of request. The assignment of the users to the cars is performed in a way that users assigned to the same car have no overlapping trajectories for the 10 minute period that is examined. The trajectory K -anonymity process uses the user movement history to compute the cells where the user was located within the last 10 minutes. Figure 4 presents the success ratio for $K = \{2, 10, 20\}$ and for a number of users $U = \{10, 50, 100\}$ assigned to each trajectory. The reported ratio is an average over 100 runs coming from different users and “unsafe” routes.

5 Related Work

Gruteser and Grunwald [4] propose the use of spatial along with temporal cloaking to provide location anonymity in LBSs. The spatial cloaking is achieved by recursive subdivision of the entire area into equi-size quadrants. The temporal cloaking delays the servicing of the request until $K - 1$ other users have visited the selected quadrant. *Gedik and Liu* [5] assign an expiration threshold to each request, during which it can be retained by the system to be anonymized. The cloaking strategy encloses the requests into spatio-temporal MBRs, generates a graph of the requests whose MBRs overlap, and identifies those cliques in the graph that satisfy the anonymity constraints. *Bettini, et al.*

[6] touch upon the privacy issues in LBSs from the viewpoint of the users' history of requests. Their approach keeps track of the users' history of location updates, along with a sequence S of spatio-temporal constraints that act as a pseudo-identifier for the user. The user is offered trajectory K -anonymity when requesting an LBS from a location that is included in S . Mokbel, *et al.* [7] partition the entire area in a grid fashion and organize it in a pyramid structure. The authors use bottom-up cloaking to iterate over the different layers of the pyramid and identify the cells that contain the requester along with $K - 1$ other users.

6 Conclusions

In this paper, we introduced a privacy model that offers customized trajectory K -anonymity to the requesters of LBSs. The proposed algorithms make use of the spatial capabilities of a database engine to allow for an efficient implementation. By using the privacy requirements of each individual and a strategy for the identification of the "unsafe" routes, our approach protects the privacy of the users when they are at risk.

Acknowledgements

This research has been partially funded by the European Union under the FP6-IST-FET programme, Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledge Discovery and Delivery, www.geopkdd.eu.

References

1. Gidófalvi, G., Pedersen, T.B.: Mining long, sharable patterns in trajectories of moving objects. In: Third Workshop on Spatio-Temporal Database Management (2006)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: 20th International Conference on Very Large Databases, pp. 487–499 (1994)
3. Jensen, C.S., Lahrman, H., Pakalnis, S., Runge, J.: The INFATI data. Time Center TR-79 (2004), <http://www.cs.aau.dk/TimeCenter>
4. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: First International Conference on Mobile Systems, Applications, and Services, pp. 31–42 (2003)
5. Gedik, B., Liu, L.: A customizable K-anonymity model for protecting location privacy. Technical report, Georgia Institute of Technology (April 2004)
6. Bettini, C., Wang, X.S., Jajodia, S.: Protecting privacy against location-based personal identification. In: Second VLDB Workshop on Secure Data Management, pp. 185–199 (2005)
7. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: query processing for location services without compromising privacy. In: 32nd International Conference on Very Large Data Bases, pp. 763–774 (2006)

HRG: A Graph Structure for Fast Similarity Search in Metric Spaces

Omar U. Florez and SeungJin Lim

Computer Science Department
Utah Sate University, Logan, UT 84322-4205, USA

Abstract. Indexing is the most effective technique to speed up queries in databases. While traditional indexing approaches are used for exact search, a query object may not be always identical to an existing data object in similarity search. This paper proposes a new dynamic data structure called Hypherspherical Region Graph (HRG) to efficiently index a large volume of data objects as a graph for similarity search in metric spaces. HRG encodes the given dataset in a smaller number of vertices than the known graph index, Incremental-RNG, while providing flexible traversal without incurring backtracking as observed in tree-based indices. An empirical analysis performed on search time shows that HRG outperforms Incremental-RNG in both cases. HRG, however, outperforms tree-based indices in range search only when the data dimensionality is not so high.

Keywords: Similarity search, Indexing, Data Structures, Query Processing.

1 Introduction

Searching is a fundamental problem in a modern information system. Indexing is the most well-known technique to speed up a search process, and user queries are evaluated for the given selection predicate over the indexed key by exact matching. However, exact matching may not be suitable for complex and less-structured data objects. For example, the probability for two iris patterns to exactly match is very low, even when they belong to the same individual. For this type of data, imprecise search or search with error, based on the notion of *similarity* (or *dissimilarity*) between objects, has a more practical value.

In metric spaces, tree- and graph-based techniques [1,2,3,4,5] have been proposed to reduce the search space and subsequently reduce the overall number of distance computations by virtue of the “triangle inequality.” In [4], for example, a tree structure called *spatial approximation tree* was proposed to approach the given query closer and closer spatially by taking advantage of the inequality property, by which distance computations are performed for a small subset of the entire search space. However, a tree structure has an inherent limitation: the search starts only from the root of the tree, and once it fails to find a reasonable solution to the given query in the current subtree, backtracking may be unavoidable. Although reduction in distance computation in graph-based solutions to

the similarity search problem has been achieved to a certain extent recently, the intrinsic cost overhead in graph construction still makes graphs less competitive than trees as a practical solution, and hence the graph-based similarity search in metric spaces has been noted as an open research problem [6,5,7,8].

In response to the need of an efficient graph-based solution which takes advantage of no backtracking while avoiding extraneous graph construction cost, we propose in this paper a new efficient, graph data structure, called *Hyperspherical Region Graph* (HRG), for the similarity search problem in a metric space consisted of a large volume of data objects, inspired by the *relative neighborhood graph* (RNG) [9]. The issue of high construction cost of the graph with a large number of data objects is mitigated by modeling the search space as an RNG of vertices each of which represents a set of data objects called *hyperspherical region*, instead of a graph of individual data objects. In other words, an RNG reduces the construction cost by having substantially less number of vertices than existing neighborhood graphs.

Our empirical analysis of the proposed HRG in comparison with the graph-based data structure (Incremental-RNG) recently proposed in [5] and two state-of-the-art tree-based structures M-tree [1] and SA-tree [4] shows that HRG always outperforms Incremental-RNG in any case with regard to i) data structure construction time and ii) similarity search performance, both of which are tested by the number of distance computations, response time and memory usage. On the other hand, HRG's performance gain over M-tree and SA-tree was noticeable during search time but not in construction time, which is anticipated for a graph structure compared to a tree. HRG performed better than the two tree indices with the 2D, 3D and 16D datasets but not with the 175D dataset used.

The paper is organized as follows: Section 2 describes the proposed technique. Section 3 shows the experimental results. Finally, we provide a concluding remark in Section 4.

2 Hyperspherical Region Graph

As introduced in Section 1, our strategy in finding a solution to the similarity search problem in a large metric space is to design a graph data structure. We prefer a graph-based approach mainly because additional edges in a graph allow us to traverse the data structure without backtracking to upper vertices which is required in a tree [1,2,3,4]. In addition, the constraint on the root node as a starting point in a tree is relaxed in a graph because any vertex can be a starting point.

Note that the flexibility in graph traversal is enabled at the expense of additional edge insertions. Subsequently, an efficient management of edge insertion must be addressed in order to make a graph practical as an alternative to a tree data structure in similarity search. We now present the graph structure proposed in this paper, called *hyperspherical region graph* (HRG).

2.1 The Graph

The proposed hyperspherical region graph is a type of the relative neighborhood graph (RNG) [9] defined as follows:

Definition 1. *Given a graph $G = (V, E)$, let $H(v_i, v_j)$, for any pair of vertices $(v_i, v_j) \in V^2$, be the hypersphere of radius $d(v_i, v_j)$, for some distance function $d : V \times V \rightarrow \mathbf{R}^+$, with v_i as the center of the hypersphere. Then, G is a relative neighborhood graph if any edge $(v_i, v_j) \in E$ satisfies the neighborhood relationship constraint: $(H(v_i, v_j) \cap H(v_j, v_i)) \cap V = \emptyset$ where $H(v_i, v_j) \cap H(v_j, v_i)$ returns the inner volume of the lune formed by the intersection of the two hyperspheres excluding the lune's surface.* $\square$

In other words, two connected vertices are neighbors in RNG. Intuitively, RNG reduces redundant neighborhood relationships by excluding the edge between adjacent vertices v_1 and v_2 if any vertex v_3 is found in the lune formed by the intersection of the hyperspheres centered at v_1 and v_2 , i.e., if the distance from v_3 to either v_1 or v_2 is smaller than the distance between v_1 and v_2 .

The reason for using RNG instead of other types of graphs such as Delaunay triangulation is that RNG is defined based only on distances between vertices. This characteristic makes RNG a suitable option to perform similarity search not only in Euclidean spaces but also in arbitrary metric spaces. HRG extends the idea of RNG to represent the structure of a metric space using representative vertices of hyperspherical regions as follows:

Definition 2. *A hyperspherical region $H = (v, c, O)$ in a metric space is a hypersphere of objects $\{v, O = \{o_1, o_2, \dots, o_n\}\}$, where (1) v is the center object of H , (2) c is the capacity of H , i.e., the maximum number of objects that can be included in H , and (3) O is the set of member objects that are currently contained in H (with the exclusion of v). Furthermore, the distance between v and the farthest object in H is called the radius of H .* $\square$

From now on, $H(v)$ will be used as a shorthand notation for $H(v, c, O)$ as long as c and O are clear in the context. Furthermore, $c_{H(v)}$ denotes c of $H(v)$, and $|H(v)| = |O| + 1$.

Definition 3. *Given a set of hyperspherical regions $\{H(v_1), \dots, H(v_n)\}$, the hyperspherical region graph $G_H = (V, E)$ is a relative neighborhood graph where $V = \{v_1, \dots, v_n\}$, i.e., the center objects of the regions, and E is the set of edges between neighboring vertices. Furthermore, $c_{H(v_1)} = \dots = c_{H(v_n)}$* $\square$

Since a hyperspherical region graph G_H is consisted of only the vertices that are centers of hyperspherical regions, not of all the data objects, the number of vertices in G_H is generally smaller than that of the RNG of the same dataset, which enables us to construct G_H by a less number of edge insertions.

The realization of this goal involves identifying the regions affected by vertex insertions, and hence we present the search technique used first.

2.2 Similarity Search in HRG: Range Search

A range search is defined by the set of objects which are at most at the distance r from the query object x .

Definition 4. Given a universe $\mathbb{U}$ of data objects, $RangeSearch(x, r) = \{s \in \mathbb{U} | d(x, s) \leq r\}$. $\square$

To reduce the search space in range search, the search is performed at two different levels in our approach: (1) the graph level, and (2) the region level.

Graph-Level Reduction. For the graph-level search space reduction, being a the starting vertex chosen randomly, the distance between x and each of the neighboring vertices of a , denoted $N(a)$, including a , is computed first. Then, a search space reduction technique, inspired by the *spatial approximation* method [4], is applied to visit only the vertices which are “close enough” to x among the vertices in $\{a\} \cup N(a)$ based on the distances. “Closeness” is defined as follows:

Definition 5. Given a set of vertices V and the query object x , $v \in V$ is a “close-enough” vertex to x if $d(x, v) \leq \min\_dist + 2r$ where $\min\_dist = \min_{w \in V} d(x, w)$ and r denotes the given distance tolerance to x from v . $\square$

This search space reduction technique is applied recursively through all the neighboring nodes that are close enough to x as long as such neighbors are found and have not been visited yet. All the visited vertices with no close-enough neighbors other than the originating neighbor are identified as candidate vertices.

Let $\{v_1, \dots, v_k\}$ be the set of candidate vertices of x . Then, any region $H(v_i)$ ($1 \leq i \leq k$) which intersects that of x , i.e., $H(x)$ is returned as a candidate region because they have a higher likelihood of containing the target objects of the range search than the non-intersecting regions.

Region-Level Reduction. The objective of this step is to find the range search result by the given query object x and radius r . The result consists of the objects that are selected from the member objects of the candidate regions by checking if they belong to the range search region centered at x with radius r . In this process, a search space reduction is achieved by using the triangular inequality to avoid computing the distance between each member object o and x , i.e., $d(x, o)$. Note that the distance between the center object v and o is already known. The distance between x and v has been already calculated also. Hence, we can approximate our decision that o belongs to $RangeSearch(x, r)$ if the condition $|d(v, x) - d(v, o)| \leq r$ holds.

Note that it is guaranteed that for any pair of vertices v_1, v_2 in a Delaunay triangulation graph, there exists one or more paths from v_1 to v_2 . Since HRG is an extension of RNG and RNG is a subgraph of Delaunay triangulation, the path of minimum length between v_1 and v_2 exists. Using both search space reduction techniques, we were able to visit 30% less edges in our approach than in SA-tree. Figure 1 contrasts the area explored by our approach and SA-tree when performing a Range Search around the object x . The range search process is summarized in Algorithms 1 and 2.

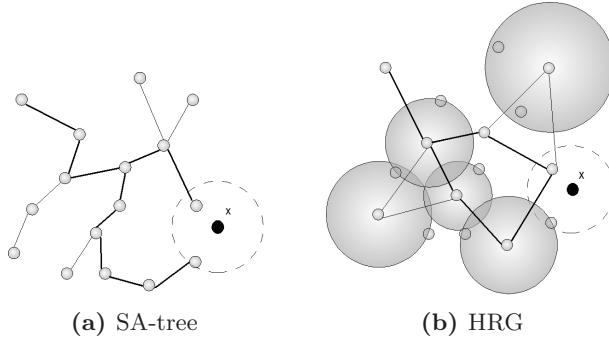


Fig. 1. Edge exploration by SA-tree and HRG. Thicker edges denote visited edges. A fewer edges are visited in HRG than in SA-tree.

Algorithm 1. RangeSearch(Object x , Range r) // x : query object, r : query radius

```

1: result :=  $\emptyset$ ;
2:  $a$  := random vertex in  $G_H$ ;
3:  $min\_dist$  :=  $\infty$ ;
4: RangeSearch( $a$ ,  $x$ ,  $r$ ,  $min\_dist$ , result);
5: return result;
```

3 Experimental Results

We tested the proposed HRG in terms of similarity search operations using two real and two synthetic datasets.

1. HSV-3: This is a real dataset consisted of 15,000 3-dimensional objects obtained from a color image by sampling the hue (H), saturation (S), and value (V) of every three other pixel.
2. Faces-175: We generated this real dataset consisted of 600 175-dimensional objects from 600 face images found in the CMU Face Database¹. Each face figure is divided into $5 \times 5 = 25$ regions and the following seven features are extracted from each region: the average and the standard deviation for each of H, S and V, and the percentage of the presence of skin color in a region, which yield 600 feature vectors of $7 \times 5 \times 5 = 175$ dimensions.
3. Synthetic-2: This synthetic dataset contains 1,000 2-dimensional vectors normally distributed in 10 clusters with overall standard deviation of 0.1 over a rectangle with a minimum and maximum value of 0 and 1 on each dimension respectively.
4. Synthetic-16: This synthetic dataset contains 1,500 16-dimensional vectors normally distributed in 10 clusters with overall standard deviation of 0.1 over a hypercube with a minimum and maximum value of 0 and 1 on each dimension respectively.

¹ <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-8/ml94faces/faces>

Algorithm 2. RangeSearch (Vertex a , Object x , Range r , Distance min_dist , Stack S)

```

1: if  $a$  has not been visited yet then
2:   if  $d(a, x) - r \leq radiusRegion(a) \vee d(a, x) + r \leq radiusRegion(a)$  then
3:     for each object  $o$  in  $H(a)$  do
4:       if  $|d(a, x) - d(o, a)| \leq r$  then
5:         if  $calculateDistance(o, x) \leq r$  then
6:            $result := result \cup \{o\}$ ;
7:         end if
8:       end if
9:     end for
10:  end if
11:   $min\_dist := \min\{\{min\_dist\} \cup \{d(b, x) : b \in N(a)\}\}$ ;
12:  for  $b \in N(a)$  do
13:    if  $d(b, x) \leq min\_dist + 2r$  then
14:      mark  $b$  as VISITED;
15:      RangeSearch( $b, x, r, min\_dist, result$ );
16:    end if
17:  end for
18: end if

```

The performance of HRG was then compared to those of the two most well known tree structures M-tree [1] and SA-tree[4], and a graph data structure Incremental-RNG [5] in terms of the number of distance computations and total response time for similarity search and data structure construction². The Euclidean distance (L_2) was used as the distance function in our experiments. Furthermore, for similarity search, we used the range query to make our results comparable to others since the experimental results of other approaches found in the literature are based on this type of search. All the algorithms were implemented in Java 1.6 and tested on a 2.0 GHz PC with 2038 MB of RAM running 32-bit Microsoft Windows Vista.

Similarity Search Performance

NDC In this test, the data structures in comparison were tested with different range search radii, ranging from 10% to 100% of the maximum distance³ between the two farthest objects in the dataset. HRG performed better than all other structures with Synthetic-2, Synthetic-16 and HSV-3, as shown in Figures 2(a), 2(b) and 2(c). With Faces-175, HRG outperformed Incremental-RNG with a comparable performance with M-tree (see Figure 2(d)). In case of Faces-175, potentially high overlap among the regions in HGR offsets the anticipated benefit of HRG. In case of such high dimensional data, we observed that SA-tree is the best out of the structures in comparison.

² Due to the page limit, the detailed figures on data structure construction times are omitted, but available by request.

³ This distance is referred as the dataset length in Figures 2 and 3.

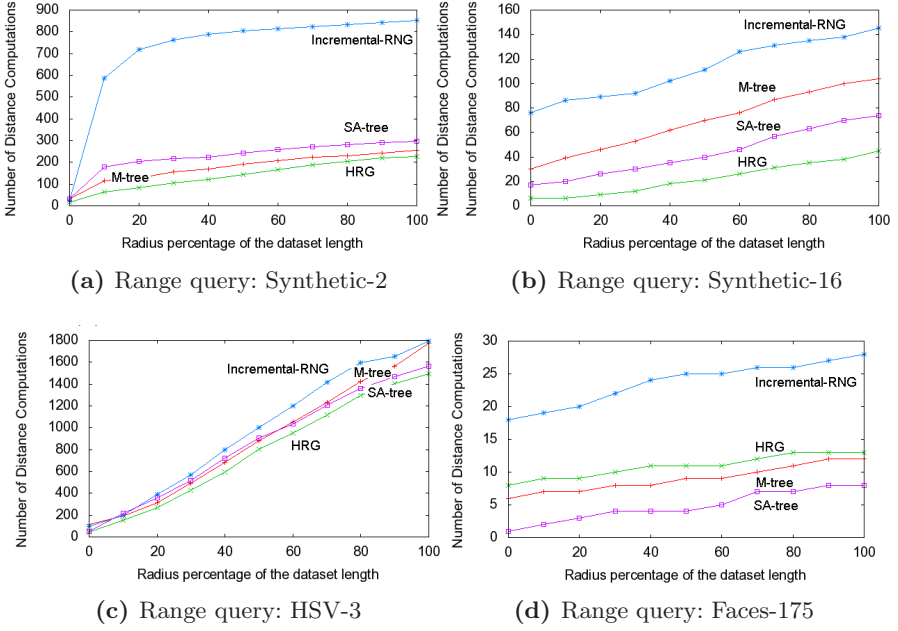


Fig. 2. Comparison of HRG by number of distance computations in range search. Note that the scale of y -axis is different from one figure to another.

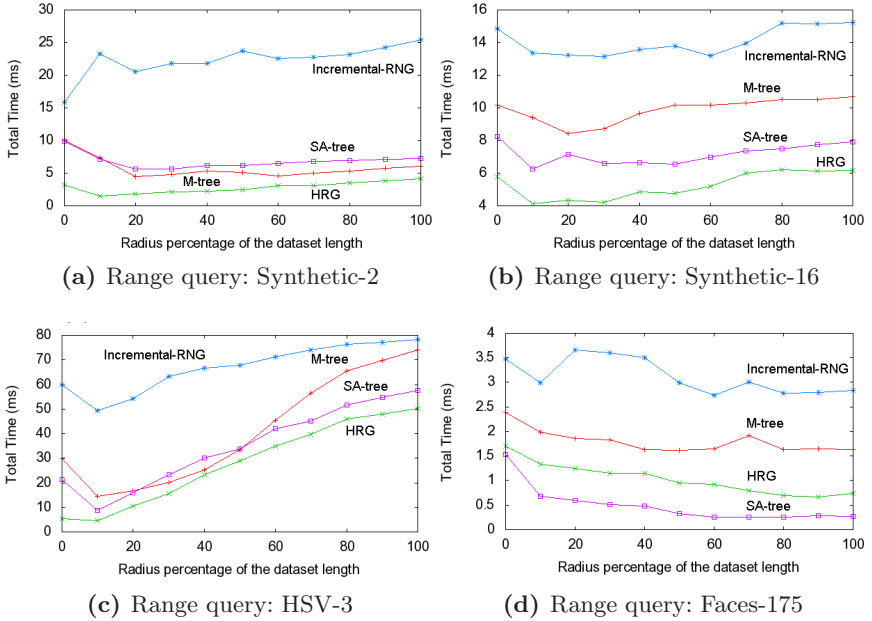


Fig. 3. Comparison of HRG by total range query evaluation time at various radii

Response Time. This test was conducted in a similar way to the above to measure the total evaluation time of range searches. All the test dataset was fit into the main memory. The performance of HRG in this test was congruent with the result of NDC, which was anticipated because the main cost during running time is the computation of distances between objects, as the result is shown in Figure 3.

4 Conclusions and Future Work

This paper introduced a graph-based indexing algorithm and discussed its performance for the similarity search problem in metric spaces. The proposed algorithm, which uses a small representation of the search space, can be classified as a compact partitioning algorithm. The notion of neighborhood present in the spatial approximation search technique [4] is fully exploited in the graph. Moreover, since each vertex is the center of a hyperspherical region, our approach only visits the neighbors which are close enough to the query object, thus saving the evaluation of distances between the query object and other objects.

The experimental results showed that the proposed method performs better than other tree and graph-based approaches in similarity search with datasets of moderate dimensionality. In case of a high dimensional dataset, we observed overlap between regions, which offsets the benefits of the proposed structure.

References

1. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Proceedings of the 23rd International Conference on Very Large Data Bases, San Francisco, CA, USA, pp. 426–435 (1997)
2. Traina, C., Traina, A., Seeger, B., Faloutsos, C.: Slim-trees: High performance metric trees minimizing overlap between nodes. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 51–65. Springer, Heidelberg (2000)
3. Vieira, M.R., Traina Jr., C., Chino, F.J.T., Traina, A.J.M.: Dbm-tree: A dynamic metric access method sensitive to local density data. In: Brazilian Symposium on Databases, pp. 163–177 (2004)
4. Navarro, G.: Searching in metric spaces by spatial approximation. The VLDB Journal 11(1), 28–46 (2002)
5. Hacid, H., Yoshida, T.: Incremental neighborhood graphs construction for multidimensional databases indexing. In: Advances in Artificial Intelligence, pp. 405–416 (2007)
6. Hacid, H., Zighed, A.D.: An effective method for locally neighborhood graphs updating. In: Database and Expert Systems Applications, pp. 930–939 (2005)
7. Zhao, D., Yang, L.: Incremental construction of neighborhood graphs for nonlinear dimensionality reduction. In: Proceedings of the 18th International Conference on Pattern Recognition, pp. 177–180 (2006)
8. Lee, C., Kim, D., Shin, H., Kim, D.-S.: Efficient computation of elliptic gabriel graph. In: International Conference on Computational Science and Applications, pp. 440–448 (2006)
9. Jaromczyk, J., Toussaint, G.: Relative neighborhood graphs and their relatives. In: Proceedings of IEEE, vol. 80, pp. 1502–1517 (1992)

Word Sense Disambiguation as the Primary Step of Ontology Integration

Marko Banek¹, Boris Vrdoljak¹, and A Min Tjoa²

¹ Faculty of Electrical Engineering and Computing, University of Zagreb,
Unska 3, HR-10000 Zagreb, Croatia
{marko.banek, boris.vrdoljak}@fer.hr

² Institute of Software Technology and Interactive Systems,
Vienna University of Technology, Favoritenstr. 9-11/188, A-1040 Wien, Austria
amin@ifs.tuwien.ac.at

Abstract. The recommendable primary step of ontology integration is annotation of ontology components with entries from WordNet or other dictionary sources in order to disambiguate their meaning. This paper presents an approach to automatically disambiguating the meaning of OWL ontology classes by providing sense annotation from WordNet. A class name is disambiguated using the names of the related classes, by comparing the taxonomy of the ontology with the portions of the WordNet taxonomy corresponding to all possible meanings of the class. The equivalence of the taxonomies is expressed by a probability function called affinity function. We apply two different basic techniques to compute the affinity coefficients: one based on semantic similarity calculation and the other on analyzing overlaps between word definitions and hyponyms. A software prototype is provided to evaluate the approach, as well as to determine which of the two disambiguation techniques produces better results.

Keywords: ontology integration, OWL, word sense disambiguation, WordNet, semantic similarity.

1 Introduction

The scientific community has recognized ontologies as a means of establishing an explicit formal vocabulary to be shared between applications. The Web Ontology Language (OWL) [13], a W3C Recommendation, has been accepted as a standard for writing ontologies. The fact that the ontologies applied by collaborating applications have been developed independently implies their possible inconsistency, thus shadowing the primary goal of their development, i.e. a common, standard source of knowledge that overcomes heterogeneity.

Ontology mismatches mostly appear when the same linguistic term describes different concepts or when different terms describe the same concept (semantic mismatches). Hence, it is necessary to disambiguate the meaning of ontology components, either by applying heuristic formulas and machine learning techniques to analyze the entire ontology content [4, 12] or by annotating the components with natural language descriptions or entries from a lexical database like WordNet that impose a clear, single meaning [3, 8, 11]. The first approach requires learning each time a

specified ontology is merged with another one. On the other hand, semantic annotations employed in the second approach can be further reused without additional effort, which is highly recommendable in cases when merging needs to be performed rather frequently. Techniques that follow the latter approach define ontology annotation as the first step of the integration process.

This paper presents an approach to automatically disambiguating the meaning of OWL ontology classes as the first step of ontology integration. The contribution of our work is to apply the names of the related ontology components to disambiguate the meaning of the target component, taking into account the degree of their relatedness. The complex process of disambiguating ontology components is performed by using two different basic disambiguation techniques: one based on semantic similarity calculation and the other based on the analysis of overlaps between word definitions and hyponyms. A Java-based prototype software tool is provided to evaluate our approach, as well as to determine which of the two basic disambiguation techniques produces a better result.

The paper is structured as follows. Section 2 gives an overview of the related work. The algorithm for disambiguating the sense of ontology classes and the two different applied techniques are described in Section 3. Section 4 presents experiments for the evaluation of the presented algorithm. Conclusions are drawn in Section 5.

2 Related Work

Most ontology integration methods are of heuristic nature or machine learning-based [5]. They produce mappings between the ontology components based on the analysis of names, natural language definitions and the ontology structure. Since natural language definitions of ontology components are not provided in a large majority of cases, the correctness of name matching depends on the lexical similarity calculation techniques (e.g. in [6]), which generally apply either the most frequent meaning, or the one whose meaning is most similar to the term being compared with.

Annotation of ontology components with WordNet senses [7] or some other meaning disambiguation entries has been defined as the first step of the integration process by many authors. However, in all those works the attention has been paid to the later steps, while annotations were either created manually [11] or considered to be provided earlier [3, 8]. In other cases the ontology components were simply supposed to have a determined unambiguous meaning [2]. To the best of our knowledge, no existing ontology integration technique performs the annotation process i.e. class sense disambiguation automatically.

An ontology mapping approach based on graph theory and described in [8] requires the source ontologies to be annotated with WordNet senses. Multiple-word component names are defined by combining WordNet senses (regarded as atomic particles) using set operators. However, the process is described from the point when the particular sense of the target word has already been determined.

The earliest disambiguation approaches [1] were based on analyzing text corpora. While supervised approaches applied machine learning, the unsupervised ones automatically translated a corpus in the target language to another language and registered cases when the same term in the target language was translated to different terms in

the other language. Recent disambiguation approaches (e.g. [9]) exploit dictionary entries of the target words as the only source data.

3 Disambiguating OWL Ontology Classes

The process of disambiguating the meaning of an ontology component is based on analyzing the related components. There are three basic component types in OWL ontologies: classes, instances and properties [13]. The central part of ontologies is given by the classes, which conform to the concept of nouns in human thought and reveal the taxonomic structure of ontologies (classes can form taxonomies by specifying subsumptions, either directly, or using complex OWL intersection and union expressions). Object properties display an attribute-like relation between nouns.

Since finding compatible classes is the core of ontology integration, we entirely focus the disambiguation process on classes. Ontology classes will be associated with a WordNet sense whenever such a possibility exists. We use WordNet due to its omnipresence in the existing ontology alignment techniques [3, 8, 11].

3.1 Ontology Class Neighborhood and Class Sense Disambiguation

WordNet divides the searchable lexicon into four categories: nouns, verbs, adjectives and adverbs. Each input word can have more than one meaning, which is also called word sense. For instance, there are two noun senses of the entry *student*: (1) *a learner who is enrolled in an educational institution*, (2) *a learned person (especially in the humanities)*. A word sense can be described by one or more synonyms and is called a synset (e.g. the first meaning of *student* has synonyms *pupil* and *educatee*, while the synonyms *scholar* and *scholarly person* conform to the second one). Each synset is given a description sentence called gloss and may have antonyms. A WordNet entry can either be a single word (*scholar*) or consist of several words (*scholarly person*).

WordNet includes a set of semantic relations for each word category. The largest spectrum of relations exists for nouns, which comprise about 80% of all WordNet entries [7]. Hyponymy/hypernymy is a transitive relation between nouns that represents subsumption and exactly conforms to the concept of subclasses/superclasses in ontologies. All WordNet nouns are arranged in a single taxonomy consisting only of the hyponymy/hypernymy relation. The part-whole relation is called meronymy/holonymy. A portion of the WordNet taxonomy containing the two senses of *student* can be seen in Fig. 1.

We associate a class name (always supposed to be noun), with a particular WordNet noun synset. Some classes will receive no annotation since there will be no appropriate WordNet noun corresponding to their names, particularly if the names consist of more than one word (e.g. *PhDStudent* in a university ontology). An annotation will be created without applying the disambiguation process if the corresponding WordNet noun term has a single sense (e.g. *Professor*). “Upper” i.e. more “general” classes in ontology taxonomies are likely to have many senses (e.g. *Student*, *Course*, *Graduate* in a university ontology).

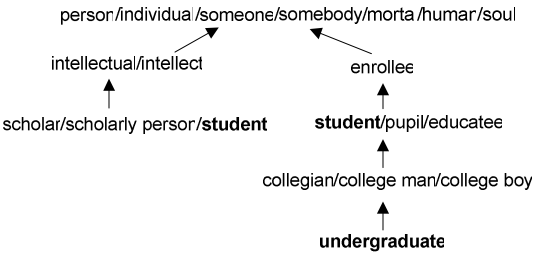


Fig. 1. Portion of the WordNet taxonomy describing synsets *student* and *undergraduate*

We start the OWL ontology class disambiguation process by translating the complex intersection and union constructs to subsumption relations, leaving subsumptions and object properties as the only two edge types in the final taxonomy graph of the ontology. The taxonomy of a university ontology fragment is shown in Fig. 2. Subsumption edges are represented by full lines, always pointing from the subclass to the superclass. Object property edges are represented by dashed lines, pointing from the domain to the range class.

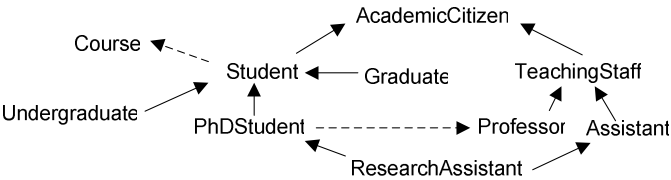


Fig. 2. Basic taxonomic relations of the university ontology

Similarly to the noun taxonomy of WordNet, a specified ontology class is “surrounded” by classes that constitute its taxonomy: its subclasses, superclasses and property range classes, which, on their part, have their own subclasses, superclasses and property ranges. We call the part of the ontology taxonomy with the specified class as its central point the *neighborhood* of that class. We disambiguate the target ontology class by comparing its neighborhood taxonomy with WordNet taxonomies of all noun synsets corresponding to the class name. The WordNet noun synset with the taxonomy “most similar” to the target ontology class neighborhood will be stated as the real meaning of the target class.

We define the *distance* between two classes in the ontology neighborhood as the number of edges that form the shortest valid path between them. The neighborhood of an ontology class of radius r consists of all classes whose distance from that class is smaller than or equal to r . The neighborhood of radius 1 of a target class C includes the following links: (1) all its direct subclasses, (2) all its direct superclasses, (3) all the ranges of its own (i.e. not inherited) properties, (4) all classes whose property range is the target class. Class *Student* in the university taxonomy has five classes in its neighborhood of size 1: *Undergraduate*, *Graduate*, *PhDStudent*, *AcademicCitizen* and *Course* (see Fig. 2). When creating the neighborhood of radius 2 or larger, we

eliminate paths going both “up” and “down” and paths containing more than one property edge, since they mostly reach classes unrelated or even disjoint with the target class (e.g. *Professor* – *TeachingStaff* – *Assistant*). Considering the target class *Student*, the neighborhood of radius 2 will also include classes *Professor* and *ResearchAssistant*, but not *TeachingStaff*.

We define the probability that a sense C_i is the true meaning of an ontology class C due to the fact that C is related to another class D in the same ontology as the *affinity* between C_i and D , $a(C_i, D) \in [0,1]$. The highest of the affinity coefficients with D across all senses of C implies the true meaning of C from the perspective of D . Considering that the affinity between C_i , the i -th sense of the target class C , and a class D_{Xj} at distance X from C_i is expressed by some coefficient $a(C_i, D_{Xj})$, the total affinity value (i.e. the one taking into account all classes from the neighborhood) of C_i is calculated as *weighted mean*:

$$a_{total}(C_i) = \frac{w_1 \cdot \sum_{j=1}^{|R_1|} a(C_i, D_{1j}) + w_2 \cdot \sum_{j=1}^{|R_2|} a(C_i, D_{2j}) + \dots + w_N \cdot \sum_{j=1}^{|R_N|} a(C_i, D_{Nj})}{w_1 \cdot |R_1| + w_2 \cdot |R_2| + \dots + w_N \cdot |R_N|}, \quad (1)$$

where N is the radius of the neighborhood, $|R_X|$ the total number of neighborhood classes at distance X from C and w_X are weights corresponding to each distance. In our experiments we adopt two interpretations of the distance. In the first case, we ignore the distance and assume that all neighborhood classes equally contribute to the final score ($w_X = 1$, $1 \leq X \leq N$). In the second case, we assume that the influence of a class is inversely proportional to the square of its distance (i.e. $w_X = 1/X^2$, $1 \leq X \leq N$).

Since classes at a very large distance from the target class are likely to represent some unrelated concept [14] and are not expected to contribute much to the target class disambiguation, we believe that the radius should not be larger than 5.

The affinity between the i -th sense of the target class C and a neighborhood class D will be calculated using two different approaches presented in Section 3.2

3.2 Word Sense Disambiguation Techniques

The first disambiguation technique we apply expresses the affinity coefficient between two ontology classes as the value of semantic similarity function between the equivalent WordNet synsets. Among the existing techniques for measuring semantic similarity in thesauri we adopt the most recent one, presented by Yang and Powers [14], which is also reported to be the most efficient. The range of the semantic similarity function is the interval $[0,1]$. The WordNet noun taxonomy is interpreted as a graph, with synsets as vertices and hyponymy/hypernymy and meronymy/holonymy relations as edges. Each edge is given a weight β ($0 < \beta < 1$) according to the relation type. All possible paths between the target vertex and all vertices corresponding to different senses of the other word are constructed. Weights are multiplied across paths and the highest product (which also denotes the shortest path) becomes the value of semantic similarity between the target synsets.

In order to disambiguate a target class using a class from its neighborhood, we construct all valid paths between every WordNet synset corresponding to the target class and each of the N synsets corresponding to the neighborhood class (the sense of

the neighborhood class is also unknown). Following the standard solution outlined in [14], the presumed true sense of the neighborhood class is the one with its WordNet representative at the shortest distance from the representative of the particular sense of the target class. Considering the neighborhood class *Undergraduate* (Fig. 2), the affinity for *scholar/scholarly person/student* is determined by a six-edge path to the only sense of *undergraduate* (see Fig. 1.). On the other hand, there is a two-edge path and a much higher affinity value between *student/pupil/educate* and *undergraduate*.

While the target class must correspond to a WordNet noun entry, the neighborhood class used to disambiguate the target class may be any complex combination of words including even abbreviations. The contribution of each word participating in the complex class name is calculated separately, using the formula given in [10].

The second disambiguation method is a “standard” disambiguation technique as introduced by Liu, Yu and Meng [9]. Word senses are disambiguated by finding the overlap among their definitions (i.e. glosses). For instance, *computer* disambiguates *terminal*, as it can only be found in the gloss of its third sense (*electronic equipment consisting of a device providing access to a computer; has a keyboard and display*).

For each two target words w_1 and w_2 the following four sets are extracted for each of the senses: (1) all synonyms, (2) all words from the gloss, (3) synonyms of all hyponyms, (4) all words from all the hyponyms’ glosses. 4×4 pair-wise comparisons of the sets are performed, which is reduced to 10 independent cases due to symmetry (we implement 10 different comparator functions). The existence of an overlap between any of the sets belonging to the senses s_1 and s_2 of words w_1 and w_2 , respectively, suggests a correlation between those two senses (i.e. synsets). If no overlap exists for other pairs of senses of w_1 and w_2 or if the size of the latter is smaller (we empirically determine the “winning” overlap to be at least 1.2 times bigger than all others), the disambiguation is successful.

4 Experiments

A Java prototype tool that implements both disambiguation techniques has been developed in order to test our approach on real-life ontologies as well as to determine which of the two disambiguation techniques works better. The experiments included three target ontologies: the Wine and Food Ontology from [13], the Infectious Disease Ontology [15] and the Computer Ontology developed at Stanford University [16].

Considering the optimal size of the target class neighborhood, we performed experiments with four different neighborhood radii: 1, 2, 3 and 5. The affinity is computed using both the distance-related weights and equal weights for all distances.

Following our notion of overlap comparison threshold (Section 3.2), we state that an ontology class C is disambiguated only if the highest affinity value (associated with a sense s of C) is at least *thr* times bigger than the second highest. We perform the experiments for the following values of *thr* : 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7.

The statistics refer only to classes that can be associated with a WordNet noun with at least two senses. We define *recall R* as the ratio between the number of ontology classes that could successfully be disambiguated (with respect to *thr*) and the total number of classes corresponding to a WordNet term with at least two senses. We compare the results of the automatic disambiguation with those obtained by human

judgment and define *precision P* as the ratio of the number of correctly disambiguated classes and the number of classes that could be successfully disambiguated. An increase of *thr* is expected to produce a higher precision but a lower recall.

For each existing combination of radius size, weight and disambiguation technique we calculate the average value of recall and precision across the three target ontologies. The percentage of ontology classes that conform to a WordNet noun entry is 7 of 20 (35%) for the Wine and Food Ontology, 90 of 150 (60%) for the Disease Ontology and 19 of 44 (43%) for the Computer Ontology.

Due to a lack of space we do not provide statistics for all threshold values. Instead, in Table 1 we give the recall and precision values for all different combinations with the threshold set to its optimal value 1.1. A rise of *thr* contributes more to a reduction of recall than to an increase of precision (the impact is measured by comparing the geometric mean of recall and precision), thus producing worse results in total.

The disambiguation technique based on semantic similarity calculation works better than the one based on analyzing glosses and hyponyms, achieving precision higher than 90% while maintaining recall almost at 85% (the gloss-based technique gives a slightly better recall, but the reduction of precision is more significant). Calculations produced over a larger neighborhood (radii 3 and 5) give better results than those made over a smaller one (although not as significantly as expected). Using different weight systems did not create any significant impact. Thus, we propose the adoption the simpler system ($w=1$, regardless of the distance).

As a conclusion to the experiments, we suggest using the disambiguation technique based on semantic similarity calculation with neighborhood radius between 3 and 5.

Table 1. Entire results of the experiments with the threshold set to its optimal value

		radius = 1		radius = 2		radius = 3		radius = 5	
		R	P	R	P	R	P	R	P
Yang & Powers	w=1	0.818	0.905	0.847	0.905	0.847	0.905	0.847	0.905
	w=1/D ²	0.818	0.905	0.832	0.905	0.832	0.905	0.832	0.905
Liu, Yu & Meng	w=1	0.886	0.781	0.847	0.801	0.915	0.742	0.915	0.742
	w=1/D ²	0.886	0.780	0.886	0.780	0.915	0.759	0.915	0.759

5 Conclusion

This paper presents an approach to automatically disambiguating the meaning of OWL ontology classes in order to produce sense annotations from WordNet, which is the first step of the ontology integration process. The approach is based on analyzing names of ontology classes most related to a disambiguation target class. The taxonomy of the target class neighborhood is compared with taxonomies of all WordNet noun synsets corresponding to the target class name. Their equivalence is then expressed by an affinity coefficient. The approach was tested on three example ontologies using the developed software prototype. Analyzing the results of the experiments, the disambiguation technique based on measuring semantic similarity between WordNet nouns emerged as the optimal solution for calculating affinity coefficient values.

References

1. Agirre, E., Edmonds, P. (eds.): Word Sense Disambiguation - Algorithms and Applications. Springer, New York (2006)
2. Aum Mueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: Özcan, F. (ed.) Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 906–908. ACM Press, New York (2005)
3. Castano, S., Ferrara, A., Montanelli, S.: H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In: Cruz, I.F., Kashyap, V., Decker, S., Eckstein, R. (eds.) Proc. 1st Int. Wshp on Sem. Web and Databases, pp. 231–250 (2003)
4. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to Map between Ontologies on the Semantic Web. In: Proc. 11th Int. WWW Conf., pp. 662–673. ACM, New York (2002)
5. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, New York (2007)
6. Euzenat, J., Valtchev, P.: Similarity-Based Ontology Alignment in OWL-Lite. In: López de Mántaras, R., Saitta, L. (eds.) Proc. 16th European Conf. on Artificial Intelligence, pp. 333–337. IOS Press, Amsterdam (2004)
7. Fellbaum, C. (ed.): WordNet. An Electronic Lexical Database. MIT Press, Cambridge (1998)
8. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an Algorithm and an Implementation of Semantic Matching. In: Bussler, C., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 61–75. Springer, Heidelberg (2004)
9. Liu, S., Yu, C.T., Meng, W.: Word Sense Disambiguation in Queries. In: Herzog, O., et al. (eds.) Proc. 2005 ACM Int. Conf. on Information and Knowledge Management, pp. 525–532. ACM Press, New York (2005)
10. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic Schema Matching with Cupid. In: Apers, P.M.G., et al. (eds.) Proc. 27th Int. Conf. Very Large Data Bases, pp. 49–58. Morgan Kaufmann, San Francisco (2001)
11. Patel, C.O., Supekar, K., Lee, Y.: OntoGenie: Extracting Ontology Instances from WWW. In: Proc ISWC 2003 Workshop on Human Language Technology for the Semantic Web and Web Services, pp. 123–126 (2003)
12. Udrea, O., Getoor, L., Miller, R.J.: Leveraging Data and Structure in Ontology Integration. In: Chan, C.Y., Ooi, B.C., Zhou, A. (eds.) Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 449–460. ACM Press, New York (2007)
13. World Wide Web Consortium: OWL Web Ontology Language Guide (W3C Recommendation, February 10, 2004), <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>
14. Yang, D., Powers, D.M.W.: Measuring Semantic Similarity in the Taxonomy of WordNet. In: Estivill-Castro, V. (ed.) Proc. 28th Australian Computer Science Conference, pp. 315–322. Australian Computer Society (2005)
15. Infectious Disease Ontology, http://www.duke.edu/~lgcowell/IDO_Workshop_Files/IDO.9.19.07.owl
16. Stanford University Computer Ontology, <http://ksl.stanford.edu/DAML/computers.owl>

Enriching Ontology for Deep Web Search

Yoo Jung An¹, Soon Ae Chun², Kuo-chuan Huang¹, and James Geller¹

¹New Jersey Institute of Technology

²CSI, City University of New York

ya8@njit.edu, chun@mail.csi.cuny.edu, kh8@njit.edu,
geller@njit.edu

Abstract. This paper addresses the problems of extracting instances from the Deep Web, enriching a domain specific ontology with those instances, and using this ontology to improve Web search. Extending an existing ontology with a large number of instances extracted from the Deep Web is an important process for making the ontology more usable for indexing of Deep Web sites. We demonstrate how instances extracted from the Deep Web are used to enhance a domain ontology. We show the contribution of the enriched ontology to Web search effectiveness. This is done by comparing the number of relevant Web sites returned by a search engine with a user's search terms only, with the Web sites found when using additional ontology-based search terms. Experiments suggest that the *ontology plus instances* approach results in more relevant Web sites among the first 100 hits.

Keywords: Deep Web, Semantic Web, Instance Extraction, Domain Ontology.

1 Introduction

The term *Deep Web* has been coined to refer to Web pages dynamically generated via query interfaces, implemented as Web forms or Web services. Due to its dynamic nature, existing Web crawlers cannot access the Deep Web [1]. Thus, accessing and maintaining the huge amount of Deep Web information remain challenging research issues [2]. In [3], information in Deep Web sites was categorized as being either in *textual* or *structured databases*. While a textual database needs input keywords for searching text documents, a structured database requires a user to fill in input fields of a query interface. This paper focuses on structured databases. It addresses the problem of how to extract data from the Deep Web to automatically enrich an ontology that can support better search engines.

The purpose of the Semantic Web (SW) is to automate tasks that humans commonly perform on the WWW, using their intelligence. The SW depends heavily on ontologies, each of which is a computer implementation of human-like knowledge [4,5]. The software agents operating on the SW will need some human-like knowledge to perform their tasks. Thus, the success of the SW critically depends upon the existence of a sufficient amount of high-quality semantics contained in ontologies [6]. In our previous work [7], the *Semantic Deep Web* was introduced as a framework that combines aspects of the Deep Web and the SW. (Note that this is not the same as the *Deep Semantic Web*.) Because of ontologies' promise of sharing knowledge, they will

play a dominant role on the SW [8] and, presumably, the Deep Web. In [7, 10], the search interface elements of Deep Web sites have been extracted to automatically build domain ontologies.

In this paper, we extract *instances* from the Deep Web (DW) to populate the ontology. The enriched ontology is then used during Web search to improve the search results for DW sites. The search terms entered by a user, augmented with domain ontology instances, are expected to better describe the user's interests. We claim that the proposed method assists users with finding more relevant Web sites. A domain ontology-based Web search module was built. An assessment was conducted by comparing the number of relevant DW sites returned by a search engine that used a user's search terms only, with the results returned by using search terms extended with ontology instances.

2 Related Work

OntoMiner [5] finds URLs on a partitioned segment of a Web page (e.g., of a hotel) which are linked to instances. From each segment, labels which are concepts and their sub-labels with corresponding values are extracted as instances. A result page after a search, called data page, is a source for extracting instances in DeepMiner [9]. In our research, we use concepts in an ontology which was automatically generated from the DW in [10], to extract instances for these concepts. The significant difference of our approach, compared with [5, 6, 9] is that we rely on extracting instances by dynamically probing backend databases of a DW site. In contrast to most ontology learning work, which focuses on Web documents, as in [6, 15, 16], the focus in our research is on the DW.

Research [19, 20] on the DW has focused on Web forms, as they are entry points to access the DW. In [19], the Hidden Web Exposer (HiWE) can automatically assign values to the form, referencing a "Label Value Set (LVS)" that is initially a human guided description for the HiWE. Form filling and submission are also research problems in [20]. In our previous work [7], automatic extraction of *attributes* of DW sites was conducted at the schema level of HTML forms, which is comparable to [19]. Ontology based search engines have been developed and equipped with specific features in [21].

3 Enriching a Domain Ontology for the Semantic Deep Web

Enrichment of an ontology is a process that extends it by adding concepts, instances and new relations between concepts [17]. Our previous work [10] dealt with the schema level while this paper deals with the data level, as we utilize instances extracted from DW result pages. While the schema level extraction finds concepts such as 'city name,' etc., the data level extraction results in instances such as 'Newark.' Our method for extracting instances from the DW is based on developing "robots" (agents) that send many queries to the same DW site to extract as many data values as possible. When a robot encounters an input field it may enter random values or leave the field empty and then submit the page to elicit an informative response. Figure 1

shows the workflow for extending the ontology with instances. The concept discovery of the robot is guided by a human in its initial stage. Initial pairs of a concept and its corresponding instances are defined, which we call a *robot image*. The robot submits input values into the query interface. If the input values are not suitable for the form, most Web sites display error messages. The analysis of the error messages often gives useful clues to the robot to guess suitable input values and launch better probing queries. Thus, the queried Web sources may provide information about concepts, instances and semantic relationships, which is recorded in the ontology.

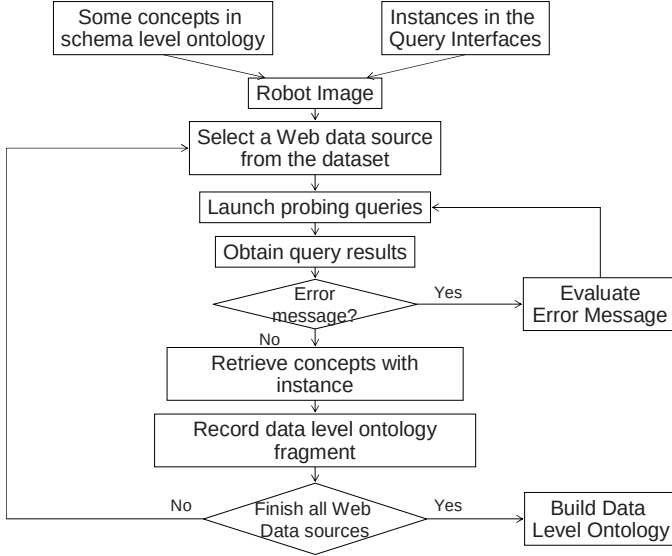


Fig. 1. A flow for generating data level ontology fragments

Consider a DW site, such as a flight reservation system. Our program generates a list of candidate airport codes from AAA to ZZZ with the assumption that airport codes consist of three letters. The probing program submits the form with a *candidate airport code* (Figure 2), gets a result page and parses it. Next, the program extracts available instances (e.g., airport name, etc.) from the result page (Figure 3). If the candidate airport exists, it will be in the result page. Otherwise, an error or a “similar” airport will be returned. The algorithm for crawling instances from a DW site is omitted due to space limitations [22]. Table 1 shows data extracted from the DW. Concepts in the first column and their corresponding concepts in the third column are in relationships named in the second column. The number of extracted relationships is in the fourth column.

The extracted instances are loaded into domain ontologies that were defined and represented in OWL, e.g., the air travel domain ontology in [10]. In order to load the DW instances into the domain ontology, a program was implemented in [22] using the Protégé API [11, 12, 18]. For example, for John F. Kennedy International airport, the property *hasAirportCode* is JFK, and it *isAirportOf* New York.

City/Airport lookup

By city name

By airport name

City/Airport lookup tool

To find an airport, please enter the city name, airport name or airport code.

From:

Region:

North America

Search site >>>

Select

City/Airport code

City/Airport name

City

State/Country

Select >>>

Fig. 2. A sample Web site with a dynamic query interface

City/Airport lookup

By city name

By airport name

City/Airport lookup tool

To find an airport, please enter the city name, airport name or airport code.

From:

Region:

North America

Search site >>>

Select

City/Airport code

City/Airport name

City

State/Country

☒

EWR

Newark Liberty International Airport

Newark

NJ / United States

☐

JFK

John F. Kennedy International Airport

New York

NY / United States

☐

LGA

La Guardia International Airport

New York

NY / United States

☐

HPN

White Plains-Westchester County Airport

White Plains-Westchester County

NY / United States

Select >>>

Fig. 3. A sample Web site with results

4 Web Search with Domain Ontology-Based Query Extension

In this research, a domain ontology-based Web search module was implemented [22]. For example, if a user clicks on assertions related to airport codes or airports, the *search module* will create an extended list of key words (i.e., “New York”, NYC, “Seoul”, SEL). The assumption that a user inputs only “New York” and “Seoul” to search for a flight which operates from “New York” to “Seoul” relies on [13], where it was reported that a user, on average, enters 2.1 terms for a Web search. Thus, we assume that a user who wants to flight from New York to Seoul may enter these two terms.

The algorithm for extending the query terms of a user can be found in [22]. It processes a user query string and separates it into phrases. Next, if a phrase is an instance of the domain ontology, extract its properties. Properties refer to relationships and objects of the properties refer to instances of the relationships in OWL. For example, New York has properties such as *isCityOf*, *hasAirport*, *hasAirportCode*, etc., and objects of the properties are New York, John F. Kennedy International Airport, and

Table 1. Instances of Extracted Relationships

Class Name	Relationship	Class Name	No. of Instances
city	<i>hasAirport</i>	Airport	1090
city	<i>isCityOf</i>	country	997
city	<i>isCityOf</i>	province	362
airport_code	<i>isAirportCodeOf</i>	airport	1090
country	<i>hasCity</i>	city	997
province	<i>hasCity</i>	city	362
country	<i>hasState</i>	province	61
province	<i>isStateOf</i>	country	61
airport	<i>hasAirportCode</i>	airport_code	1090
airport	<i>isAirportOf</i>	city	1090
airport code	<i>nearBy</i>	airport code	102
airport	<i>nearBy</i>	airport	102
airport	<i>sameAs</i>	airport_code	1090

Search with Ontology

Term

We can search more details about your terms, please check the term(s) below:

- ☐ seoul isCityOf south korea
- ☐ seoul hasAirport seoul
- ☐ seoul hasAirport seoul incheon international airport
- ☐ seoul hasAirport seoul gimpo international airport
- ☐ seoul isAirportOf seoul
- ☐ seoul hasAirportCode sel
- ☐ new york isCityOf ny
- ☐ new york isCityOf united states
- ☐ new york hasAirport new york
- ☐ new york hasAirport la guardia international airport
- ☐ new york hasAirport john f. kennedy international airport
- ☐ new york isAirportOf new york
- ☐ new york hasAirportCode nyc

Fig. 4. User feedback interface

NYC, respectively. These are appended to the user query (Q), and the extended query (EQ) is resubmitted.

Using the user terms $Q = \{\text{New York, Seoul}\}$ only resulted in a display with few flight reservation Web sites. Then our system extended Q to $EQ = \{\text{New York, John F. Kennedy International Airport, Seoul, Seoul Incheon International Airport}\}$ with respect to the user selections of semantics in Figure 4. Details of the evaluation process can be found in [22]. Table 2 gives a flavor of the results. The extended search string EQ results in several orders of magnitude fewer sites returned by Google. More importantly, looking at the first 100 returned sites only, the number of relevant sites found with EQ is always greater or equal to the number of sites found with Q .

Table 2. Case study results (small subset)

<i>Q</i>	<i>EQ</i>	Sites for Q (Millions)	Sites for EQ	Relev. Sites* in top 100 for Q	Relev. sites in top 100 for EQ
New York, Seoul	New York, John F. Kennedy International Airport, Seoul, Seoul Incheon International Airport	4.77	267	9	29
New York, Tokyo	New York, John F. Kennedy International Airport, Tokyo, Narita International Airport	118.00	503	1	8
* Relevant. sites are the Web pages which contain entry points to Deep Web sites relevant to the flight reservation domain.					

5 Conclusions, Future Work and Discussion

In this paper, we first presented a novel approach to enhancing a domain ontology by adding a large number of instances extracted from the DW to it. This process was semi-automatic. The program crawled information from the DW site (<http://www.united.com/>) but was not intelligent enough to automatically define semantic relationships between classes. Humans defined them based on the structure of a result page. In all, 3,385 instances of concepts and 8,494 instances of semantic relationships were extracted. Secondly, a prototype Web search module was implemented. We evaluated the success of searching for airline Web sites, using the extended domain ontology. Our experiments suggest that Web search results of a general purpose search were improved by our approach. More sites relevant to a user’s needs were located in the first 100 sites, and a smaller number of sites was returned. As few users look beyond 100 sites, this is an important improvement.

Naturally, the cases used for assessing the performance of the Web search module are limited in this paper. For example, some users prefer cheap flights, while others prefer direct flights. In addition, only the *Air travel* domain ontology among the eight ontologies from [10] was enriched in this paper. After the other domain ontologies are enriched, our ontology-based Web search module can process a user’s query across the different domains by adding a domain related semantic choice to the user feed-back interface (Figure 4). This is non-trivial future work and relies on the fact that there is a bounded number of domains within E-Commerce.

While a large number of instances was extracted from a specific DW site, there are many E-Commerce DW sites which a crawling program cannot access, due to security reasons [14]. In fact, we faced several problems in reading HTML results from a number of Web sites. A message, “Please activate scripting” means that the site detects a Web browser of a client and its script option before it answers a user query. Thus, a crawling program needs a facility of a Web browser. Similarly, a message of “Permission denied” was encountered when we attempted what has been called *cross-site scripting*.

On the other hand, automatic downloading of Web documents from a hidden Web site [3] is easier than retrieving information from a backend database. In fact, a crawling program, which failed to read a DW page’s HTML content, could download Web content from a textual database such as PubMed without problems. Considering this

paper's result, namely that ontology instances extracted from the DW contribute to locating relevant Web sites, community-wide efforts are necessary to extract large numbers of instances from the DW.

References

1. Singh, M.P.: Deep Web structure. *IEEE Internet Computing* 6(5), 4-5 (September 2002)
2. He, B., Patel, M., Zhang, Z., Chang, K.C.-C.: Accessing the Deep Web: A Survey. *Communications of the ACM (CACM)* 50(5), 94-101 (2007)
3. Ntoulas, A., Zerefos, P., Cho, J.: Downloading Textual Hidden Web Content Through Keyword Queries. In: *Proc. of the ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, pp. 100-109 (2005)
4. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* 43(5), 907-928 (1995)
5. Davulcu, H., Vadrevu, S., Nagarajan, S., Ramakrishnan, I.V.: OntoMiner: bootstrapping and populating ontologies from domain-specific Web sites. *IEEE Intelligent Systems* 18(5), 24-33 (2003)
6. McDowell, L.K., Cafarella, M.: Ontology-driven Information Extraction with OntoSyphon. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006. LNCS*, vol. 4273, pp. 428-444. Springer, Heidelberg (2006)
7. An, Y.J., Geller, J., Wu, Y.-T., Chun, S.A.: Semantic Deep Web: Automatic Attribute Extraction from the Deep Web Data Sources. In: *Proc. of the 22nd Annual ACM Symposium on Applied Computing (SAC 2007)*, Seoul, Korea (March 2007)
8. Dou, D., McDermott, D.V., Qi, P.: Ontology Translation on the Semantic Web. *Journal on Data Semantics* 2, 35-57 (2005)
9. Wu, W., Doan, A., Yu, C.T., Meng, W.: Bootstrapping Domain Ontology for Semantic Web Services from Source Web Sites. In: Bussler, C.J., Shan, M.-C. (eds.) *TES 2005. LNCS*, vol. 3811, pp. 11-22. Springer, Heidelberg (2006)
10. An, Y.J., Geller, J., Wu, Y.-T., Chun, S.A.: Automatic Generation of Ontology from the Deep Web. In: *Proceedings of 6th International Workshop on Web Semantics (WebS 2007)*, September 3-7, 2007 (in Press, 2007)
11. Stanford Medical Informatics, Protégé 3.2.5 [Computer program API] (Retrieved, May 2007), <http://protege.stanford.edu/doc/pdk/api/index.html>
12. Stanford Medical Informatics, Protégé - OWL 3.2.1 [Computer program API] (Retrieved, May 2007), <http://protege.stanford.edu/download/release-javadoc-owl/>
13. Jansen, B.J., Spink, A., Saracevic, T.: Real Life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web. *Information Processing & Management* 36(2), 207-227 (2000)
14. Little, S., Embley, D., Scott, D., Yau, S.: Extracting Data Behind Web. In: *Proceedings of the Joint Workshop on Conceptual Modeling Approaches for E-business: A Web Service Perspective (eCOMO 2002)*, pp. 38-49 (October 2002)
15. Omelayenko, B.: Learning of ontologies for the Web: the analysis of existent approaches. In: *Proceedings of the International Workshop on Web Dynamics* (Retrieved, 2001), <http://www.dcs.bbk.ac.uk/webDyn/webDynPapers/omelayenko.pdf>
16. Weber, N., Buitelaar, P.: Web-based Ontology Learning with ISOLDE. In: *Proceedings of ISWC 2006 Workshop on Web Content Mining with Human Language Technologies* (2006), <http://www2.dfki.de/~paulb/>

17. Faatz, A., Steinmetz, R.: Ontology Enrichment Evaluation. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 497–498. Springer, Heidelberg (2004)
18. Stanford Center for Biomedical Informatics Research, Protégé 3.3.1 [Computer Program] (Retrieved, 2007), <http://protege.stanford.edu/download/registered.html>
19. Raghavan, S., Garcia-Molina, H.: Crawling the Hidden Web. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 29–138 (2001)
20. Liddle, S., Embley, D., Scott, D., Yau, S.: Extracting Data Behind Web Forms. In: Proceedings of the Joint Workshop on Conceptual Modeling Approaches for E-business: A Web Service Perspective (eCOMO 2002), pp. 38–49 (2002)
21. Ramachandran, R., Movva, S., Graves, S., Tanner, S.: Ontology-based Semantic Search Tool for Atmospheric Science. In: Proceedings of 22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, <http://ams.confex.com/ams/Annual2006/>
22. An, Y.J.: Ontology Learning for the Semantic Deep Web, Ph.D. Dissertation in Computer Science, New Jersey Institute of Technology (January 2008)

POEM: An Ontology Manager Based on Existence Constraints^{*}

Nadira Lammari, Cédric du Mouza and Elisabeth Métais

Lab. CEDRIC, CNAM
Paris, France
{lammari, dumouza, metais}@cnam.fr

Abstract. Whereas building and enriching ontologies are complex and tedious tasks, only few tools propose a complete solution to assist users. This paper presents POEM (Platform for Ontology crEation and Management) which provides a complete environment for ontology building, merge and evolution. POEM's algorithms are based on existence constraints and Boolean equations that allow to capture the semantic of concepts. POEM also integrates a semantic fusion that relies on Word-Net and supports different storage formats. The different functionalities are implemented as a set of web services.

1 Introduction

Sharing data from multiple heterogeneous sources has been for several years a challenging issue in the area of Information Systems. From federated multi-sources Data Bases to the Semantic Web challenge, modern applications need to intensively exchange information. There is now evidence that ontologies are the best solution for dealing with semantically heterogeneous data. By adding sense to data sources, ontologies allow to “understand” their contents - and then to provide pertinent information to the users.

Nowadays, all experts recognize ontology's building and management as one of the most critical problem in their large scale utilization. Many tools have been proposed but they mainly lack in automatically taking into account the semantic aspects in building, merging and validating ontologies. Usually the semantic part of the work is done manually without guideline. Nowadays, many -small or large- ontologies have been developed around the world; building a new one has to take into account these legacy systems.

The POEM tool provides a complete environment for ontology building, merging, evolution, edition, transformation and storage. Due to space limitation, only two aspects will be particularly developed in this paper: the concepts of *existence constraints* and Boolean equations that help to capture the semantic of concepts, and the construction and reuse (*i.e.*, merging) algorithms.

^{*} Work partially funded by SEMWEB project, <http://bat710.univ-lyon1.fr/~semweb/>

Related Work

Many research work had been dedicated for building and maintaining ontologies: (a) ontology learning methodologies from dictionary, from text, from XML documents, or from knowledge base [6,10,17], (b) ontology merging methods of existing ontologies [14,13], (c) ontology re-engineering methods [8], and finally (d) ontology construction from scratch [7,6,9].

Moreover, several approaches were recently proposed to perform the merging. For instance [11] presents a merging based on WordNet and the Jaccard measure. However this technique does not take into account the attributes of the different concepts leading to the merging of non-similar concepts or redundancy in the final ontology. [16] introduces another approach called FCA-Merge that uses natural language techniques and formal concept analysis techniques. In [15] the authors propose an automatic merging algorithm named OM algorithm, based on the confusion theory for detecting concept matching. This algorithm is however syntactic and does not merge concepts that are semantically equivalent.

Besides methodologies for building ontologies many tools are now available like PROTÉGÉ [4], CORPORUM [2], TEXT-TO-ONTO [5], CHIMAERA [1], ONTOLINGUA [3], etc. All these tools have an editing functionality. Some of them also present merge and/or validation and/or reasoning functionalities. They however lack mechanisms and policies implementing methods rules and guidelines.

The rest of the paper is structured as follows: Section 2 introduces the existence constraints and associated algorithms, Section 3 details the ontology merging technique, Section 4 emphases implementation and Section 5 concludes.

2 Existence Constraints and Associated Techniques

This section presents the concept of existence constraints and two of the basic algorithms (for normalization and extraction) that will be compound to build high level functionalities. These algorithms are detailed in [12].

2.1 Definitions of Existence Constraints

We distinguish three kinds of existence constraints: mutual, exclusive and conditioned existence constraints. A mutual existence constraint defined between two properties x and y of a concept c , denoted $x \leftrightarrow y$, describes the fact that any instance associated to c has simultaneously the properties x and y . An exclusive existence constraint defined between two properties x and y of a concept c , denoted $x \nleftrightarrow y$, means that any instance associated to c that has a property x can not have y as another property and conversely. Finally, a conditioned existence constraint defined between two properties x and y of a concept c , denoted $x \mapsto y$, captures the fact that any instance of c described by the property x must also be described by the property y . The inverse is not always true. For example, let **vehicle** be a concept representing vehicles such as planes and boats, and described by a set of properties: **move**, **transport**, **fly** and **float**. The fact

that any vehicle moves and transports is translated by **transport** $\leftrightarrow$ **moves**. Also, every vehicle that moves doesn't systematically fly. However, every vehicle that flies must move. These two assertions can be formalized using the conditioned existence constraint: **fly** $\rightarrow$ **move**. We suppose in our example that only planes fly and there is no plane that floats. This hypothesis is translated into the exclusive existence constraint: **float** $\nrightarrow$ **fly**.

2.2 Normalization Technique

The normalization technique enables to build ontologies from a set of properties and a set of existence constraints. It consists in automatically deducing valid property subsets that are compatible with the existence constraints. Let S be a subset of the property set of a given ontology O . S is said to be a valid subset of properties of O if and only if S satisfies the following three rules:

- **Homogeneity Rule:** Any property of O linked to a property of S by a mutual existence constraint is in S .
- **Non-partition Rule:** There is no exclusive existence constraint between properties of S .
- **Convexity Rule:** Any property of O required by a group of properties of S is in S (taking into account the conditioned existence constraints).

Intuitively, the sets of coexisting properties are first deduced, by taking into account the mutual existence constraints. Then, by means of exclusive existence constraints, the sets of properties that may coexist are derived. Finally, sets that are compatible with the conditioned existence constraints are selected among the sets of properties that may coexist. These sets describe concepts represented by the ontology. These concepts are then organized into an Is_A inheritance graph.

2.3 Translation Techniques

We present here two translation techniques described in [12] for the building of ontologies from scratch and that we implanted in POEM. The first one called **O_TO_BF**, transforms an ontology into a Boolean function while the second called **BF_TO_O** is the reverse operation. **O_TO_BF** consists in building a disjunctive Boolean function $\phi(x_1, \dots, x_n)$ where each x_i corresponds to a property of the ontology and each maxterm T represents a type of instance represented by each concept C of the ontology. This maxterm is a conjunction of x'_i variables where each x'_i is either equal to x_i if the property associated to x_i describes T or equal to $\bar{x}_i$ otherwise.

BF_TO_O derives from a Boolean function the set of existence constraints and then using the normalization technique deduces the ontology. For the generation of the existence constraints from a Boolean function $\phi(x_1, \dots, x_n)$ where each x_i corresponds to a property a_i , the technique transforms ϕ from a disjunctive form to a conjunctive one and then, by analyzing the different minterms of the transformed expression, it derives the existence constraints by applying the following rules to the conjunctive function:

- **Rule 1:** A minterm of ϕ of type x_i is translated into the non-null constraint
- **Rule 2:** A minterm of type $(\bar{x}_i + \dots + \bar{x}_j + x_k)$ describes a conditioned existence constraint “ $a_i, \dots, a_j \mapsto a_k$ ”.
- **Rule 3:** A minterm of type $(\bar{x}_i + \bar{x}_j)$ is translated into the exclusive existence constraint “ $a_i \not\mapsto a_j$ ”.
- **Rule 4:** Two attributes a_i and a_j coexist in R ($a_i \leftrightarrow a_j$) iff the two conditioned existence constraints $a_i \mapsto a_j$ and $a_j \mapsto a_i$ are derived from ϕ .

3 Ontology Merging

POEM allows to build an ontology either from scratch or by reuse. For lack of space, we will focus on our building mechanism based on merging. The different ontology construction processes from scratch may be found in [12]. The merging process can basically be decomposed into two steps: the matching step and the integration step.

3.1 The Matching Step

The matching technique in POEM is realized for two normalized ontologies. It encompasses two phases, property merging and concept merging. Let O_1 and O_2 be these two ontologies. To achieve the matching between them, we first retrieve, for each property p from O_1 and O_2 , its synset S_p from the general ontology WordNet and then we compute for any couple of properties (p_i, p_j) , such that $p_i \in O_1$ and $p_j \in O_2$, the similarity degree (denoted $SimDegree(p_i, p_j)$) between them according to the following formula:

$$SimDegree(p_i, p_j) = \frac{|S_{p_i} \cap S_{p_j}|}{|S_{p_i} \cup S_{p_j}|}$$

According to this formula, the more numerous the synonyms of the two properties are, the higher the similarity degree is. Once the similarity degrees between properties measured, we perform the property matching. In POEM, we merge p_i and p_j iff they share at least one synonym, and there is no property in O_1 (resp. O_2) semantically closer to p_j (resp. p_i). More formally two properties p_i and p_j are matched if and only if the three following conditions are satisfied:

- (i) $SimDegree(p_i, p_j) > 0$,
- (ii) $\nexists p_k \in P_2, SimDegree(p_i, p_k) > SimDegree(p_i, p_j)$,
- (iii) $\nexists p_k \in P_1, SimDegree(p_k, p_j) > SimDegree(p_i, p_j)$.

where P_1 (resp. P_2) denotes the set of properties from O_1 (resp. O_2).

When merging two properties, we choose as common name for them the most specific synonym syn_{ij} they shared, *i.e.* the first one that occurs in $(S_{p_i} \cap S_{p_j})$. Once a property merging is performed, we replace p_i and p_j by syn_{ij} in their respective ontology.

For the merging of concepts, our approach distinguishes itself from other existing techniques because it considers all the following cases for the similarity between concepts:

- **Case 1:** a similarity both between concept names and between names of their respective properties,
- **Case 2:** a similarity only between concept names,
- **Case 3:** a similarity only between property names of the concepts.

More precisely, let c_i and c_j be two concepts from an ontology O and π_{c_i} (resp. π_{c_j}) be the property set of c_i (resp. c_j). To estimate the similarity between c_i and c_j , we refer to Jaccard measure defined by the ratio between the number of shared properties of these two concepts and the union of their properties, *i.e.*:

$$jaccard(c_i, c_j) = \frac{|\pi_{c_i} \cap \pi_{c_j}|}{|\pi_{c_i} \cup \pi_{c_j}|}$$

We assume the existence of a threshold value τ for the Jaccard measure to assert the similarity or not between two concepts. We also define a decreasing factor γ to attribute a lower similarity value when both concept names and property names are not similar (what is considered as the highest similarity). Our algorithm for estimating the similarity between two concepts c_i and c_j is:

```

if  $c_i$  and  $c_j$  share at least one synonym (their synsets are not disjoint) then
  if  $jaccard(c_i, c_j) \geq \tau$  then
     $c_i$  and  $c_j$  are similar with a similarity degree equal to  $jaccard(c_i, c_j)$  (case 1)
  else
    if  $c_i$  and  $c_j$  have at least one parent  $Parent(c_i)$  and  $Parent(c_j)$  such that these
      two parents share at least one synonym (their synsets are not disjoint) and
      ( $jaccard(Parent(c_i), Parent(c_j)) \geq \tau$ ) then
        they are similar with a (lower) similarity degree equal to  $\gamma \times jaccard(c_i, c_j)$  (case 2)
      else unable to evaluate the similarity; return the concepts to the ontology engineer
    endif
  endif
else
  if  $jaccard(c_i, c_j) \geq \tau$  then
    they are similar with a (lower) similarity degree equal to  $\gamma \times jaccard(c_i, c_j)$  (case 3)
  else the two concepts are not similar
  endif
end

```

After estimating all the similarity degrees between concepts, we merge two concepts c_i and c_j if they have a positive similarity degree, and if a higher value can not be found by replacing one of the two concepts by another one from the same ontology. The property set of the resulting concept c_{res} is the union of c_i and c_j property sets. To chose a name for c_{res} , we adopt the following strategy:

- in case 1, we choose for c_{res} the name of the concept that participates more in its building, that is to say the concept which has more properties in c_{res} ,
- in case 2, we choose the most specific name, *i.e.* the name that appears first in the common synset of the two concepts,

- in case 3: we compute the confusion value (see [15] for definition) between concept names according to their respective position in WordNet and we choose the name of the concept with the lowest value.

Finally, the matching technique allows us to give the same names to similar concepts and properties. The two ontologies O_1 and O_2 are transformed according to the result of the matching into respectively O'_1 and O'_2 .

3.2 The Integration Step

The final step of our process consists in building by a merge the ontology O_3 from the two ontologies O'_1 and O'_2 obtained after matching. For that purpose, we first translate the two ontologies into two Boolean functions ϕ_1 and ϕ_2 using the `O_TO_BF` mapping technique for both ontologies, then we gather the two resulting Boolean functions into one Boolean function ϕ_3 and finally, by applying the `BF_TO_O` mapping technique, we deduce the resulting ontology.

To gather the Boolean functions ϕ_1 and ϕ_2 into a Boolean function ϕ_3 we proceed to the following step-by-step algorithm:

```

let  $B$  the set of variables of  $\phi_1$  and  $\phi_2$ 
for each minterm  $T$  of  $\phi_1$  and  $\phi_2$  do
  for each variable  $x$  in  $B$  do
    if neither  $x$  nor  $\bar{x}$  appears in  $T$  then  $T = T.\bar{x}$  endif
  endfor
endfor
if  $T$  does not appear in  $\phi_3$  then  $\phi_3 = \phi_3 + T$  endif

```

4 Implementation

We design POEM in accordance with four principles: (i) modularity (POEM is built as a set of seven modules, one for each basic functionality), (ii) reusability (a web service architecture), (iii) extensibility and (iv) guidance (three levels of guidance are proposed to give a cognitive support to human, expert or not).

The global architecture is basically a three-tier architecture with POEM, the Sesame system and a database. The services in POEM are implemented in Java JDK1.5 using MyEclipse 5.0 and the web service library XFire. Sesame acts as a middleware for gathering and storing ontologies and metadata. It is deployed with the Tomcat web server.

Figure 1 shows the interface of the POEM tool when merging two ontologies. The upper part of the interface is dedicated to the selection of ontologies and export parameters. You can for instance with this interface select a ntriples ontology and export it in RDF. You can also load it and proceed to concept extraction that are displayed in the window below. For instance here we have for the first ontology classes **Airplane**, **Boat** and **Vehicle** along with their properties. Finally POEM displays the ontologies loaded at bottom of the window. In Figure 2(a) we see the result got by proceeding to the fusion with POEM. We

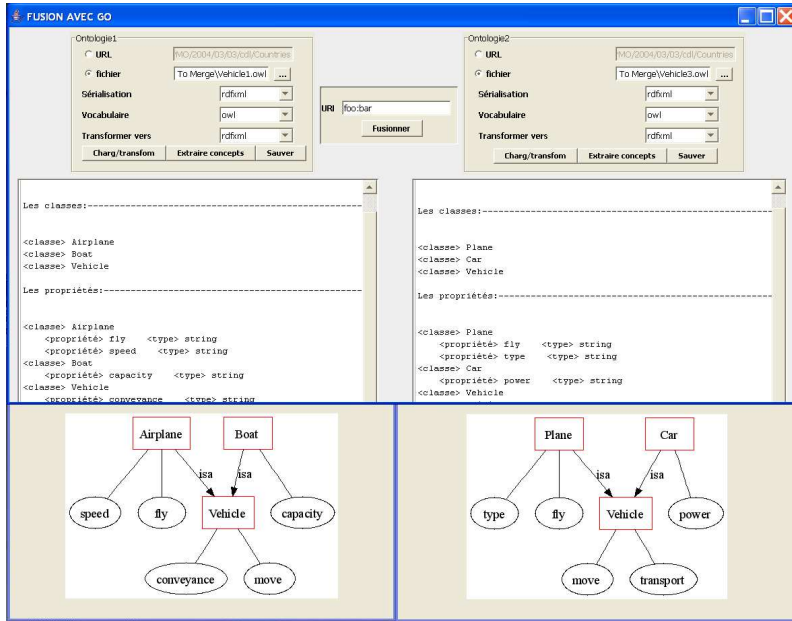
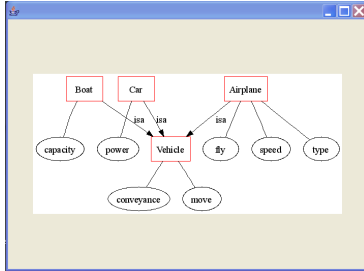
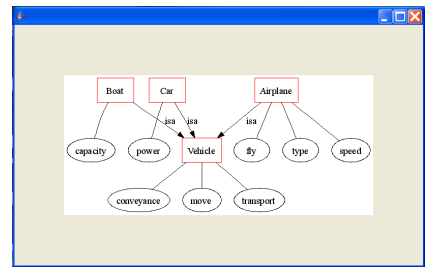


Fig. 1. Merging two ontologies with POEM



(a) with POEM



(b) with PROTÉGÉ+PROMPT

Fig. 2. Result of the merging

notice that POEM, based on semantical similarity of the concept names and of their properties, decided to merge the concepts **Airplane** and **Plane** and the resulting concept **Airplane** beholds the union of the properties that are semantically different. The concept **Vehicle** has only two properties, since the former **conveyance** and **transport** properties were identified as similar. Figure 2(b) represents the merge with PROTÉGÉ 3.3.1 (one of the most frequently used ontology manager tool). If the concepts were, in this case, merge correctly, we see that the merge did not apply to properties. Similarly, our experiments show that concepts whose names are not synonyms in WordNet but have numerous similar properties are also not merge in PROTÉGÉ.

5 Conclusion

In this paper we present the POEM existence constraints based platform to assist ontology creation and maintenance with a guidance module. POEM takes into account the semantic, via semantic constraints for the building tool and via WordNet access for ontology merging. In this paper we focused on the building and the merging algorithms, although other algorithms presented in [12] are also implemented. For the next step we plan to integrate the detection of similarities between properties in the ontology construction processes. Another perspective is to extract properties and constraints from textual sources.

References

1. Chimaera, <http://www.ksl.stanford.edu/software/chimaera/>
2. Corporum-OntoExtract, <http://www.Ontoknowledge.org>
3. OntoLingua, <http://ontolingua.stanford.edu/>
4. Protege, <http://protege.stanford.edu/>
5. Text-To-Onto, <http://webster.cs.uga.edu/~mulye/SemEnt/>
6. Agirre, E., Ansa, O., Hovy, E.H., Martínez, D.: Enriching Very Large Ontologies Using the WWW. In: Proc. Intl. ECAI Workshop on Ontology Learning (OL) (2000)
7. Davies, J., Fensel, D., van Harmelen, F.: Towards the Semantic Web: Ontology-driven Knowledge Management. Wiley, Chichester (January 2003)
8. Gómez-Pérez, A., Rojas-Amaya, D.: Ontological Reengineering for Reuse. In: Fensel, D., Studer, R. (eds.) EKAW 1999. LNCS (LNAI), vol. 1621, pp. 139–156. Springer, Heidelberg (1999)
9. Grüninger, M., Fox, M.S.: Methodology for the Design and Evaluation of Ontologies. In: Proc. Intl. IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 1–10 (1995)
10. Khan, L., Luo, F.: Ontology Construction for Information Selection. In: Proc. Intl. Conf. on Tools with Artificial Intelligence (ICTAI), pp. 122–127 (2002)
11. Kong, H., Hwang, M., Kim, P.: Efficient Merging for Heterogeneous Domain Ontologies Based on WordNet. Jour. of Advanced Computational Intelligence and Intelligent Informatics (JACIII) 10(5), 733–737 (2006)
12. Lammari, N., Métais, E.: Building and Maintaining Ontologies: a Set of Algorithms. Data Knowl. Eng. (DKE) 48(2), 155–176 (2004)
13. McGuinness, D., Fikes, R., Rice, J., Wilder, S.: An Environment for Merging and Testing Large Ontologies. In: Proc. Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR), pp. 483–493 (2000)
14. Noy, N., Musen, M.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proc. Nat. Conf. on Artificial Intelligence and on Innovative Applications of Artificial Intelligence, pp. 450–455 (2000)
15. Rasgado, A., Guzman, A.: A Language and Algorithm for Automatic Merging of Ontologies. In: Proc. Intl. Conf. on Computing (CIC), pp. 180–185 (2006)
16. Stumme, G., Maedche, A.: FCA-MERGE: Bottom-Up Merging of Ontologies. In: Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI), pp. 225–234 (2001)
17. Suryanto, H., Compton, P.: Discovery of Ontologies from Knowledge Bases. In: Proc. Intl. Conf. on Knowledge Capture (K-CAP), pp. 171–178 (2001)

Extending Inconsistency-Tolerant Integrity Checking by Semantic Query Optimization

Hendrik Decker*

Instituto Tecnológico de Informática, Valencia, Spain
hendrik@iti.es

Abstract. The premise that all integrity constraints be totally satisfied guarantees the correctness of methods for simplified integrity checking, and of semantic query optimization. This premise can be waived for many integrity checking methods. We study how it can be relaxed also for applying semantic query optimization to an inconsistency-tolerant evaluation of simplified constraints.

1 Introduction

For a given update, all known approaches to simplified integrity checking only check certain instances of constraints, called ‘relevant cases’. Informally, relevant cases are those that are potentially violated by the update. By focusing on relevant cases, or further simplifications thereof, the otherwise prohibitive cost of integrity checking can be reduced to feasible proportions [3,7].

Many methods for simplified integrity checking proceed along two phases. First, a so-called *simplification* (essentially, a query, or a set of queries, derived from relevant cases) is computed. Part or all of the simplification phase can be computed at constraint specification time, for parametrized update patterns. Second, the simplification is evaluated, at update time. The evaluation phase typically involves access to stored data, but usually is more efficient than a brute-force evaluation of all integrity constraints [3,7].

In [5] it was shown that several integrity checking methods can be soundly applied also in databases that are inconsistent with some constraints. This came as a surprise, because the focusing of such methods on relevant cases has always been justified by the premise that integrity be totally satisfied, before updates could be admitted and efficiently checked for preserving consistency. For convenience, let us call this requirement the *total integrity premise*. Methods that continue to work well when this premise is waived are called *inconsistency-tolerant*.

In [4], we have investigated how different steps taken during the simplification phase may have an effect on inconsistency tolerance. Up to now, the evaluation phase has not been studied with regard to its relationship to inconsistency tolerance. Rather, that phase has been deemed independent of inconsistency tolerance. However, that point of view can no longer be upheld as soon as the

* Partially supported by FEDER and the Spanish MEC grant TIN2006-14738-C02-01.

evaluation of simplifications is optimized by taking integrity constraints into account.

A particular form of using integrity constraints for speeding up query evaluation is *semantic query optimization (SQO)* [2]. In this paper, we study the use of SQO for optimizing the evaluation of inconsistency-tolerant integrity checking.

In section 2, we ask the reader to consent on some preliminary conventions and a basic definition. In section 3, we revisit the concept of inconsistency-tolerant integrity checking. In section 4, a relationship of the latter to SQO is established. Surprisingly, it turns out that SQO can be used for optimizing integrity checking, even if the total integrity premise does not hold. In section 5, we conclude with an outlook to future work.

2 Preliminaries

Unless specified otherwise, we are going to use terminology and notations that are conventional in the logic databases community (see, e.g., [9]). Throughout this paper, let ‘method’ always mean an integrity checking method.

An *update* is a finite set of database clauses to be inserted or deleted. An *integrity constraint* (in short, *constraint*) is a first-order predicate logic sentence, represented either as a denial (i.e., a clause without head whose body is a conjunction of literals) or in *prenex normal form* (i.e., quantifiers outermost, negations innermost). An *integrity theory* is a finite set of constraints.

From now on, let the symbols D , IC , U , I and $\mathcal{M}$ always denote a database, an integrity theory, an update, a constraint and, resp., a method. For simplicity (and by a slight deviation from divergent definitions in the literature), let us assume that the semantics of D is given by a distinguished Herbrand model of D , and that I is *satisfied* (*violated*) in D iff I is *true* (resp., *false*) in that model. Further, let $D(IC) = sat$ and $D(I) = sat$ denote that IC or, resp., I is satisfied in D , and $D(IC) = vio$ ($D(I) = vio$) that it is violated. ‘Consistency’ and ‘inconsistency’ are synonymous with ‘satisfied’ and, resp., ‘violated’ integrity. We write D^U to denote the updated database; D and D^U are also referred to as the *old* and the *new state*, respectively.

Each $\mathcal{M}$ can be formalized as a mapping that takes as input a triple (D, IC, U) and outputs upon termination either *ok* or *ko*, where *ok* means that $\mathcal{M}$ sanctions the update and *ko* that it does not.

For simplicity, we only consider classes of triples (D, IC, U) such that the computation of $\mathcal{M}(D, IC, U)$ terminates. In practice, that can always be achieved by a timeout mechanism with output *ko*.

Note that, in general, *ok* and *ko* are not the same as the answers *yes* or, resp., *no* to integrity constraints as closed queries; after all, the interpretation of positive and negative answers to queried constraints in denial form is opposed to the interpretation of the same answers to queried constraints in prenex normal form. Similarly, we deliberately do not use, instead of *ok* and *ko*, the truth values *true* and *false*, because, in general, the semantics of integrity may not be based on logical consequence. Also, we do not use the integrity values *sat* and *vio* to

denote the output of $\mathcal{M}$ (as done, somewhat carelessly, in earlier papers), since *sat* in place of *ok* could suggest that $D^U(IC) = \text{sat}$, although $\mathcal{M}$ may tolerate inconsistency and hence output *ok* even if $D^U(IC) = \text{vio}$.

Soundness and completeness of a method $\mathcal{M}$ can be defined as follows.

Definition 1. (*Sound and Complete Integrity Checking*)

$\mathcal{M}$ is called *sound* or, resp., *complete* if, for each (D, IC, U) such that $D(IC) = \text{sat}$, (1) or, resp., (2) holds.

$$\text{If } \mathcal{M}(D, IC, U) = \text{ok} \text{ then } D^U(IC) = \text{sat}. \quad (1)$$

$$\text{If } D^U(IC) = \text{sat} \text{ then } \mathcal{M}(D, IC, U) = \text{ok}. \quad (2)$$

Although (1) and (2) only refer to the output *ok* of $\mathcal{M}$, symmetric conditions for *ko* could be defined. We omit that, since, as is easily seen, soundness and, resp., completeness for *ko* is equivalent to.

Several methods (e.g., in [8,3]) have been shown to be sound and complete for significant classes of triples (D, IC, U) . Other methods (e.g., in [6]) are only shown to be sound. Thus, they provide sufficient but not necessary conditions for guaranteeing integrity of D^U . In this paper, each named method is known to be sound, and each anonymous method is tacitly assumed to be sound, unless indicated otherwise.

3 Inconsistency-Tolerant Integrity Checking

In this section, we revisit the approach to inconsistency-tolerant integrity checking as introduced in [5].

Methods usually justify the correctness (completeness) of their efficiency improvements by the total integrity premise $D(IC) = \text{sat}$ of (1) (resp., (2)). The idea of inconsistency-tolerant integrity checking is to have methods that are capable of preserving integrity without insisting on that premise. Definition 2, below, provides a way to realize this idea.

Each constraint I can be conceived as a set of particular instances, called ‘cases’, of I , such that I is satisfied iff all of its cases are satisfied. Thus, integrity maintenance can focus on satisfied cases, and check if their satisfaction is preserved across updates. Violated cases can be ignored or dealt with at some more convenient moment. This is captured by the following definition.

Definition 2. (*Inconsistency-Tolerant Integrity Checking*)

a) A variable x is called a *global variable* in I if x is $\forall$ -quantified in I and $\exists$ does not occur left of the quantifier of x .

b) If I is of the form $\leftarrow B$ or QW (where B is a conjunction of literals, Q is a vector of $\forall$ -quantifiers of all global variables in I and W is a formula in prenex normal form), and if ζ is a substitution of the global variables in I , then $\leftarrow B\zeta$ or, resp., $\forall(W\zeta)$ is called a *case* of I , where $\forall(\cdot)$ denotes universal closure. If ζ is a ground substitution, then $\leftarrow B\zeta$ or, resp., $\forall(W\zeta)$ is called a *basic case*.

c) Let $\text{Cas}(IC)$ denote the set of all cases of all $I \in IC$, and $\text{SatCas}(D, IC)$ the set of all $C \in \text{Cas}(IC)$ such that $D(C) = \text{sat}$.

d) $\mathcal{M}$ is called *sound*, resp., *complete wrt. inconsistency tolerance* if, for each (D, IC, U) and each $C \in \text{SatCas}(D, IC)$, (3) or, resp., (4) holds.

$$\text{If } \mathcal{M}(D, IC, U) = \text{ok} \text{ then } D^U(C) = \text{sat}. \quad (3)$$

$$\text{If } D^U(C) = \text{sat} \text{ then } \mathcal{M}(D, IC, U) = \text{ok}. \quad (4)$$

For convenience, we may speak, from now on, of an ‘inconsistency-tolerant method’ or a ‘complete inconsistency-tolerant method’ when it is clear from the context that we actually mean a method that is sound or, resp., complete wrt. inconsistency tolerance.

Example 1. For relations p and q , let the second column of q be subject to the foreign key constraint $I = \forall x, y \exists z (q(x, y) \rightarrow p(y, z))$, referencing the primary key column of p , constrained by $I' = \leftarrow p(x, y), p(x, z), y \neq z$. The global variables of I are x and y ; all variables of I' are global. For $U = \text{insert } q(a, b)$, a typical method $\mathcal{M}$ only evaluates the simplified basic case $\exists z p(b, z)$ of I . If, for instance, (b, b) and (b, c) are rows in p , $\mathcal{M}$ outputs *ok*, ignoring all irrelevant violated cases such as, e.g., $\leftarrow p(b, b), p(b, c), b \neq c$ and I' , i.e., all extant violations of the primary key constraint. $\mathcal{M}$ is inconsistency-tolerant if it always ignores irrelevant violations. Of course, $\mathcal{M}$ outputs *ko* if there is no tuple matching (b, z) in p .

Obviously, each sound (3), resp., complete (4) inconsistency-tolerant method is sound and, resp., complete in the sense of (1), resp., (2). Thus, inconsistency-tolerant integrity checking significantly generalizes the traditional approach. The latter imposes the total integrity premise, which deprives most applications of integrity checking of a theoretical foundation and justification, since they often are used also in the presence of inconsistency (which sometimes may not even be noticed). As opposed to that, inconsistency-tolerant methods are capable of tolerating any amount of inconsistency, including unsatisfiable integrity theories, while making sure that updates do not introduce new violated cases of integrity.

Many known methods for integrity checking are inconsistency-tolerant. For a representative selection of them, their inconsistency tolerance or intolerance has been assessed in [5].

4 Semantic Query Optimization for Integrity Checking

Most methods in the literature pay little or no attention to the mechanisms employed for evaluating simplifications. But in practice, considerable differences of performance can be obtained by different query plans. In particular, query optimization mechanisms such as SQO may use “semantic input”, i.e., integrity constraints, for speeding up their performance. The integrity constraints that provide such semantic input for query evaluation must be satisfied. Hence, to use SQO for evaluating simplifications is problematic, since a possibly violated

integrity theory presumably cannot be used soundly as semantic input for SQO. However, we are going to see that SQO can be used for optimizing the evaluation of simplifications if the constraints provided as semantic input are satisfied.

In 4.1, we show how to use SQO for optimizing the evaluation phase of integrity checking if the total integrity premise holds. In 4.2, we show that SQO also may serve for optimizing inconsistency-tolerant integrity checking.

4.1 How to Use SQO for Integrity Checking

SQO uses integrity constraints for optimizing query answering [2]. In general, using an integrity theory as semantic input for SQO is sound if that theory is satisfied in the given database state. For example, the integrity constraint $I = \leftarrow p$ can be used to give a negative answer to the query $\leftarrow p, q$ in some state D as follows. $D(I) = \text{sat}$ means that p is *false* in D , hence the answer to $\leftarrow p, q$ must be empty. Note that, by using the semantic knowledge that I is satisfied in D , this answer can be given without any access to stored data.

The premise that the semantic input be satisfied, which ensures the soundness of SQO, is similar to the total integrity premise, which ensures the soundness of traditional integrity checking (1). So, in analogy to inconsistency-tolerant integrity checking, the question arises whether SQO also works when this premise does not hold. In general, the answer is ‘no’, as can be seen by a slight modification of the example above. If $D(I) = \text{vio}$ because p is *true* in D , then SQO cannot use I for answering $\leftarrow p, q$.

Despite this negative result, we are going to see below that there is a way to use SQO for optimizing the evaluation phase of simplified integrity checking, which, in subsection 4.2, turns out to be inconsistency-tolerant.

The basic idea to optimize the evaluation of a simplification S is to use constraints as semantic input only if they are known to be satisfied in the state in which S is to be evaluated. For instance, if the total integrity premise holds, then each constraint that is not relevant for a given update is known to be satisfied in the old and in the new state. Thus, irrelevant constraints can serve as semantic input for SQO. More generally, each irrelevant case C of any constraint I can be used by SQO for optimizing the evaluation phase if C it is known to be satisfied, even if I is violated in the old state or potentially violated by the update.

This idea is formalized as follows. For each $\mathcal{M}$ and each triple (D, IC, U) , let $\text{Sim}_{\mathcal{M}}(D, IC, U)$ denote the simplification of IC computed by $\mathcal{M}$ for the update U of D . Thus, $\mathcal{M}(D, IC, U)$ is determined by the evaluation of $\text{Sim}_{\mathcal{M}}(D, IC, U)$. Now, we recall that it depends on $\mathcal{M}$ whether $\text{Sim}_{\mathcal{M}}(D, IC, U)$ is evaluated in the old state D (such as, e.g., in the method described in [3]) or in the new state D^U (e.g., the method in [8]). Thus, let the state in which $\text{Sim}_{\mathcal{M}}(D, IC, U)$ is evaluated be denoted by D^* . (For simplicity, this denotation does not indicate its dependence on $\mathcal{M}$, IC and U , since that will always be clear from the context.)

If $\text{Sim}_{\mathcal{M}}(D, IC, U)$ is evaluated by SQO, then also the integrity theory used as semantic input for SQO must be specified. Let us describe that integrity theory abstractly as a mapping sgo , which, for each method $\mathcal{M}$ and each triple

(D, IC, U) , determines the semantic input $sqo(\mathcal{M}, D, IC, U)$ to be used by SQO for evaluating $\text{Sim}_{\mathcal{M}}(D, IC, U)$.

Last, let $\mathcal{M}^{sqo}$ be obtained from $\mathcal{M}$ by evaluating $\text{Sim}_{\mathcal{M}}(D, IC, U)$ by SQO with $sqo(\mathcal{M}, D, IC, U)$ as semantic input. From this formalization, the result below follows.

Theorem 1. $\mathcal{M}^{sqo}$ is a sound integrity checking method if, for each triple (D, IC, U) , $D^*(sqo(\mathcal{M}, D, IC, U)) = sat$ holds. $\square$

Informally, theorem 1 means that the evaluation phase of each method $\mathcal{M}$ can be optimized by using SQO for evaluating the simplification $\text{Sim}_{\mathcal{M}}(D, IC, U)$ in the state D^* with semantic input $sqo(\mathcal{M}, D, IC, U)$ if that input is satisfied in D^* . As already mentioned, $sqo(\mathcal{M}, D, IC, U)$ is satisfied in both D and D^U if it consists of nothing but cases of constraints that are considered irrelevant by $\mathcal{M}$, provided that the total integrity premise holds. Example 2 illustrates this.

Example 2. Let $D = \{q(a, b), r(c)\}$, $IC = \{\leftarrow q(x, x), \leftarrow q(a, y), r(y)\}$ and $U = insert\ r(a)$. Clearly, $D(IC) = sat$. For most methods $\mathcal{M}$, $\text{Sim}_{\mathcal{M}}(D, IC, U) = \leftarrow q(a, a)$ (the conjunct $r(a)$ is dropped from the relevant case $\leftarrow q(a, a), r(a)$ since $r(a)$ is *true* in D^U). Now, the evaluation of $\text{Sim}_{\mathcal{M}}(D, IC, U)$ by $\mathcal{M}^{sqo}$ does not need to access the database, since SQO detects that $\leftarrow q(a, a)$ is subsumed by the irrelevant constraint $\leftarrow q(x, x)$, which is satisfied in both D and D^U . Hence, $\mathcal{M}^{sqo}(D, IC, U) = ok$.

4.2 SQO for Inconsistency-Tolerant Integrity Checking

The following trivial example shows that the use of SQO for evaluating simplifications is in general not sound if the total integrity premise does not hold.

Example 3. Let $D = \{p\}$ and $IC = \{\leftarrow p, \leftarrow p, q\}$. Clearly, $D(IC) = vio$. Further, let $U = insert\ q$, and assume that $\mathcal{M}$ correctly identifies $\leftarrow p, q$ as the only relevant case. Since $\leftarrow p$ is not relevant wrt. U , $\mathcal{M}$ may infer by the total integrity premise that $\leftarrow p$ is satisfied in both D and D^U . For evaluating $\leftarrow p, q$, SQO detects that $\leftarrow p, q$ is subsumed by $\leftarrow p$, which yields the wrong output *ok*.

However, example 4 illustrates that SQO may be used for optimizing integrity checking also in the presence of integrity violations.

Example 4. Let $D = \{q(a, b), r(b), r(c), s(b, c)\}$, $U = insert\ r(a)$ and $IC = \{\leftarrow q(x, x), \leftarrow q(a, y), r(y), s(y, z)\}$. Clearly, the case $\leftarrow q(a, b), r(b), s(b, c)$ is violated in D , while all other basic cases are satisfied. For most methods $\mathcal{M}$ for simplified integrity checking, $\text{Sim}_{\mathcal{M}}(D, IC, U) = \leftarrow q(a, a), s(a, z)$. The evaluation of $\leftarrow q(a, a), s(a, z)$ does not need to access the database if SQO is used, since this query is subsumed by $\leftarrow q(x, x)$. The latter is an irrelevant constraint and hence can be assumed to be part of the semantic input for the evaluation of $\leftarrow q(a, a), s(a, z)$ by SQO. Hence, $\mathcal{M}^{sqo}(D, IC, U) = ok$.

Example 4 illustrates that SQO can be used for optimized integrity checking, even if not all constraints are satisfied. However, as seen in example 3, methods which use SQO are in general not inconsistency-tolerant in the sense of definition 2. So, the question arises under which conditions $\mathcal{M}^{sqo}$ is inconsistency-tolerant. An answer is given by theorem 2. It can be shown by applying the definitions.

Theorem 2. $\mathcal{M}^{sqo}$ is inconsistency-tolerant if $\mathcal{M}$ is, and, for each triple (D, IC, U) , $D^*(sqo(\mathcal{M}, D, IC, U)) = sat$ holds. $\square$

A possible problem with theorem 2 is that $sqo(\mathcal{M}, D, IC, U)$, i.e., the semantic input, is required to be satisfied in D^* , i.e., the state in which $\text{Sim}_{\mathcal{M}}(D, IC, U)$ is to be evaluated by SQO. That is immaterial if the total integrity premise holds, since then, it suffices to make sure that the semantic input consists of nothing but irrelevant cases of constraints. However, the integrity status of $sqo(\mathcal{M}, D, IC, U)$ is not necessarily known at update time if inconsistency is to be tolerated.

A possible solution of this problem is an offline computation of a set of cases of constraints in IC that are satisfied in D . Thus, at update time, any available set of such cases that also are irrelevant wrt. U can be used in place of $sqo(\mathcal{M}, D, IC, U)$ as semantic input to optimize the evaluation of $\text{Sim}_{\mathcal{M}}(D, IC, U)$.

Another scenario where the problem identified above is ruled out is given whenever the total integrity premise is enforced on distinguished, ‘hard’ parts of the integrity theory, while violations may be tolerated for other, ‘soft’ constraints. Then, theorem 2 applies nicely if all elements in $sqo(\mathcal{M}, D, IC, U)$ are hard constraints, or cases thereof.

To conclude, we offer a thought that needs further study. Instead of limiting the semantic input $sqo(\mathcal{M}, D, IC, U)$ to sets of irrelevant cases, it may be more advantageous to reason with ‘generalized cases’ that exclude violated cases from otherwise satisfied constraints or cases thereof. The following modification of example 4 illustrates this thought. Consider $D' = D \cup \{q(b, b)\}$, and IC, U as above. Again, we obtain the sound output $\mathcal{M}^{sqo}(D', IC, U) = ok$ without access to the database, but this time in the presence of the violated irrelevant case $\leftarrow q(b, b)$. The subsumption-based reasoning of SQO as in example 4 remains sound because the simplification $\leftarrow q(a, a), s(a, z)$ is subsumed by the set of irrelevant cases of $\leftarrow q(x, x)$. Instead of having to compute that set at or before update time, it seems to be easier to reason with the generalized case $\leftarrow q(x, x), x \neq b$, a parametrized version of which can be easily obtained at constraint specification time. That generalized case excludes the violated case $\leftarrow q(b, b)$ from $\leftarrow q(x, x)$ in IC , and it is easy to infer that it subsumes the simplification $\leftarrow q(a, a), s(a, z)$.

5 Conclusion

Traditionally, the correctness of simplified integrity checking relies on the total integrity premise. It requires that the database be consistent with the integrity theory imposed on it at all times, and in particular before each update. In [5] we have shown that the total integrity premise can be waived without any cost.

Similarly, SQO requires that its semantic input, i.e., the integrity theory used by SQO for optimizing the evaluation of given queries, be totally satisfied. Although this requirement cannot be abdicated as easily as the total integrity premise for integrity checking, we have shown that it is possible to use SQO for optimizing the evaluation of simplifications of possibly violated constraints. More precisely, we have shown that traditional integrity checking, which imposes the total integrity premise, can be optimized by using irrelevant cases of constraints as semantic input for SQO. Also, we have shown that the use of irrelevant cases for applying SQO to the evaluation of simplifications is possible even if the total integrity premise does not hold, as long as simplifications are computed by some inconsistency-tolerant integrity checking method.

More research is needed to find out about further possible benefits of using query optimization (not just SQO) for optimizing inconsistency-tolerant integrity checking. We also intend to study the relationship between inconsistency-tolerant integrity checking and consistent query answering (CQA) [1]. That should be interesting since, like using SQO for optimizing inconsistency-tolerant integrity checking, CQA also is an inconsistency-tolerant application of SQO. Other situations where violated constraints need to be tolerated can often be encountered in replicated databases. We are currently studying the applicability of inconsistency-tolerant integrity checking for concurrent transactions in replicated databases.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. 18th PODS, pp. 68–79. ACM Press, New York (1999)
2. Chakravarthy, U.S., Grant, J., Minker, J.: Logic-based approach to semantic query optimization. *ACM Trans. on Database Syst (TODS)* 15(2), 162–207 (1990)
3. Christiansen, H., Martinenghi, D.: On simplification of database integrity constraints. *Fundam. Inform.* 71(4), 371–417 (2006)
4. Decker, H., Martinenghi, D.: Classifying Integrity Checking Methods with regard to Inconsistency Tolerance. In: Proc. 10th PPDP. ACM Press, New York (to appear, 2008)
5. Decker, H., Martinenghi, D.: A relaxed approach to integrity and inconsistency in databases. In: Hermann, M., Voronkov, A. (eds.) *LPAR 2006. LNCS (LNAI)*, vol. 4246, pp. 287–301. Springer, Heidelberg (2006)
6. Gupta, A., Sagiv, Y., Ullman, J.D., Widom, J.: Constraint checking with partial information. In: Proc. 13th PODS, pp. 45–55 (1994)
7. Martinenghi, D., Christiansen, H., Decker, H.: Integrity checking and maintenance in relational and deductive databases and beyond. In: Ma, Z. (ed.) *Intelligent Databases: Technologies and Applications*, pp. 238–285. Idea Group (2006)
8. Nicolas, J.-M.: Logic for improving integrity checking in relational data bases. *Acta Informatica* 18, 227–253 (1982)
9. Ramakrishnan, R., Gehrke, J.: *Database Management Systems*, 3rd edn. McGraw-Hill, New York (2003)

On the Evaluation of Large and Sparse Graph Reachability Queries

Yangjun Chen*

Department of Applied Computer Science
University of Winnipeg
Winnipeg, Manitoba, Canada R3B 2E9
ychen2@uwinnipeg.ca

Abstract. Given a directed graph G with n nodes and e edges, to check whether a node v is reachable from another node u through a path is often required. Such a problem is fundamental to numerous applications, including geographic navigation, Internet routing, ontology queries based on *RDF/OWL*, and metabolic network, as well as XML indexing. Among them, some involve huge but sparse graphs and require fast answering of reachability queries. In this paper, we propose a novel method called core labeling to handle reachability queries for massive, sparse graphs. The goal is to optimize both query time and labeling time. Our method consists of two schemes: Core-I and Core-II. For the Core-I labeling scheme, both the time and space requirements are bounded by $O(n + e + s \cdot b)$, where s is the number of the start nodes of all non-tree edges (edges that do not appear in the spanning tree of G); and b is the width of a subgraph of G . The size of that subgraph is bounded by $O(t)$, where t is the number of all the non-tree edges. The query time of Core-I is bounded by $O(\log b)$. The Core-II labeling scheme has constant query time, but the labeling time is increased to $O(n + e + s \cdot b \cdot \log b)$.

1 Introduction

Given two nodes u and v in a directed graph $G = (V, E)$, we want to know if there is path from u to v . The problem is known as *graph reachability*. In many applications (e.g., XML query processing), graph reachability is one of the most basic operations, and therefore needs to be efficiently supported. A naive method is to precompute the reachability between every pair of nodes – in other words, to compute and store the transitive closure (*TC* for short) of the graph. Then, a reachability query can be answered in constant time. However, this requires $O(n^2)$ space, which makes it impractical for massive graphs.

Recently, the interest in this problem is rekindled on large and sparse graphs for some important applications such as XML data processing, gene-regulatory networks or metabolic networks. It is well known that XML documents are often represented by tree structures. However, an XML document may contain *IDREF/ID* references that turn itself into a directed, but sparse graph: a tree structure plus a few reference links. For a metabolic network, the graph reachability models a relationship whether

* The author is supported by NSERC 239074-01 (242523) (Natural Sciences and Engineering Council of Canada).

two genes interact with each other or whether two proteins participate in a common pathway. Many such graphs are sparse.

In [7], Wang *et al.* discussed an interesting approach, called *Dual-I*, for sparse graphs. It assigns to each node v a dual label: (a_v, b_v) and (x_v, y_v, z_v) . In addition, a $t \times t$ matrix N (called a *TLC* matrix) is maintained, where t is the number of non-tree edges. Another node u with (a_u, b_u) and (x_u, y_u, z_u) is reachable from v iff $a_u \in [a_v, b_v]$, or $N(x_v, z_u) - N(y_v, z_u) > 0$. The size of all labels is bounded by $O(n + t^2)$ and can be produced in $O(n + e + t^3)$ time. The query time is $O(1)$. As a variant of *Dual-I*, one can also store N as a tree (called a *TLC* search tree), which can reduce the space overhead from a practical viewpoint, but increases the query time to $\log t$. This scheme is referred to as *Dual-II*.

In this paper, we propose two schemes: *Core-I* and *Core-II* to handle reachability queries for massive, sparse graphs. For the *Core-I* labeling scheme, both the time and space requirements are bounded by $O(n + e + s \cdot b)$, where s is the number of the start nodes of all non-tree edges (edges that do not appear in the spanning tree of G); and b is the width of a subgraph of G . The size of that subgraph is bounded by $O(t)$, where t is the number of all the non-tree edges. The query time of *Core-I* is bounded by $O(\log b)$. The *Core-II* labeling scheme has constant query time, but needs $O(n + e + s \cdot b \cdot \log b)$ labeling time. For sparse graphs, we assume $t \ll n$. In general, $s \leq t$.

The remainder of the paper is organized as follows. In Section 2, we discuss the concept of the *core* of a G . In Section 3, we describe our graph labeling schema. In Section 4, a short conclusion is set forth.

2 Tree Labeling and Core of G

In this section, we present our labeling approach. The input is a directed graph G with n nodes and e edges. We assume that it is acyclic. If not, we find all the *strongly connected components* (*SCCs*) of G and collapse each of them into a representative node. Obviously, each node in an *SCC* is equivalent to its representative node as far as reachability is concerned. This process takes $O(e)$ time using Tarjan's algorithm [6].

The main idea of our approach is a new tree structure generated for G , called the *core* of G , to do non-tree labeling. First, we define the core tree in 2.1. Then, in 2.2, we show our labeling scheme.

2.1 Tree Labeling

As with *Dual-I* labeling, we will first find a spanning tree T of G . This can be done by running the algorithm given in [7]. Then, we label T as follows.

By traversing T in *preorder*, each node v will obtain a number $pre(v)$ to record the order in which the nodes of the tree are visited. In a similar way, by traversing T in *postorder*, each node v will get another number $post(v)$. These two numbers can be used to characterize the ancestor-descendant relationships as follows.

Proposition 1. Let v and v' be two nodes of a tree T . Then, v' is a descendant of v iff $pre(v') > pre(v)$ and $post(v') < post(v)$.

Proof. See Exercise 2.3.2-20 in [5]. □

The following example helps for illustration.

Example 1. In Fig. 1(a), we show a graph G . We find a spanning tree T in G and represent all its edges by the solid arrows. The non-tree edges are represented by the dashed arrows. See the pairs associated with the nodes of T . The first element of each pair is the preorder number of the corresponding node and the second is its postorder number. With such labels, the ancestor-descendant relationships can be easily checked.

For instance, by checking the label associated with r against the label for g , we see that r is an ancestor of g in terms of Proposition 1. Note that r 's label is $(5, 7)$ and g 's label is $(8, 5)$, and we have $5 < 8$ and $7 > 5$. We also see that since the pairs associated with d and e do not satisfy the condition given in Proposition 1, d must not be an ancestor of e (with respect to T) and *vice versa*.

Let (p, q) and (p', q') be two pairs associated with nodes u and v . We say that (p, q) is subsumed by (p', q') , if $p > p'$ and $q < q'$. Then, u is a descendant of v if (p, q) is subsumed by (p', q') . In addition, if u and v are not on the same path in T , we have either $p < p'$ and $q < q'$, or $p' < p$ and $q' < q$. In the former case, we say, (p, q) is smaller than (p', q') , denoted $(p, q) \prec (p', q')$. In the latter case, (p, q) is said to be larger than (p', q') .

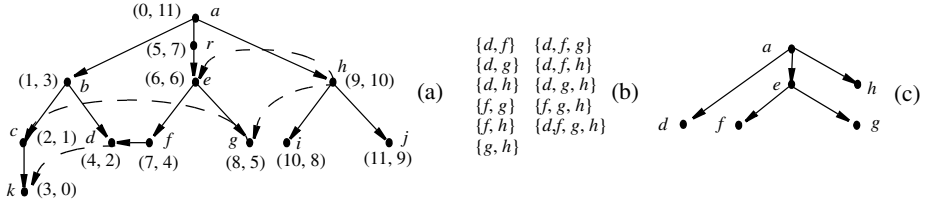


Fig. 1. Sparse graph, anti-subsuming subsets, and core of G

2.2 Core of G

In order to capture the reachability through non-tree edges, we generate and keep a new tree structure for the purpose of non-tree labeling.

Let T be a spanning tree of G . We denote by E' the set of all the non-tree edges. Denote V' the set of all the end points of the non-tree edges. Then, $V' = V_{start} \cup V_{end}$, where V_{start} stands for a set containing all the start nodes of the non-tree edges and V_{end} for all the end nodes of the non-tree edges.

Definition 1. (anti-subsuming subset) A subset $S \subseteq V_{start}$ is called an *anti-subsuming* set iff $|S| > 1$ and for any two nodes $u, v \in S$, labeled respectively with α and β , α is not subsumed by β , nor is β subsumed by α .

As an example, consider G and T shown in Fig. 1(a) once again. With respect to T , there are 5 non-tree edges: (d, k) , (f, d) , (g, c) , (h, e) , and (h, g) . Then, $V_{start} = \{d, f, g, h\}$. We have altogether 11 anti-subsuming sets as shown in Fig. 1(b).

Definition 2. (critical node) A node v in a spanning tree T of G is *critical* if $v \in V_{start}$ or there exists an anti-subsuming subset $S = \{v_1, v_2, \dots, v_k\}$ for $k \geq 2$ such that v is the

□

lowest common ancestor of $v_1, v_2, \dots, v_k$. We denote $V_{critical}$ the set of all critical nodes.

In the graph shown in Fig. 1(a), node e is the lowest common ancestor of $\{f, g\}$, and node a is the lowest common ancestor of $\{d, f, g, h\}$. So e and a are critical nodes. In addition, each $v \in V_{start}$ is a critical node. So all the critical nodes of G with respect to T are $\{d, f, g, h, e, a\}$.

Definition 3. (*core of G*) Let $G = (V, E)$ be a directed graph. Let T be a spanning tree of G . The *core* of G with respect to T is a tree structure with the node set being $V_{critical}$ and there is an edge from u to v ($u, v \in V_{critical}$) iff there is a path p from u to v in T and p contains no other critical nodes. The core of G with respect to T is denoted $G_{core} = (V_{core}, E_{core})$. $\square$

Example 2. Consider the graph G and the corresponding spanning tree T shown in Fig. 1(a). The core of G with respect to T is shown in Fig. 1(c). $\square$

In the past several decades, the problem of finding the lowest common ancestor of a given pair of nodes in a tree has attracted much attention, starting with Aho *et al.* [1]. The query can be answered in $O(1)$ time with $O(n)$ preprocessing time and space [2, 4]. Especially, the method discussed in [2] is based on the concept of *Euler tour* and can be quite easily implemented. However, if we use such algorithms to find all the critical nodes, we have to check all the possible anti-subsuming subsets and the worst-case time complexity is $O(2^s)$, where $s = |V_{start}|$.

For this reason, we devise an algorithm that explores T bottom-up to check only those anti-subsuming subsets which should be checked. The time complexity of this algorithm is bounded by $O(n)$.

The process can be described as follows.

Algorithm core-generation(T)

1. Mark any node in T , which belongs to V_{start} .
2. Let v be the first marked node encountered during the bottom-up searching of T . Create the first node for v in G_{core} .
3. Let u be the currently encountered node in T . Let u' be a node in T , for which a node in G_{core} is created just before u is met. Do (4) or (5), depending on whether u is a marked node or not.
4. If u is a marked node, then do the following.
 - (a) If u' is not a child (descendant) of u , create a link from u to u' , called a *left-sibling* link and denoted as $left-sibling(u) = u'$.
 - (b) If u' is a child (descendant) of u , we will first create a link from u' to u , called a *parent* link and denoted as $parent(u') = u$. Then, we will go along a left-sibling chain starting from u' until we meet a node u'' which is not a child (descendant) of u . For each encountered node w except u'' , set $parent(w) \leftarrow u$. Set $left-sibling(u) \leftarrow u''$. Remove $left-sibling(w)$ for each child w of u . (See Fig. 2 for illustration.)
5. If u is a non-marked node, then do the following.
 - (c) If u' is not a child (descendant) of u , no node will be created.
 - (d) If u' is a child (descendant) of u , we will go along a left-sibling chain starting from u' until we meet a node u'' which is not a child (descendant) of u . If the number of the nodes encountered during the chain navigation (not including

u'') is more than 1, we will create new node in G_{core} and do the same operation as (4.b). Otherwise, no node is created.

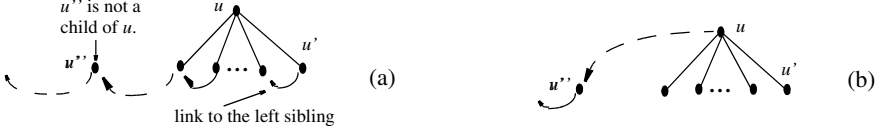


Fig. 2. Illustration for the construction of G_{core}

In Fig. 2(a), we show the navigation along a left-sibling chain starting from u' when we find that u' is a child (descendant) of u . This process stops whenever we meet u'' , a node that is not a child (descendant) of u . Fig. 2(b) shows that the left-sibling link of u is set to point to u'' , which is previously pointed to by the left-sibling link of u 's left-most child. In addition, all the left-sibling links of the child nodes of u are discarded since they will no longer be used.

Obviously, the algorithm requires only $O(n)$ time since each node in T is accessed at most two times.

Example 3. Consider the graph shown in Fig. 1(a). Applying the above algorithm to its spanning tree T represented by the solid arrows, we will generate a series of data structures shown in Fig. 3.

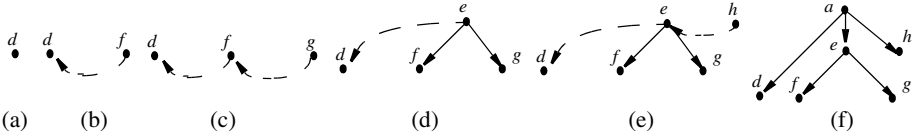


Fig. 3. Sample trace

First, the nodes d, f, g , and h in T are marked. During the bottom-up searching of T , the first node created for G_{core} is node d (see Fig. 3(a).) In a next step, node b is met. But node for G_{core} is created since b is not marked and has only one child (see 5.d in Algorithm *core-generation*()). In the third step, node f is encountered. It is a marked node and to the right of node d . So a link $left-sibling(f) = d$ is created (see Fig. 3(b).) In the fourth step, node g is encountered and another left-sibling link is generated (see Fig. 3(c).) In the fifth step, node e is met. It is not marked. But it is the parent of node g . So the left-sibling chain starting from node g will be searched to find all the children (descendants) of e along the chain, which appear in G_{core} . Furthermore, the number of such nodes is 2. Therefore, a node for e is created in G_{core} (see Fig. 3(d).) Here, special attention should be paid to the replacement of $left-sibling(f) = d$ with $left-sibling(e) = d$, which enable us to find easily the lowest common ancestor of d and some other critical nodes. In the next two steps, we will meet node i and j . But no nodes will be created for them. Fig. 3(e) and (f) demonstrate the last two steps of the whole process.

The following proposition shows the correctness of the algorithm.

Proposition 2. Let $G = (V, E)$ be a directed graph. Let T be a spanning tree of G . Algorithm *core-generation*() generates G_{core} of G with respect to T correctly.

Proof. To show the correctness of the algorithm, we should prove the following: (1) each node in G_{core} is a critical node; (2) any node not in G_{core} is not a critical node; (3) for each edge (u, v) in G_{core} there is a path from u to v in T .

First, we prove (1) by induction on the height h (the height of a node v in G_{core} is defined to be the longest path from v to a leaf node), at which a node in G_{core} appears.

Basis step. When $h = 0$, each leaf node in G_{core} is a node in V_{start} . So it is a critical node.

Induction hypothesis. Assume that every node appearing at height $h = k$ in G_{core} is a critical node. We prove that every node v at height $k + 1$ in G_{core} is also a critical node. If $v \in V_{start}$, v is a critical node by definition. Assume that $v \notin V_{start}$. According to the algorithm, v has at least two children (see 5.d in Algorithm *core-generation*()). If all its children belong to V_{start} , v is the lowest common ancestor of these child nodes. Assume that at least one of them does not belong to V_{start} . Let $v_1, \dots, v_i$ be the children of v in G_{core} . Assume that $v_1, \dots, v_j$ ($j \leq i$) are those not belong to V_{start} . Consider $v_1, \dots, v_j$ ($1 \leq r \leq j$). It must be at height $l \leq k$. According to the induction hypothesis, v_r is a critical node. Therefore, there exists an anti-subsuming subset S_r of V_{start} such that v_r is the lowest common ancestor of all the nodes in S_r ($1 \leq r \leq j$). Therefore, v is the lowest common ancestor of all the nodes in $S_1 \cup \dots \cup S_j \cup (\{v_1, \dots, v_i\} \setminus \{v_1, \dots, v_j\})$.

In order to prove (2), we notice that only in two cases no node is generated in G_{core} for a node $v \notin V_{start}$: (i) v is to the right of a node, for which a node in G_{core} is created just before v is encountered; (ii) v has only one child, for which a node in G_{core} is generated. Obviously, in both cases, v cannot be a critical node.

(3) can be seen from the fact that each *parent* link corresponds to a path in T and such a path cannot contain another critical node (except the two end points) since the nodes in T are checked level by level bottom-up. $\square$

In the following, we analyze the size of G_{core} .

First, we notice that the number of the leaf nodes in G_{core} is bounded by $|V_{start}|$. Secondly, for each critical node that does not belong to V_{start} has at least two children. Therefore, the total number of such nodes is bounded by the number of the leaf nodes in G_{core} , so bounded by $|V_{start}|$. Thus, the size of G_{core} cannot be beyond $2|V_{start}|$.

Proposition 3. The size of G_{core} is bounded by $O(|V_{start}|)$.

proof. See the above analysis. $\square$

3 Graph Labeling

In this subsection, we show our approach for graph labeling. The approach works in two steps. In the first step, we generate a data structure, called the *core label* (for G). It is in fact a set of pair sequences. In the second step, the core label is used to create non-tree labels for the nodes in G .

3.1 Core Labeling

Definition 4. (*link graph*) Let $G = (V, E)$ be a directed graph. Let T be a spanning tree of G . The *link graph* of G with respect to T is a graph, denoted G_{link} , with the node set being V' (the end points of all the non-tree edges) and the edge set $E' \cup E''$, where $(v, u) \in E''$ iff $v \in V_{end}$, $u \in V_{start}$, and there exists a path from v to u in T .

Example 3. The link graph of G shown in Fig. 4(a) with respect to the corresponding T is shown in Fig. 4(a).

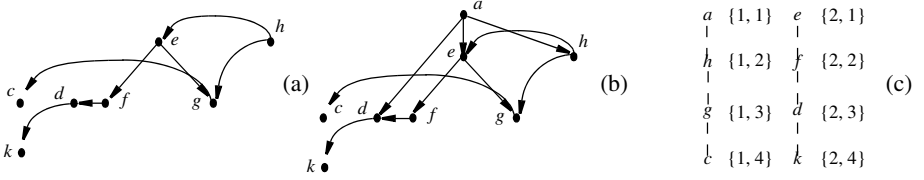


Fig. 4. Link graph, combined graph, and disjoint chains

As the first step to generate a core label for G , we will unite G_{core} and G_{link} to create a combined graph, denoted $G_{com} = G_{core} \cup G_{link}$, as shown in Fig. 4(b). In a next step, we will use the algorithm discussed in [4] to decompose G_{com} into a minimal set of disjoint chains as shown in Fig. 4(c). On each chain, if node v appears above node u , there is a path from v to u in G_{com} . Based on such a chain decomposition, we can assign each node in G_{com} an index as follows [4].

- (1) Number each chain and number each node on a chain.
- (2) The j th node on the i th chain will be assigned an index $\{i, j\}$ as its index.

In addition, each node v (which is not a leaf node in G_{com}) on the i th chain will be associated with an index sequence of length b : $L(v) = \{1, j_1\} \dots \{b, j_b\}$ such that for any node with index $\{x, y\}$ if $x = i$ and $y \geq j_x$ it is a descendant of v , where b is the number of the disjoint chains. Such index sequences make up the core label of G . (see Fig. 5(a) for illustration).

3.2 Non-tree Labeling: Core-I and Core-II

Based on the core label of G , we assign non-tree labels to nodes, which would enable us to answer reachability queries quickly.

Let v be a node in the spanning tree T of G . Consider the set of all the critical nodes in $T[v]$, denoted C_v . We denote v^* a critical node in C_v , which is closest to v . We further denote v^* the lowest ancestor of v (in T), which has a non-tree incoming edge.

Core label of G :

$s_1 L(a) = \{1, 1\} \{2, 1\}$	$\phi(a) = 1$
$s_2 L(h) = \{1, 2\} \{2, 1\}$	$\phi(h) = 2$
$s_3 L(e) = \{1, 3\} \{2, 1\}$	$\phi(e) = 3$
$s_4 L(f) = \{1, \_ \} \{2, 2\}$	$\phi(f) = 4$
$s_5 L(d) = \{1, \_ \} \{2, 3\}$	$\phi(d) = 5$
$s_6 L(g) = \{1, 3\} \{2, \_ \}$	$\phi(g) = 6$

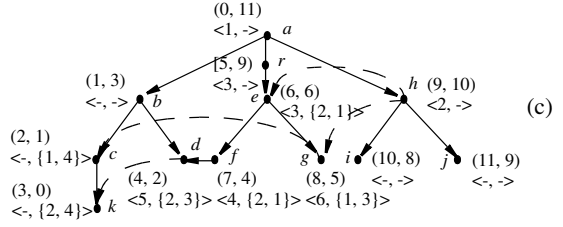


Fig. 5. Core label and non-tree labeling

The following two lemmas are critical to our non-tree labeling method.

Lemma 1. Any critical node in C_v appears in $T[v]$.

Proof. Assume that there exists a critical node u in C_v , which does not appear in $T[v]$. Let $C_v = \{u_1, \dots, u_k\}$ for some k . Consider the lowest common ancestor node of $u, u_1, \dots, u_k$. It must be an ancestor node of $\bar{v}$, which is closer to v than $\bar{v}$, contradicting the fact that $\bar{v}$ is the closest critical node (in $T[v]$) to v . $\square$

Lemma 2. Let u be a node, which is not an ancestor of v in T ; but v is reachable from u via some non-tree edges. Then, the only way for u to reach v is through v^* .

Proof. This can be seen from the fact that any node which reaches v via some non-tree edges is through v^* to reach v [16]. $\square$

Let $V_{core} = \{v_1, \dots, v_j\}$. We store the core label of G as a list: $s_1 = L(v_1), \dots, s_j = L(v_j)$ (see Fig. 12(a)). Then, we define a function $\phi: V_{core} \rightarrow \{1, \dots, j\}$ such that for each $v \in V_{core}$ $\phi(v) = i$ iff $s_i = L(v)$. Based on the above concepts, we define Core-I below.

Definition 5 (Core-I) Let v be a node in G . The non-tree label of v is a pair $\langle \delta, \tau \rangle$, where

- $\delta = i$ if $\bar{v}$ exists and $\phi(\bar{v}) = i$. If $\bar{v}$ does not exist, let δ be the special symbol “-”.
- $\tau = \{x, y\}$ if v^* exists and $\{x, y\}$ is the index of v^* . If v^* does not exist, let y be “-”. $\square$

Example 3. Consider G and T shown in Fig. 3(a). The core label of G with respect to T is shown in Fig. 5(a). The values of the corresponding ϕ -function are shown in Fig. 5(b).

Fig. 5(c) shows the tree labels and the non-tree labels. For instance, the non-tree label of node r is $\langle 3, \_ \rangle$ because (1) $r^- = e$; (2) $\phi(r^-) = \phi(e) = 3$; and (3) r^* does not exist. Similarly, the non-tree label of node f is $\langle 4, \{2, 1\} \rangle$. Special attention should be paid the non-tree label of node e : $\langle 3, \{2, 1\} \rangle$. First, we note that e^- is e itself. So $\phi(e^-) = \phi(e) = 3$. Furthermore, e^* is also e itself. Therefore, the tree label of e^* is in fact the index of e . $\square$

Proposition 5. Assume that u and v are two nodes in G , labeled $((a_1, b_1), \langle \delta_1, \tau_1 \rangle)$ and $((a_2, b_2), \langle \delta_2, \tau_2 \rangle)$, respectively. Node v is reachable from u iff one of the following conditions holds:

- (i) (a_2, b_2) is subsumed by (a_1, b_1) , or
- (ii) There exists an index $\{x, y\}$ in such that for $\tau_2 = \{x', y'\}$ we have $x = x'$ and $y \leq y'$.

Proof. The proposition can be derived from the following two facts:

- (1) v is reachable from u through tree edges iff (a_2, b_2) is subsumed by (a_1, b_1) .
- (2) In terms of Lemma 6 and Lemma 7, v is reachable from u via non-tree edges iff both u^- and v^* exist and the index sequence of u^- contains an index $\{x, y\}$ such that for the index $\{x', y'\}$ of v^* we have $x = x'$ and $y \leq y'$. $\square$

Proposition 6. Let v and u be two nodes in G . It needs $O(\log b)$ time to check whether u is reachable from v via non-tree edges or *vice versa*.

Proof. The proposition is derived from the fact that each index sequence in the core label of G is sorted and its length is bounded by b . $\square$

As a variant of Core-I, we can store the core label of G as a matrix M . Number the nodes in G_{com} with $1, 2, \dots, |G_{com}|$. For any $i, j \in G_{com}$, let $\{x_i, y_i\}$ be the index of i and $L(j)$ be the index sequence of j . If there exists an index $\{x, y\}$ in $L(j)$ such that $x = x_i$ and $y \leq y_i$, then set $M(j, i) = 1$. Otherwise, $M(j, i) = 0$. We refer this scheme as *Core-II*. Obviously, we need $O(s \cdot b \cdot \log b)$ time to establish M . But the query time is reduced to $O(1)$.

4 Conclusion

In this paper, we have discussed a new method to handle reachability queries for massive, sparse graphs. The goal is to optimize both query time and labeling time. Our method consists of two schemes: Core-I and Core-II. For the Core-I labeling scheme, both the time and space requirements are bounded by $O(n + e + s \cdot b)$, where s is the number of the start nodes of all non-tree edges (edges that do not appear in the spanning tree of G); and b is the width of a subgraph of G . The size of that subgraph is bounded by $O(t)$, where t is the number of all the non-tree edges. The query time of Core-I is bounded by $O(\log b)$. The Core-II labeling scheme has constant query time, but the labeling time is increased to $O(n + e + s \cdot b \cdot \log b)$.

References

- [1] Aho, A.V., Hopcroft, J.E., Ullman, J.D.: On finding lowset common ancestors in trees. *SIAM J. Comput.* 5(1), 115–132 (1976)
- [2] Bender, M.A., Farach-Colton, M.: The LCA Problem Revisited. In: Gonnet, G.H., Viola, A. (eds.) *LATIN 2000*. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000)
- [3] Chen, Y.: On the decomposition of DAGs into disjoint chains. In: *Proc. of 18th Int. DEXA Conf. on Database and Expert Systems Application*, September 2007. LNCS, vol. 4653, pp. 243–253. Springer, Heidelberg (2007)
- [4] Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* 13, 338–355 (1984)
- [5] Knuth, D.E.: *The Art of Computer Programming*, vol. 1. Addison-Wesley, Reading (1969)
- [6] Tarjan, R.: Depth-first Search and Linear Graph Algorithms. *SIAM J. Comput.* 1(2), 140–146 (1972)
- [7] Wang, H., He, H., Yang, J., Yu, P.S., Yu, J.X.: Dual Labeling: Answering Graph Reachability Queries in Constant time. In: *Proc. of Int. Conf. on Data Engineering*, Atlanta, USA, April 8, 2006 (2006)

SQL TVF Controlling Forms - Express Structured Parallel Data Intensive Computing

Qiming Chen and Meichun Hsu

HP Labs
Palo Alto, California, USA
Hewlett Packard Co.
{qiming.chen,meichun.hsu}@hp.com

Abstract. A key issue in supporting the synthesis of data intensive computation and data management is to liberate users from low-level parallel programming, by specifying applications functionally independent of the underlying server infrastructure, and further, by providing high-level primitives to express the control flow of applying functions to data partitions. Currently only few such primitives, e.g. Map-Reduce and Cross-Apply, are available, and their expressive power is limited to “flat parallel computing”. To deal with “structured parallel computing” where a function is applied to multiple objects with execution order dependencies, a general framework for creating and combining such primitives is required.

We propose the SQL-FCF framework as the database centric solution to the above problem. We embed into SQL queries the Function Controlling Forms (FCFs) to specify the flow control of applying Table Valued Functions (TVFs) to multiple data partitions. We further support the extensibility of this framework by allowing new FCFs to be defined from existing ones with SQL phrases. Based on this approach, we provided a SQL based high-level interface for “structured parallel computing” in architecting a hydrologic scientific computation platform. Envisioning that the simple parallel computing primitives will evolve and form a general framework, our effort is a step towards that goal.

1 Introduction

Data Intensive Computation is the technical merge of parallel computation and scalable data management [1,3,6,7,14-19]. When the support to parallel computing has spread over multiple system layers, to liberate users from low-level parallel programming has become a pressing need.

The basic way for lifting parallel programming interface to a higher-level is to specify computation job functionally without mentioning “addresses”, such as memory, CPUs, and cluster nodes, leaving computation parallelization to a system layer. To specify how functions are applied to data partitions, the notion of Function Controlling Forms (FCF) is required. Simply speaking, a FCF is a meta-operator that takes functions as parameter objects to form a new function. In fact the Map operator in Map-Reduce [4] may be viewed as a simple FCF as

$$(\text{Map } f): \langle x_1, \dots, x_n \rangle = \langle f:x_1, \dots, f:x_n \rangle.$$

It expresses applying function f to all objects concurrently. This notion is brought in from functional programming [2,10,11]. However, we are not dealing with a formal

symbolic programming system. Instead, our effort is motivated by extending SQL for expressing the control flow of applying TVFs that have effects on database states. Along this line there exist some informal approaches where MapReduce [4,12] and CROSS APPLY (Sql Server) [5] only deal with flat data, CONNECT BY (Oracle) has determinate semantics only for data organized in a hierarchy rather than in a graph.

In order to cover richer applications, it is necessary to provide an extended set of FCFs for controlling the applications of a function to multiple objects based on certain execution order dependencies, and further, to allow the FCF set itself extensible.

We have developed a novel framework, SQL-FCF, for interfacing data intensive computation and parallel database management at a high-level. The context and contributions of our research can be outlined as below.

- Motivated by providing database centric solutions, we push down applications to databases for reduced data movement. We wrap them as a kind of User Defined Functions (UDFs), Table Valued Functions (TVFs). The input of a TVF is a row that represents a data partition directly or referentially, and the output is a row set. In addition to conveying an output row set, a TVF may have database update effects, leading to the order sensitivity of its executions. Thus we introduce FCFs on TVFs to deal with the control flow of applying TVFs to data partitions.
- In order to support non-flat but structured parallel computing, we define FCFs to specify the order dependency of applying TVFs to data partitions.
- We make the SQL-FCF framework extensible by providing the mechanisms for defining new FCFs from existing ones with SQL phrases.

We have applied this approach to architect a hydro-informatics system for supporting data intensive applications on a database cluster. The hydrologic applications are wrapped as TVFs; data are partitioned, and TVFs are made available over multiple server nodes. A TVF is applied to the data partitions representing geographic regions in certain order. Task management is handled at a system layer that interprets SQL-FCF, parallelizes computing whenever possible, and invokes TVFs. The SQL-FCF framework allows users to specify the order dependency of computations at a high-level, leaving the parallel computing opportunities, either static implied in the specification or dynamically appeared during the execution, interpreted and handled by the system.

The rest of this paper is organized as follows: Section 2 provides an example on data dependent structured parallel computation; Section 3 discusses the proposed SQL FCF framework; Section 4 concludes.

2 Introductory Example on Structured Parallel Computing

In this section we explain the need for handling structured parallel computing by the watershed observation application. A watershed observation function is designed for a kind of data intensive hydrologic computation, which is periodically applied to a river drainage network, for hydrologic monitoring, feature extraction and prediction.

The data about a river drainage network are location sensitive geographic information divided by regions. Multiple regions are modeled as a tree where recursively a

parent node represents a downstream region, and a child node represents an upstream region. The data are stored in the *regions* table with attributes *region_id*, *region_level*, *parent_region_id*, and *region data*. The *regions* table is partitioned over multiple server nodes to be accessed in parallel whenever possible. The data partitioning scheme is visible to all server nodes (actually replicated).

The *region_level* of a region represents its level of partitioning, as the length of its longest descendant path counted bottom-up from the leaves of the regions tree. Figure 1 shows the hierarchical organization of regions and their levels of partitioning.

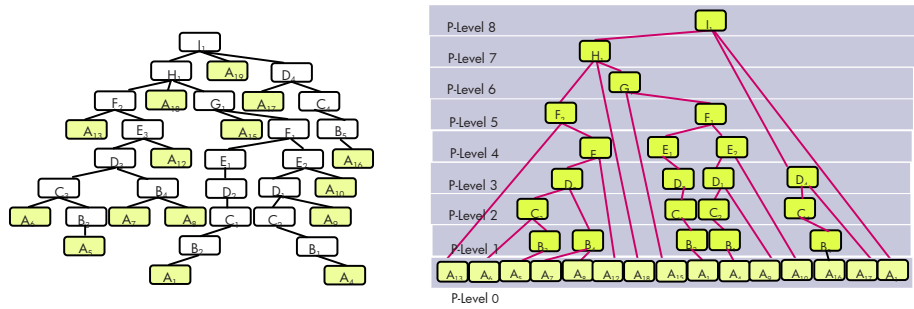


Fig. 1. River drainage modeled as a regions tree with levels of data partition, computation must be made bottom-up – parallelism opportunity exists but not a single step of “apply to all”

The watershed observation function is implemented in the following way.

- It is coded as a user defined TVF, whose execution can be pushed down to the database engine for eliminating the data transfer round trip.
- The TVF, say f , is defined on a river region to conduct a computation job on that region. The input of f is a *region_id*, but a small amount of information of the upstream regions is also required for the computation. The output of f is some information as a table or row set, but f also has effects on database updates which will affect the successive computations.
- The same TVF is made available to all the participating server nodes.
- The watershed observation function is applied region by region from upstream to downstream, i.e. the region tree is *post-order traversed* - the root is visited *last*.
- The parallel computation opportunities exist *statically* in processing the non-conflict regions at the same level, or the regions without a common parent. Parallel computations opportunities also exist *dynamically* in processing the regions with all their children regions have been processed. These two kinds of opportunities will be interpreted and realized at a system layer. Data communication is made through database access.

The procedure given in Figure 2 shows the invocation of a function, Watershed Analysis, in the post-order. Consider *post-order tree processing* as a “meta operator”, **postorder**, for applying *any* function f that processes tree nodes in the post order. Together, (**postorder** f) denotes a function, where f , that is consider as the operand

Function PostorderAnalysis (t)
Input: Tree t
Output: watershed current/predicted status
Procedure
 1: if $t = \emptyset$ then
 2: return
 3: for each i
 4: PostorderAnalysis ($t.\text{child}(i)$)
 5: WatershedAnalysis (t)

Fig. 2. A post-order processing function

Function (postorder f) (t)
Input: Tree t
Output: watershed current/predicted status
Procedure
 1: if $t = \emptyset$ then
 2: return
 3: for each i
 4: (postorder f) ($t.\text{child}(i)$)
 5: $f(t)$

Fig. 3. A “meta operator” takes function f as its parameter for applying f in post-order

of **postorder**, can be substituted by a function, e.g. WatershedAnalysis. This is illustrated in Figure 3. This shows how a meta-operator controls the way of applying a function. We will investigate how to express such function control operators in SQL.

3 The Proposed SQL-FCF Framework

Motivated by providing a SQL based high-level interface to support database-based data intensive computation, we propose the SQL-FCF framework, which is characterized by the following.

- Certain procedure semantics is embedded into SQL query statements for expressing how TVFs are applied, in addition to what data are returned.
- It focuses on “structured parallel computation” in terms of SQL based FCFs.
- The SQL-FCF framework is extensible where new FCFs can be derived from existing ones using SQL constructs.

3.1 SQL Framework for User Defined Functions

SQL Framework and its Changeable Parts. Let us distinguish two parts of the SQL language. First, its framework which gives its overall rules and operators: select, join, project (SJP), order, sort, etc, and second, its changeable parts, whose existence is anticipated by the framework but whose particular behavior is not specified by it. For example, the SJP related phrases are part of the SQL framework, but the UDFs are its changeable parts. Thus the SQL language describes its fixed features and provides a general environment for its changeable features.

Changeable Parts and Controlling Forms. We envisage that the most important enhancement to the changeable parts in SQL language is the availability of controlling forms that can be generally used to build new functionalities from existing UDFs.

So far the only notable SQL language construct capable of affect the applying of TVFs, a kind of UDFs, is the CROSS APPLY (plus OUTER APPLY) operator provided in Transact-SQL [5]; it expresses the way to apply a TVF to the rows of a table and then union the results.

Although CROSS APPLY is more powerful than Map-Reduce in representing the “apply-to-all” semantics, in that the input object can be filtered, and output results can

be further manipulated by SQL phrases, it is the only primitive for controlling TVFs, and it actually does not indicate the order dependencies in applying TVF to multiple objects.

Fixed and Extensible Controlling Forms. As we have seen, a SQL framework has a set of primitive functions, plus a possible set of UDFs. In case the set of UDF controlling forms, FCFs, are empty, then the behavior of those UDFs is not extensible; if the set of FCFs is not empty but fixed once and for all, then the flexibility of changing the behavior of UDF applications is still limited. In fact, the set of FCFs determines the capability of the SQL framework for embedding applications in a major way. To make such capability extensible, further allowing the creation of new FCFs from existing ones with SQL phrases, is essential.

3.2 A SQL-FCF Example

The following SQL-FCF example illustrates the kind of hydrologic computation jobs applied region by region along a geographic region tree in the post-order, which represents the general feature of a class of similar scientific computations. The SQL-FCF controls the “structured” flow of applying a TVF, f , from upstream regions to downstream regions, i.e. bottom-up along the region tree represented in table “regions”

```
CONNECT APPLY  $f$ (region_id) TO regions
  BY region_id = PRIOR ALL parent_region_id
  START WITH region_level = 0;
```

This phrase does return a row set, however, different from a recursive query, it also indicates the step of TVF executions, as

- the processing starts from the leaf regions at region_level 0;
- the processing is bottom-up, where f is fired on a parent region (in downstream) after all its child regions (in upstream) have been processed;
- stepwise parallel computing opportunities are implied for non-dependent function applications.

The unary operator, PRIOR, has the same precedence as the unary + and - arithmetic operators. It evaluates the immediately following expression for matching the parent_region_id of ALL *child* rows with the region_id of the *current* row, in the order from children to parent. ALL is required as a parent can have multiple children. On TVF executions, the above phrase dictates the post-order traversal order dependency of applying f to multiple regions; on resulting data, it return the transitive closure of the data hierarchy, plus other designed database effects of f .

Note that the semantics, syntax as well as implementation of this FCF are different from Oracle’s CONNECT BY. Our implementation ensures the order of TVF applications, and a TVF is applied to each data partition (tree node) only once. Similarly we also changed the SQL syntax for CROSS APPLY as shown later.

3.3 Core FCFs

A FCF, viewed as a “meta operator”, takes one or more TVFs as its parameters, applying a FCF to the parameter TVFs, denotes a new function.

As samples, below we discuss two core FCFs. Note that this sample set is non-exclusive, and the selection of them is based on their suitability to SQL semantics.

We start with the existing CROSS APPLY with syntactical alteration. For a TVF, f , defined on row set R , CROSS APPLY f to R means applying f to all the rows of R , without any constraint on the applying order; the union of the resulting row sets of applying f to each row of R , is joined to R as the return row set.

[CROSS APPLY] CROSS APPLY a TVF, f , to a row set R is denoted by $\alpha f_{\kappa} : R$ that returns a row set. α denotes the FCF, CROSS APPLY; κ denotes the attribute name list, say $A_1, \dots, A_j$, of R whose values are taken as the function input; f_{κ} maps a row in R to a row set. The procedure semantics of this FCF is illustrated in Figure 4.

[CROSS APPLY] $\alpha f_{\kappa} : R$ Procedure 1: Set $S = \emptyset$	2: for each $i \in \{1, \dots, n\} \wedge t_i \in R$ 3: $S = S \cup f_{\kappa}(t_i)$ 4: return eq-join (R, S) on κ

Fig. 4. Procedure semantics of CROSS APPLY

The next FCF – CONNECT APPLY, is introduced for applying a TVF along the data object hierarchy with order dependency (e.g. due to the TVF's effects on database states). The data objects (or their Ids) to be processed by the TVF are organized in the tree structure, and stored in a table with a pair of parent-child attributes P and C. Accordingly we represent the *parent-child ordering* as $\langle P, C \rangle$.

Controlled by CONNECT APPLY, the executions start from applying the TVF to the rows selected on a given condition, the order of tree traversal – in pre-order (top-down) or in post-order (bottom-up), is indicated by another input parameter.

[CONNECT-APPLY] $\gamma_{\phi, P, C, O} f_{\kappa} : R$ Procedure 1: Set $S = \emptyset$ 2: for each row $t \in \sigma_{\phi}(R)$ 3: if $O == 0$ // $O == 0$ indicates pre-order apply // where the function is applied to // the current row first before applied to // its children <i>recursively</i> . // Otherwise, in post-order apply, // the order is reversed	4: then // first apply to the current row t 5: $S = S \cup f_{\kappa}(t)$ // then to child rows in $\sigma_{P=t, C}(R)$ 6: $S = S \cup \gamma_{P=t, C, P, C, O} f_{\kappa}(R)$ 7: else // first apply to child rows 8: $S = S \cup \gamma_{P=t, C, P, C, O} f_{\kappa}(R)$ // then to the current row t 9: $S = S \cup f_{\kappa}(t)$ 10: return eq-join (R, S) on κ

Fig. 5. Procedure semantics of CONNECT APPLY

This FCF specifies both the steps of applying TVF and the returned data, which is the join of the input table and the union of the results from stepwise function applications.

[CONNECT APPLY] CONNECT APPLY a TVF f_K to a row set R is denoted by $\gamma_{\phi, P, C, o} f_K : R$ where γ stands for the FCF; K for the input attributes of f ; ϕ is a condition for selecting the rows in R , i.e. $\sigma_{\phi}(R)$, to start with. Attributes P and C are the parent-child pair on R , underlying the “connect” condition $t_{parent}.C = t_{child}.P$. The ordering of applying is represented by $O = \{0, 1\}$ with 0 for “pre-order” and 1 for “post-order”. The procedure semantics is given in Figure 5.

3.4 Derive New SQL-FCFs from Existing Ones

While just a few FCFs already offer significant flexibility of TVF utilization, the expressive power of SQL-FCF can be further enhanced by deriving new FCFs from existing ones, in the following two major ways:

- specialize a FCF, and
- combine FCFs.

Specialization. A new FCF can be simply derived from an existing one, say F , by instantiating certain parameters of F . Below is the example of specializing CONNECT APPLY to PREORDER APPLY and POSTORDER APPLY.

[PREORDER APPLY, POSTORDER APPLY]

In CONNECT APPLY $\gamma_{\phi, P, C, o} f_K$, the apply ordering parameter $O = \{0, 1\}$ with 0 for “pre-order” and 1 for “post-order”, can be specialized such that

- $\gamma_{\phi, P, C, 0} f_K$ represents PREORDER APPLY, and
- $\gamma_{\phi, P, C, 1} f_K$ represents POSTORDER APPLY.

These can be illustrated by the following SQL examples.

```
PREORDER APPLY f(region_id) TO regions
  BY parent_region_id, region_id
  START WITH region_level = MAX(region_level);

POSTORDER APPLY f(region_id) TO regions
  BY parent_region_id, region_id START WITH region_level = 0;
```

Interpreted by the above semantics, they are equivalent to

```
CONNECT APPLY f(region_id) TO regions
  BY PRIOR region_id = parent_region_id
  START WITH region_level = MAX(region_level);

CONNECT APPLY f(region_id) TO regions
  BY region_id = PRIOR ALL parent_region_id START WITH region_level = 0;
```

respectively.

Combination. A FCF can be defined from existing FCFs in the SQL framework that can also be viewed as a special kind of parameterized query.

For instance, a new FCF, CONC-CROSS APPLY, can be defined by joining the results of two concurrent cross apply operations, with the following semantics

[CONC-CROSS APPLY]

- Let αf_K and αg_K be two CROSS APPLY functions defined on R where f_K and g_K have the same input attribute list K on R , such as $A_i, \dots, A_j$.
- Let the join two row sets R_1, R_2 on attributes K (i.e. on condition $R_1.A_i = R_2.A_i \wedge \dots \wedge R_1.A_j = R_2.A_j$) be represented by $J_K(R_1, R_2)$.
- CONC-CROSS APPLY αf_K and αg_K to R means $J_K(\alpha f_K : R, \alpha g_K : R)$.

Then in SQL-like syntax, for example,

```

CONC-CROSS APPLY [f(region_id), g(region_id)] TO regions
means
(CROSS APPLY f(region_id) TO regions) a JOIN
(CROSS APPLY g(region_id) TO regions) b ON a. region_id = b. region_id

```

4 Concluding Remarks

This research is driven by the synthesis of data intensive computation and data management [7,9,13,14], where one aspect for seamlessly integrating them is to provide high-level parallel computing interface functionally independent of the underlying cluster infrastructure. Map-Reduce and Cross-Apply primitives are developed along this direction, but need to evolve into more general ones and to form a framework for dealing with “structured parallel computing”, not just “flat parallel computing”.

Motivated by providing a database centric solution, we pushed down data intensive computation to the database layer for reduced data traffic, by wrapping computation jobs as UDFs/TVFs, and study how to control their applications to data objects.

Up to now, the expressive power of SQL for specifying applications, in the major way, depends on its set of UDFs. In a SQL dialect without the CROSS APPLY primitive, its entire set of non-system functions are just the set of individual UDFs. By supporting CROSS APPLY, the expressive power of T-SQL in utilizing TVFs is expanded, but only limited to the case of applying a TVF to all objects in one shot. In fact, as a “flat parallel computing” primitive, CROSS APPLY does not really require the query processor to concern about the order of applying a TVF to the input rows.

In this work we have introduced the general notion of TVF Controlling Forms (FCFs) into the SQL framework. This effort represents an initial step to systematically and imperatively embed certain flow control on UDF applications into SQL query statements (rather than scripts), in a way integrated with the data flows in query processing. With this extension, applying TVFs can be controlled in a way consistent with the dependency of data processing, such as the post-order tree traversal illustrated in this report, implied by the nature of the applications. Specifically, our contributions include the following.

- We proposed the notion of FCF and embedded the corresponding procedural semantics to SQL, as a database centric solution to dealing with data dependent, structured parallel computing.
- We provided a sample set of core FCFs for controlling the operational flow of applying TVFs to data partitions.
- We developed the mechanisms for FCF extensibility, allowing new FCFs to be defined from existing ones with SQL phrases.

Defining FCFs in the extended SQL framework complies with our vision on database oriented data intensive computation. SQL-FCF is currently defined at a virtual system layer. The interpretation of the “active SQL” statements involving FCF operators, together with the rules for task parallelization, is handled by a virtual system that “thinks in parallel”. This virtual system is built on top of DBMS but works on the operational environment provided by the DBMS - whose data are stored in the underlying database, and whose operators are implemented as UDFs (or stored procedures) whenever possible.

This work also opens several directions to future research, in the synthesis of scientific computing and database technology, in the synthesis of file based parallel computing model and database based one, and in the synthesis of handling imperative operation flows and declarative query results - towards a computational query framework.

References

1. Asanovic, K., Bodik, R., Catanzo, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., Yelick, K.A.: The landscape of parallel computing research: A view from Berkeley, Technical Report UCB/EECS-2006-183, U. C., Berkeley, December 18 (2006)
2. Backus, J.: Can Programming be Liberated from the von Neumann Style? A functional style and its algebra of programs. *CACM* 21(8) (1978)
3. Barclay, T., Gray, J., Chong, W.: TerraServer Bricks – A High Availability Cluster Alternative, Technical Report, MSR-TR-2004-107 (October 2004)
4. Barroso, L.A., Dean, J., Hölze, U.: Web search for a planet: The Google cluster architecture. *IEEE Micro* 23(2), 22–28 (2003)
5. Ben-gan, I., et al.: Inside Microsoft SQL Server 2005: T-SQL Programming (2006)
6. Bryant, R.E.: Data-Intensive Supercomputing: The case for DISC, CMU-CS-07-128 (2007)
7. Chen, Q., Hsu, M., Dayal, U.: A Data-Warehouse/OLAP Framework for Scalable Telecommunication Tandem Traffic Analysis. In: *ICDE 2000*, pp. 201–210 (2000)
8. Chen, Q., Hsu, M.: Inter-Enterprise Collaborative Business Process Management. In: *Proc. of 17th Int’l Conf. on Data Engineering (ICDE-2001)*, Germany (2001)
9. Chen, Q., Dayal, U., Hsu, M.: A Distributed OLAP Infrastructure for E-Commerce. In: *Proc. Fourth IFCIS CoopIS Conference*, UK (1999)
10. Chen, Q., Kambayashi, Y.: Nested Relation Based Database Knowledge Representation. In: *ACM-SIGMOD Conference*, pp. 328–337 (1991)
11. Chen, Q., Gardarin, G.: An Implementation Model for Reasoning with Complex Objects. In: *ACM-SIGMOD Conference*, pp. 164–172 (1988)
12. Dean, J.: Experiences with MapReduce, an abstraction for large-scale computation. In: *International Conference on Parallel Architecture and Compilation Techniques*. ACM, New York (2006)

13. DeWitt, D., Gray, J.: Parallel Database Systems: the Future of High Performance Database Systems. CACM 35(6) (June 1992)
14. Gray, J., Liu, D.T., Nieto-Santisteban, M.A., Szalay, A.S., Heber, G., DeWitt, D.: Scientific Data Management in the Coming Decade. SIGMOD Record 34(4) (2005)
15. Hsu, M., Xiong, Y.: Building a Scalable Web Query System. In: Bhalla, S. (ed.) DNIS 2007. LNCS, vol. 4777, Springer, Heidelberg (2007)
16. HP Neoview enterprise datawarehousing platform, <http://h71028.www7.hp.com/ERC/downloads/4AA0-7932ENW.pdf>
17. Netz, A., Chaudhuri, S., Bernhardt, J., Fayyad, U.: Integration of data mining and relational databases. In: Proceeding of the 26th Conference on Very Large Databases, pp. 719–722 (2000)
18. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: VLDB (2008)
19. Saarenvirta, G.: Operational Data Mining, DB2 Magazine 6 (2001)

A Decidable Fuzzy Description Logic $F\text{-}ALC(G)$

Hailong Wang and Z.M. Ma

School of Information Science & Engineering, Northeastern University
Shenyang 110004, China
zongmin_ma@yahoo.com

Abstract. The existing Web Ontology languages and the corresponding description logics don't support customized data type information. Furthermore, they can't process imprecise and uncertain information in the Semantic Web. In order to solve these limitations, we propose a new kind of fuzzy description logic, $F\text{-}ALC(G)$, which can not only support the representation and reasoning of fuzzy concept knowledge, but also support fuzzy data information with customized fuzzy data types and customized fuzzy data type predicates.

Keywords: fuzzy $ALC(G)$; customized fuzzy data types; Tableau algorithm.

1 Introduction

In real applications, there are a big deal of imprecision and uncertainty. So the fuzzy extensions to description logics (DLs) which can process fuzzy ontology are necessary. In fact, much work has been carried out towards combining fuzzy logic [1] and DLs [2] during the last decade. The existing extensions [3], [4] can only process fuzzy concept knowledge while cannot cover the representation and reasoning of fuzzy data information. The Semantic Web is expected to process the fuzzy data information in an intelligent way. Data type support [5] is one of the most useful features that OWL is expected to provide. As an attempt to process fuzzy data information, Straccia proposes $f\text{-}SHOIN(D)$ in [6], but there is no reasoning algorithm for it, and it cannot support customized fuzzy data types and predicates.

To overcome the above limitations of the existing DLs , this paper presents a fuzzy DL $F\text{-}ALC(G)$ which can support fuzzy data information with customized fuzzy data types and predicates. Furthermore, this paper gives an complete tableau algorithm for $F\text{-}ALC(G)$ and proposes a reasoning architecture for fuzzy data type reasoning. Also the decidability of the tableau algorithm is discussed.

2 Syntax and Semantics of $F\text{-}ALC(G)$

Definition 1. A data type [7] is characterized by a lexical space, $L(d)$, which is a non-empty set of Unicode strings; a value space, $V(d)$, which is a nonempty set, and a total mapping $L2V(d)$ from the lexical space to value space.

Definition 2. A base data type d is a special kind of data type which is built-in in XML Schema, such as *xsd: integer* and *xsd: string* etc.

Definition 3. A *data type predicate* p is used to constrain and limit the corresponding data types. It is characterized by an arity $a(p)$.

Definition 4. The semantic of a *fuzzy data type predicate* p can be given by $(\Delta_D, \bullet^D)$, where Δ_D is the fuzzy data type interpretation domain and $\bullet^D$ is the fuzzy data type interpretation function. Here $\bullet^D$ maps each n -ary fuzzy predicate p to a function $p^D: \Delta_D^n \rightarrow [0, 1]$ which is an n -ary fuzzy relation over Δ_D . It means that the relationship of concrete variables $v_1, \dots, v_n$ satisfies predicate p in a degree belonging to $[0, 1]$. We also use $a(p)$ to represent the arity of fuzzy predicate p .

Definition 5. A *fuzzy data type group* G is a triple (ϕ_G, D_G, dom) , where ϕ_G is a set of predicates which have been defined by corresponding known predicate URIs, D_G is a set of base data types, and $dom(p)$ is the domain of a fuzzy data type predicate,

$$dom(p) = \begin{cases} p & \text{if } p \in D_G \\ (d_1, \dots, d_n), \text{ where } d_1, \dots, d_n \in D_G & \text{if } p \in \phi_G \setminus D_G \text{ and } a(p) = n \end{cases}$$

Definition 6. Given a fuzzy data type group $G = (\phi_G, D_G, dom)$ and a base data type $d \in D_G$, the *sub-group* of d in G , abbreviated as *sub-group* (d, G) , is defined as:

$$sub\text{-}group(d, G) = \{p \in \phi_G \mid dom(p) = \{d, \dots, d\} \text{ (} a(p) \text{ times)}\}$$

Definition 7. Let G be a fuzzy data type group. Valid *fuzzy G -data type expression* E is defined by the following abstract syntax:

$$E ::= \top_D \mid \bot_D \mid p \mid \bar{p} \mid \{l_1, \dots, l_s\} \mid (E_1 \wedge \dots \wedge E_s) \mid (E_1 \vee \dots \vee E_s) \mid [u_1, \dots, u_s]$$

Here $\top_D$ represents the fuzzy top data type predicate while $\bot_D$ the fuzzy bottom data type predicate; p is a predicate URIref; $\bar{p}$ is the relativized negated form of p ; $l_1, \dots, l_s$ are typed literals; u_i is a unary predicate in the form of $\top_D$, $\bot_D$, p , or $\bar{p}$. The semantic of fuzzy data type expressions can be defined as follows:

- 1) $\top_D^D(v_1, \dots, v_n) = 1$;
- 2) $\bot_D^D(v_1, \dots, v_n) = 0$
- 3) $p^D(v_1, \dots, v_n) \rightarrow [0, 1]$;
- 4) $\bar{p}^D(v_1, \dots, v_n) = 1 - p^D(v_1, \dots, v_n)$
- 5) $\{l_1, \dots, l_s\}^D(v) = \bigvee_{i=1}^s (l_i^D = v)$
- 6) $(E_1 \wedge \dots \wedge E_s)^D(v_1, \dots, v_n) = (E_1)^D(v_1, \dots, v_n) \wedge \dots \wedge (E_s)^D(v_1, \dots, v_n)$
- 7) $(E_1 \vee \dots \vee E_s)^D(v_1, \dots, v_n) = (E_1)^D(v_1, \dots, v_n) \vee \dots \vee (E_s)^D(v_1, \dots, v_n)$
- 8) $[u_1, \dots, u_s]^D(v_1, \dots, v_s) = u_1^D(v_1) \wedge \dots \wedge u_s^D(v_s)$

In the fuzzy data type group, the supported data type predicates have been defined in ϕ_G , while the unsupported ones can be defined based on the supported ones used fuzzy data type expressions.

Definition 8. A *fuzzy data type group* G is *conforming* iff:

- 1) for any $p \in \phi_G \setminus D_G$ with $a(p) = n$ ($n \geq 2$), $dom(p) = (w, \dots, w)$ (n times) for some $w \in D_G$, and
- 2) for any $p \in \phi_G \setminus D_G$, there exists $p' \in \phi_G \setminus D_G$ such that $p'^D = \bar{p}^D$, and

- 3) the satisfiability problems for finite fuzzy predicate conjunctions of each subgroup of G is decidable.

Definition 9. $F\text{-}ALC(G)$ consists of an alphabet of distinct concept names ($\mathbf{C}$), role names ($\mathbf{R} = \mathbf{R}_A \cup \mathbf{R}_D$) and individual names ($\mathbf{I}$). $F\text{-}ALC(G)$ -roles are simply role names in $\mathbf{R}$, where $R \in \mathbf{R}_A$ is called an abstract role and $T \in \mathbf{R}_D$ is called a data type role. Let A be an atomic concept in $\mathbf{C}$, $R \in \mathbf{R}_A$, $T_1, \dots, T_n \in \mathbf{R}_D$, E is fuzzy G -data type expression. Then the valid $F\text{-}ALC(G)$ -concepts are defined as follows:

$$C ::= \top \mid \perp \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \exists T_1, \dots, T_n. E \mid \forall T_1, \dots, T_n. E \mid \geq m \mid \leq m \mid T_1, \dots, T_n. E$$

The semantics of $F\text{-}ALC(G)$ is given by an interpretation of the form $I = (\Delta^I, \bullet^I, \Delta_D, \bullet^D)$. Here $(\Delta^I, \bullet^I)$ is an interpretation of the object domain, $(\Delta_D, \bullet^D)$ is an interpretation of the fuzzy data type group G , and Δ^I and Δ_D are disjoint each other. $\bullet^I$ is an individual interpretation function and the semantics of abstract $F\text{-}ALC(G)$ -concepts like $\top$, $\perp$, A , $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, and $\forall R.C$ can be referred to [3]. $\bullet^D$ is an interpretation of the fuzzy data type group G , which respectively assigns each concrete individual to an element in Δ_D , each simple data type role $T \in \mathbf{R}_D$ to a function $T^I: \Delta^I \times \Delta_D \rightarrow [0, 1]$, and each n -ary fuzzy predicate p to a function $p^D: \Delta_D^n \rightarrow [0, 1]$ which is an n -ary fuzzy relation over Δ_D . We have the following semantics for concrete $F\text{-}ALC(G)$ -concepts with fuzzy data type expressions-related constructors:

- 1) $(\exists T_1, \dots, T_n. E)^I(x) = \sup_{v_1, \dots, v_n \in \Delta_D} ((\bigwedge_{i=1}^n T_i^I(x, v_i)) \wedge E^I(v_1, \dots, v_n))$
- 2) $(\forall T_1, \dots, T_n. E)^I(x) = \inf_{v_1, \dots, v_n \in \Delta_D} ((\bigwedge_{i=1}^n T_i^I(x, v_i)) \rightarrow E^I(v_1, \dots, v_n))$
- 3) $(\geq m T_1, \dots, T_n. E)^I(x) = \sup_{v_{11}, \dots, v_{1n}, \dots, v_{m1}, \dots, v_{mn} \in \Delta_D} \bigwedge_{i=1}^m ((\bigwedge_{j=1}^n T_{ij}^I(x, v_{ij})) \wedge E^I(v_{i1}, \dots, v_{in}))$
- 4) $(\leq m T_1, \dots, T_n. E)^I(x) = 1 - (\geq (m+1) T_1, \dots, T_n. E)^I(x)$

An $F\text{-}ALC(G)$ knowledge base $\mathcal{K}$ is in the form of $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a fuzzy $TBox$ and $\mathcal{A}$ a fuzzy $ABox$.

- A fuzzy $TBox$ is a finite set of fuzzy concept axioms of the form $C \sqsubseteq D$, where C and D are $F\text{-}ALC(G)$ -concepts.
- A fuzzy $ABox$ is a finite set of fuzzy assertions of the form $\langle \alpha \bowtie n \rangle$. Here $\bowtie \in \{>, \geq, <, \leq\}$, $n \in [0, 1]$, and α is the form either $x: C$, $(x, v): T$ (v is concrete variable) or $(x, y): R$ ($x, y \in \mathbf{I}$).

An $F\text{-}ALC(G)$ -concept C is satisfiable iff there exists some fuzzy interpretation I for which there is some $x \in \Delta^I$ such that $C^I(x) = n$, and $n \in (0, 1]$. A fuzzy interpretation I satisfies a fuzzy $TBox$ $\mathcal{T}$ iff $\forall x \in \Delta^I, C^I(x) \leq D^I(x)$ for each $C \sqsubseteq D$. I is a model of $TBox$ $\mathcal{T}$ iff I satisfies all axioms in $\mathcal{T}$. An interpretation I satisfies $\langle x: C \bowtie n \rangle$ iff $C^I(x) \bowtie n$, satisfies $\langle (x, v): T \bowtie n \rangle$ iff $T^I(x, v) \bowtie n$, satisfies $\langle (x, y): R \bowtie n \rangle$ iff $R^I(x, y) \bowtie n$. Then I is a model of $ABox$ $\mathcal{A}$ iff I satisfies all assertions in $\mathcal{A}$. Finally, a fuzzy knowledge base $\mathcal{K}$ is satisfiable iff there exists an interpretation I which satisfies the $TBox$ $\mathcal{T}$ and $ABox$ $\mathcal{A}$.

3 Reasoning for $F\text{-ALC}(G)$

3.1 Tableau Algorithm for $F\text{-ALC}(G)$ -Concepts

For ease of presentation, we assume all concepts to be in *negation normal form* (NNF) [8]. In addition, we use the symbols $\triangleright$ as a placeholder for the inequalities $>$, $\geq$ and $<$ for $<$, $\leq$, use the symbol $\bowtie$ as a placeholder for all types of inequations. Also we use the symbols $\bowtie^{-1}$, $\triangleright^{-1}$, $<^{-1}$ to denote their reflections, respectively. If ψ is an assertion in $F\text{-ALC}(G)$, then ψ^C is the conjugation [3] of ψ .

The tableaux algorithm for $F\text{-ALC}(G)$ tries to prove the satisfiability of a concept expression D by constructing a model of D . The model is represented by a so-called completion forest. Its nodes correspond to either individuals (labeled nodes) or variables (nonlabeled nodes), each labeled node being labeled with a set of triples of the form $\langle C, X, k \rangle$, which respectively denote the concept, the type of inequality (X the set $\{\geq, >, <, \leq\}$), and the membership degree that the individual of the node has been asserted to belong to C . We call such triples *membership triples*. While testing the satisfiability of an $F\text{-ALC}(G)$ -concept D , the sets of concepts C appearing in membership triples are restricted to subsets of $cl(D)$, which denotes the set of all sub-concepts of D . We use $cl_G(D)$ to denote the set of all the fuzzy G -data type expressions and their negations occurring in these sub-concepts.

Definition 10. Let D be a $F\text{-ALC}(G)$ -concept in NNF, R_A^D the set of abstract roles occurring in D , R_D^D the set of concrete roles occurring in D , G a fuzzy data type group, $E \in cl_G(D)$ a (possibly negated) fuzzy data type expression, and X the set $\{\geq, >, <, \leq\}$. A *tableau* $T = (S_A, S_D, L, DC, \varepsilon_A, \varepsilon_D)$ for D is defined as follows:

- S_A is a set of individuals; S_D is a set of variables,
- $L: S_A \rightarrow 2^{cl(D)} \times X \times [0,1]$ maps each individual in S_A to membership triples,
- DC : is a set of fuzzy data type constraints of the form $\langle E(v_1, \dots, v_n), X, k \rangle$ or inequations of the form $\langle \langle v_{i1}, \dots, v_{in} \rangle, \langle v_{j1}, \dots, v_{jn} \rangle, \neq \rangle$, where $E \in cl_G(D)$, $v_{i1}, \dots, v_{in}, v_{j1}, \dots, v_{jn} \in S_D$, $k \in [0, 1]$,
- $\varepsilon_A: R_A^D \rightarrow 2^{S_A \times S_A} \times X \times [0,1]$ maps abstract role in R_A^D to membership triples,
- $\varepsilon_D: R_D^D \rightarrow 2^{S_A \times S_D} \times X \times [0,1]$ maps data type role in R_D^D to membership triples.

A tableaux algorithm for $F\text{-ALC}(G)$ works on a completion forest $\mathcal{F}$ for D . There are two kinds of nodes in the completion forest: abstract nodes (the normal labeled nodes) and data type nodes (nonlabeled leaves of $\mathcal{F}$). Each abstract node x is labeled by a set of triples $L(x)$, which contains membership triples. We define $L(x) = \{\langle C, X, k \rangle\}$, where $C \in cl(D) \cup \{\uparrow(R, \{o\}) \mid R \in R_A^D \text{ and } \{o\} \in I^D\}$. Each edge $\langle x, y \rangle$ is labeled with a set of $L(\langle x, y \rangle)$ defined as, $L(\langle x, y \rangle) = \{\langle R, X, k \rangle\}$ where R is either abstract roles occurring in R_A^D , or data type roles in R_D^D . In the first case, y is a node and called an abstract successor of x ; in the second case, y is a data type node, and called a data type successor of x . For each $\{o\} \in I^D$, there is a distinguished node $z_{\{o\}}$ in $\mathcal{F}$ such that $\{o\} \in L(z)$.

We use a set $DC(x)$ to store the fuzzy data type expressions that must hold with regard to data type successors of a node x in $\mathcal{F}$. Each element of $DC(x)$ is either of the

form $\langle E(v_1, \dots, v_n), X, k \rangle$, or of the form $\langle v_{i1}, \dots, v_{in} \rangle, \langle v_{j1}, \dots, v_{jn} \rangle, \neq$, where $E \in cl_d(D)$, $v_1, \dots, v_n, v_{i1}, \dots, v_{in}, v_{j1}, \dots, v_{jn} \in S_D$, and $k \in [0, 1]$. Here $\langle v_1, \dots, v_n \rangle, \langle v_{i1}, \dots, v_{in} \rangle$ and $\langle v_{j1}, \dots, v_{jn} \rangle$ are tuples of fuzzy data type successors of x . The tableaux algorithm calls a fuzzy data type reasoner as a subprocedure for the satisfiability of $DC(x)$. We say that $DC(x)$ is satisfiable if the fuzzy data type query

$$\wedge_{\langle E(v_{i1}, \dots, v_{im}), X, k \rangle \in DC(x)} \langle E(v_{i1}, \dots, v_{im}), X, k \rangle \wedge \wedge_{(\langle v_{i1}, \dots, v_{in} \rangle, \langle v_{j1}, \dots, v_{jn} \rangle, \neq) \in DC(x)} \neq (v_{i1}, \dots, v_{in}; v_{j1}, \dots, v_{jn}) \quad (1)$$

is satisfiable. $DC(x)$ plays as the interface of the fuzzy DL concept reasoner and the fuzzy data type reasoner.

Definition 11. A node x of a completion forest $\mathcal{F}$ is said to contain a *clash* if (at least) it contains either an F - ALC -clash [3] or a G_f -clash. A node x of a completion forest $\mathcal{F}$ contains a G_f -clash if it contains one of the following:

- 1) for some fuzzy data type roles $T_1, \dots, T_n, \leq m T_1, \dots, T_n \cdot E, \triangleright, k \in L(x)$ and there are $m+1$ $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_{i1}, \dots, v_{in} \rangle$ ($1 \leq i \leq m+1$) of x with $\langle E \langle v_{i1}, \dots, v_{in} \rangle, \triangleright, k \rangle \in DC(x)$ ($1 \leq i \leq m+1$), and $(\langle v_{i1}, \dots, v_{in} \rangle, \langle v_{j1}, \dots, v_{jn} \rangle, \neq) \notin DC(x)$ ($1 \leq i < j \leq m+1$);
- 2) for some abstract node x , $DC(x)$ is unsatisfiable.

In the following, we give a tableaux algorithm that can construct a fuzzy tableau for an F - $ALC(G)$ -concept D , which is an abstraction of a model of the concept.

Input: F - $ALC(G)$ -concept D ;

Output: Boolean;

Algorithm:

- Step1. Initialization for the complete forest $\mathcal{F}$: If $\{o_1\}, \dots, \{o_l\} = I^D$, the algorithm initializes the completion forest $\mathcal{F}$ to contain $l+1$ root nodes $x_0, z_{\{o_1\}}, \dots, z_{\{o_l\}}$ with $L(x_0) = \{\langle D, X, k \rangle\}$ and $L(z_{\{o_i\}}) = \{\langle \{o_i\}, \geq, 1 \rangle\}$. The $DC(x)$ of any abstract node x is initialized to the empty set.
- Step2. $\mathcal{F}$ is then expanded by repeatedly applying the completion rules, including F - ALC -complete rules [3] and G_f -complete rules (shown in Fig.1), until no rules can be applied any more.
- Step3. If the completion rules can be applied in such a way that they yield a complete, clash-free completion forest, then D is satisfiable, and the algorithm returns TRUE; otherwise, D is unsatisfiable and the algorithm returns FALSE.
- Step4. Terminate.

3.2 A Flexible Reasoning Architecture

In order to provide reasoning services for F - $ALC(G)$, we propose a reasoning architecture with three kinds of components: a tableaux-based fuzzy concept reasoner

- 1) $\exists_{p\triangleright}$ -rule: if 1. $\langle \exists T_1, \dots, T_n. E, \triangleright, k \rangle \in L(x)$, and 2. x has no $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_1, \dots, v_n \rangle$ s.t.: $L(\langle x, v_i \rangle) = \{ \langle T_i, \triangleright, k \rangle \} (1 \leq i \leq n)$, and $\langle E(v_1, \dots, v_n), \triangleright, k \rangle \in DC(x)$, then create a $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_1, \dots, v_n \rangle$ of x , such that: $L(\langle x, v_i \rangle) = \{ \langle T_i, \triangleright, k \rangle \} (1 \leq i \leq n)$, and $DC(x) = DC(x) \cup \{ \langle E(v_1, \dots, v_n), \triangleright, k \rangle \}$.
- 2) $\forall_{p\triangleleft}$ -rule: if 1. $\langle \forall T_1, \dots, T_n. E, \triangleleft, k \rangle \in L(x)$, and 2. x has no $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_1, \dots, v_n \rangle$ s.t.: $L(\langle x, v_i \rangle) = \{ \langle T_i, \triangleleft^{-1}, 1-k \rangle \} (1 \leq i \leq n)$, and $\langle E(v_1, \dots, v_n), \triangleleft, k \rangle \in DC(x)$, then create a $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_1, \dots, v_n \rangle$ of x , such that: $L(\langle x, v_i \rangle) = \{ \langle T_i, \triangleleft^{-1}, 1-k \rangle \} (1 \leq i \leq n)$, and $DC(x) = DC(x) \cup \{ \langle E(v_1, \dots, v_n), \triangleleft, k \rangle \}$.
- 3) $\exists_{p\triangleleft}$ -rule: if 1. $\langle \exists T_1, \dots, T_n. E, \triangleleft, k \rangle \in L(x)$, and 2. x has a $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_1, \dots, v_n \rangle$ s.t.: $L(\langle x, v_i \rangle) = \{ \langle *, \triangleright, r \rangle \} (1 \leq i \leq n)$, and $\langle E(v_1, \dots, v_n), \triangleleft, k \rangle \notin DC(x)$, where $\langle *, \triangleright, r \rangle$ is conjugated with $\langle *, \triangleleft, k \rangle$, $*$ stands for any binary relationship of the form $\langle x, v_i \rangle$, then $DC(x) = DC(x) \cup \{ \langle E(v_1, \dots, v_n), \triangleleft, k \rangle \}$.
- 4) $\forall_{p\triangleright}$ -rule: if 1. $\langle \forall T_1, \dots, T_n. E, \triangleright, k \rangle \in L(x)$, and 2. x has a $\langle T_1, \dots, T_n \rangle$ -successor $\langle v_1, \dots, v_n \rangle$ s.t.: $L(\langle x, v_i \rangle) = \{ \langle *, \triangleright, r \rangle \} (1 \leq i \leq n)$, and $\langle E(v_1, \dots, v_n), \triangleright, k \rangle \notin DC(x)$, where $\langle *, \triangleright, r \rangle$ is conjugated with $\langle *, \triangleright^{-1}, 1-k \rangle$, $*$ stands for any binary relationship of the form $\langle x, v_i \rangle$, then $DC(x) = DC(x) \cup \{ \langle E(v_1, \dots, v_n), \triangleright, k \rangle \}$.
- 5) $\geq_{p\triangleright}$ -rule: if 1. $\langle \geq m T_1, \dots, T_n. E, \triangleright, k \rangle \in L(x)$, and 2. there are no m $\langle T_1, \dots, T_n \rangle$ -successors $\langle v_{i1}, \dots, v_{in} \rangle (1 \leq i \leq m)$ connected to x s.t.: $L(\langle x, v_{ij} \rangle) = \{ \langle T_j, \triangleright, k \rangle \} (1 \leq i \leq m, 1 \leq j \leq n)$, and $\langle E(v_{i1}, \dots, v_{in}), \triangleright, k \rangle \in DC(x) (1 \leq i \leq m)$ and $\langle v_{i1}, \dots, v_{in} \rangle, \langle v_{j1}, \dots, v_{jn} \rangle \neq \in DC(x) (1 \leq i < j \leq m)$, then create m new $\langle T_1, \dots, T_n \rangle$ -successors $\langle v_{i1}, \dots, v_{in} \rangle (1 \leq i \leq m)$ connected to x s.t.: 1. $L(\langle x, v_{ij} \rangle) = \{ \langle T_j, \triangleright, k \rangle \} (1 \leq i \leq m, 1 \leq j \leq n)$, and 2. $DC(x) = DC(x) \cup \langle E(v_{i1}, \dots, v_{in}), \triangleright, k \rangle (1 \leq i \leq m)$, and 3. $DC(x) = DC(x) \cup \{ \langle v_{i1}, \dots, v_{in} \rangle, \langle v_{j1}, \dots, v_{jn} \rangle \neq \} (1 \leq i < j \leq m)$.
- 6) $\leq_{p\triangleleft}$ -rule: if $\langle \leq m T_1, \dots, T_n. E, \triangleleft, k \rangle \in L(x)$, then apply $\geq_{p\triangleright}$ -rule for the triple $\langle \leq (m+1) T_1, \dots, T_n. E, \triangleleft^{-1}, 1-k \rangle$.
- 7) $\leq_{p\triangleright}$ -rule: if 1. $\langle \leq m T_1, \dots, T_n. E, \triangleright, k \rangle \in L(x)$, and 2. x has $m+1$ $\langle T_1, \dots, T_n \rangle$ -successors $\langle v_{i1}, \dots, v_{in} \rangle (1 \leq i \leq m+1)$ s.t.:
 - a) $L(\langle x, v_{ij} \rangle) = \{ \langle *, \triangleright, r \rangle \} (1 \leq i \leq m+1, 1 \leq j \leq n)$, where $\langle *, \triangleright, r \rangle$ is conjugated with $\langle *, \triangleright^{-1}, 1-k \rangle$, $*$ stands for any binary relationship of the form $\langle x, v_{ij} \rangle$, and
 - b) $\langle E(v_{i1}, \dots, v_{in}), \triangleright, k \rangle \in DC(x) (1 \leq i \leq m+1)$, and
 - c) In the above $m+1$ $\langle T_1, \dots, T_n \rangle$ -successors, there exists two different $\langle T_1, \dots, T_n \rangle$ -successors $\langle v_{s1}, \dots, v_{sn} \rangle$ and $\langle v_{t1}, \dots, v_{tn} \rangle (s \neq t)$ s.t. $\{ \langle v_{s1}, \dots, v_{sn} \rangle, \langle v_{t1}, \dots, v_{tn} \rangle \neq \} \notin DC(x)$, then for each pair v_{sj}, v_{tj} , where v_{sj} and v_{tj} are not the same data type T_j -successor of x , do: 1. $L(\langle x, v_{sj} \rangle) = L(\langle x, v_{sj} \rangle) \cup L(\langle x, v_{tj} \rangle)$, and 2. replace v_{tj} with v_{sj} in $DC(x)$, remove v_{tj} , and 3. $L(\langle x, v_{tj} \rangle) = \emptyset$.
- 8) $\geq_{p\triangleleft}$ -rule: if $\langle \geq m T_1, \dots, T_n. E, \triangleleft, k \rangle \in L(x)$, then apply $\leq_{p\triangleright}$ -rule for the triple $\langle \geq (m-1) T_1, \dots, T_n. E, \triangleleft^{-1}, 1-k \rangle$.

Fig. 1. G_F -complete rules

running the $F\text{-}ALC(G)$ -rules and two kinds of fuzzy data type reasoners. The first kind of fuzzy data type reasoners is called fuzzy data type manager, while the second is called fuzzy data type checkers. The detailed design of these can be referred to [9].

As shown in Fig. 2, the reasoning of fuzzy data type group is not included in the fuzzy concept reasoner, but performs in the fuzzy data type reasoners. We can use the fuzzy data type reasoners to decide the satisfiability of the query in formula (1). The tableaux algorithm calls a fuzzy data type reasoner as a subprocedure

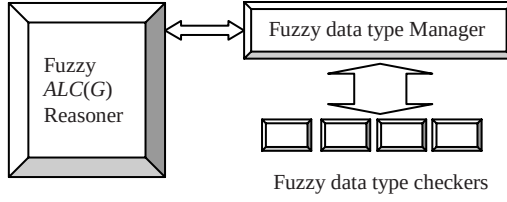


Fig. 2. Framework architecture

for the satisfiability of $DC(x)$. The fuzzy data type reasoner together with the tableau algorithm for $F\text{-}ALC(G)$ form the complete procedure to check the satisfiability of the $F\text{-}ALC(G)$ -concepts.

3.3 Decidability of the $F\text{-}ALC(G)$ Tableau Algorithm

Theorem 1. *If $G = (\phi_G, D_G, dom)$ is a conforming fuzzy data type group, then the satisfiability problem for finite fuzzy predicate conjunctions of G is decidable.*

Proof. Let the fuzzy predicate conjunction be $\zeta = \zeta_{w_1} \wedge \dots \wedge \zeta_{w_k} \wedge \zeta_U$, where $D_G = \{w_1, \dots, w_k\}$, ζ_{w_i} is the fuzzy predicate conjunction for *sub-group*(ζ_{w_i}, G), and ζ_U is the sub-conjunction of ζ and only unsupported fuzzy predicates appear. According to Definition 8, $\zeta_S = \zeta_{w_1} \wedge \dots \wedge \zeta_{w_k}$ is decidable. ζ_U is unsatisfiable iff there exist $\langle p(v_1, \dots, v_n), \triangleright, k \rangle$ and $\langle p(v_1, \dots, v_n), \triangleright^-, k' \rangle$ ($k \geq k'$) for some $p \notin \phi_G$ appearing in ζ_U . The satisfiability of ζ_U is clearly decidable.

Theorem 2. *Let G be a conforming fuzzy data type group. The fuzzy G -data type expression conjunctions of formula (1) are decidable.*

Proof. We can reduce the satisfiability problem for fuzzy G -data type expressions to the satisfiability problem for fuzzy predicate conjunctions over G ;

- 1) $[u_1, \dots, u_s]$ constructor simply introduces fuzzy predicate conjunctions. Similarly, its negation introduces disjunctions. According to the Definition 8, we can eliminate relativised negations in $[u_1, \dots, u_s]$.
- 2) The *and* and *or* constructors simply introduce disjunctions of fuzzy predicate conjunctions of G .
- 3) We can transform the tuple-level constraints to a disjunction of conjunctions of variable-level constraints in form $\neq (v_i, v_j)$. The conjunctions is clearly decidable.

According to Theorem 1, the satisfiability problem of fuzzy predicate conjunctions of G is decidable. So a fuzzy G -data type expression conjunction is satisfiable if one of its disjuncts is satisfiable.

Theorem 3. *For each $F\text{-}ALC(G)$ -concept D in NNF, its satisfiability is decidable.*

Proof. According to the reasoning architecture shown in Fig.2, the reasoning in fuzzy concept and fuzzy data type group can be divided separately.

The tableau algorithm of $F\text{-}ALC(G)$ is decidable. Let $m = |\text{sub}(\mathcal{A})|$, $k = |\mathbf{R}_A^{\mathcal{A}}|$, $w = |\mathbf{I}|$, $p_{max} = \max\{p \mid p \geq pT_1, \dots, T_n, E\}$, l be the number of different membership degrees

appearing in $\mathcal{A}$. The new nodes are generated by the $\exists_{\triangleright-}$, $\forall_{\triangleleft-}$, $\exists_{\triangleright\triangleright-}$, $\forall_{\triangleright\triangleleft-}$, $\geq_{\triangleright\triangleright-}$, and $\leq_{\triangleright\triangleleft-}$ -rules as successors of an abstract node x . For x , each of these concepts can trigger the generation of successors once at most, even if the node(s) generated are later removed by either the $\leq_{\triangleright\triangleright-}$ or $\geq_{\triangleright\triangleleft-}$ -rules. Since $sub(\mathcal{A})$ contains a total of m $\exists_{\triangleright-}$, $\forall_{\triangleleft-}$, $\exists_{\triangleright\triangleright-}$, $\forall_{\triangleright\triangleleft-}$, $\geq_{\triangleright\triangleright-}$, $\leq_{\triangleright\triangleleft-}$ -concepts at most, the out-degree of the forest is bounded by $2Imp_{max}$. The nodes are labeled with the sets of triples of the form $\langle C, \bowtie, k \rangle$, where $C \in sub(\mathcal{A}) \cup \{\uparrow(R, \{o\}) \mid R \in \mathbf{R}_A^{\mathcal{A}} \text{ and } \{o\} \in \mathbf{I}\}$. Since the concrete nodes have no labels and are always leaves, there are at most $2^{4l(m+kw)}$ different abstract node labels. Hence paths are of length at most $2^{4l(m+kw)}$.

The reasoning of fuzzy data type group is actually to check the satisfiability of a set of $DC(x)$ of formula (1), which is decidable according to Theorem 2. So the procedure of checking the satisfiability of $F\text{-}ALC(G)$ -concept D in NNF is decidable.

4 Conclusions

This paper presents a fuzzy DL $F\text{-}ALC(G)$. We give its tableau algorithm and propose a reasoning architecture for fuzzy data type reasoning. We will investigate the performance of the algorithm and develop a reasoning machine to verify its efficiency.

Acknowledgments. Work is supported by the Program for New Century Excellent Talents in University (NCET-05-0288).

References

1. Zadeh, L.A.: Fuzzy sets. *J. Information and Control* 8, 338–353 (1965)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F.: *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge (2003)
3. Straccia, U.: Reasoning within fuzzy description logics. *J. Journal of Artificial Intelligence Research* 14, 137–166 (2001)
4. Stoilos, G., Stamou, G., Pan, J.Z., Tzouvaras, V., Horrocks, I.: Reasoning with Very Expressive Fuzzy Description Logics. *J. Journal of Artificial Intelligence Research* 30, 273–320 (2007)
5. Horrocks, I., Sattler, U.: Ontology reasoning in the SHOQ(D) description logic. In: *Proc. 17th International Joint Conference on Artificial Intelligence*, pp. 199–204 (2001)
6. Straccia, U.: Towards a fuzzy description logic for the Semantic Web (Preliminary Report). In: *Proc. the 2nd European Semantic Web Conference*, pp. 167–181 (2005)
7. Pan, J.Z., Horrocks, I.: OWL-Eu: Adding Customised Data types into OWL. *J. Journal of Web Semantics* 4, 29–49 (2006)
8. Pan, J.Z.: A Flexible Ontology Reasoning Architecture for the Semantic Web. *J. IEEE Transaction on Knowledge and Data Engineering* 19, 246–260 (2007)
9. Wang, H.L., Ma, Z.M., Yan, L., Cheng, J.W.: A Fuzzy Description Logic with Fuzzy Data Type Group. In: *Proc. the 2008 IEEE International Conference on Fuzzy Systems*, pp. 1534–1541. IEEE Press, Hong Kong (2008)

Ranking Entities Using Comparative Relations

Takeshi Kurashima, Katsuji Bessho, Hiroyuki Toda,
Toshio Uchiyama, and Ryoji Kataoka

NTT Cyber Solutions Laboratories, NTT Corporation,
1-1 Hikarinooka Yokosuka-shi, Kanagawa 239-0847, Japan
{kurashima.takeshi, bessho.katsuji, toda.hiroyuki,
uchiyama.toshio, kataoka.ryoji}@lab.ntt.co.jp

Abstract. This paper proposes a method for ranking entities (e.g. products, people, etc.) that uses the comparative sentences described in text such as reviews, blogs, etc. as an indicator of an individual entity's value. A comparative sentence expresses a relation between two entities. The comparative sentence "The quality of A is better than that of B" is expressed by the comparative relation $\{A, B, \text{quality}, \text{better}\}$. Given a query (set of queries), the proposed method automatically finds the competitive entities and extracts the comparative relations among them. From the vast amount of comparative relations so extracted, the proposed method then generates a graph modeling the behavior of a "potential customer" to assess the relative importance of every entity against its competitors. These ranking results help potential customers to know the position of the entity among related entities and decide which one to choose.

1 Introduction

The prevalence of CGM (consumer generated media) enables consumers who experienced the products to express their opinions on the Internet. One of the important goals of potential customers is to choose one from among the many candidates. To perform this, until recently, opinions about a product (e.g. "The picture quality of camera A is good/bad") were acquired by text extraction and mining because the vast amounts of information available are mainly presented as texts [1][2]. Existing systems attempt to help the user to choose a product by providing side-by-side comparisons of summarized opinions on several products.

Systems that use opinions about a product, however, do not necessary lead to the correct order among the competitors because the opinions about a product are largely dependent on the person's evaluation perspective. For example, a product appreciated by someone who has not used other candidates may not be appreciated by someone else who has tried other, better products. Only those who have a wide perspective (i.e. knows two or more candidates) can estimate the value of a product relative to its competitor(s) correctly. We thus focus on comparative sentences such as "The picture quality of X is better than Y ". Different from opinions about an entity, comparative sentences, in many cases, lead to the correct order since they are usually made by customers who have a wide perspective, those who have actually experienced and used both entities.

This paper proposes a novel method for ranking the entities. We assume that the user query(queries) can be a product name, a person name, or etc. The competitors of the query(queries) are automatically discovered and ranked by consumers' opinions. The proposed method is based on graph-based analysis. We generate a graph wherein each node corresponds to an entity, and each edge expresses the relation between two entities. The feature of the graph is to use comparative sentences in the CGM data as an indicator of an individual entity's value. The relation between two entities (e.g. X and Y) is induced by the collection of the comparative sentence between X and Y present in the CGM data (e.g. "The quality of X is better than Y ", "I prefer Y to X " or etc.). A comparative sentence "The quality of X is better than Y " is expressed by a comparative relation $\{X, Y, \text{quality}, \text{better}(\text{good})\}$. We find the competitive entities of a query(queries) among the results of reliable pattern matching and thus extract comparative relations more accurately (In this paper, the extraction of features is not described, and will be tackled in future work). From the graph we can measure the importance of every entity within a set of competitors. The basic idea for entity ranking is that an entity has high rank if it is superior to other "important" entities. The proposed graph structure models the behavior of the "Potential Customer" who shops around for a better one. The centrality score of a node is given by a probability distribution that represents the likelihood that a potential customer will arrive at that product (entity). Systems that implement our proposed method will provide a new search framework since the value of an entity is not estimated from isolated opinions but rather its relationship with other entities. An experiment shows that the proposed algorithm adopted raises the precision and recall of the extracted comparative relations. The result of movie ranking and visualization using the real blog data is also presented.

2 Related Work

A significant amount of work has been done on opinion and sentiment mining in the research fields of natural language processing and text mining. They are usually classified as "sentiment classification" and "opinion extraction". Sentiment classification assigns a document or sentence into either a positive or negative group. Turney et al. proposed an unsupervised method based on mutual information between document phrases and the words "excellent" and "poor" to find indicative words for sentiment classification[3]. Pang et al. treat the problem as a topic-based text classification problem. They tried applying a supervised method, e.g. SVM, kNN, or etc., to sentiment classification [4]. Dave et al. also classified documents using a score function [5]. The classification is usually at the document or sentence level. No details are extracted about what people liked or didn't like. On the other hand, the task of opinion extraction is to extract all or part of the opinion relations that express entities, features, and opinions. Liu et al. proposed a method for extracting language patterns to identify features that uses association rule mining [6]. They also proposed the system called *Opinion Observer* which enables the user to compare opinions on products [1][2]. Our

research is related to opinion extraction but differs in that it mines comparative relations. Jindal et al. proposed a method based on CSR and LSR for extracting language patterns indicative of comparative relations [7][8]. The language patterns so extracted are used to extract each item of the comparative relations. Our research focuses on knowledge discovery among extracted comparative relations while they focused only on language pattern extraction. Our approach is query dependent and selects only that information related to a query from among all comparative relations described in the text. We then summarize them to find the relative importance terms associated with the query’s competitors.

3 The Proposed Method for Ranking Entities

Our proposed method consists of three steps as listed below.

1. Extract comparative relations from CGM texts
2. Generate graph based on the “Potential Customer Model”
3. Compute graph centrality

We assume that the user query (queries) is an entity(entities), e.g., person, product, or company, etc. It is used to identify the entity cluster which the user is interested in. For the first step, the competitive entities that should have some features in common with the query (queries) are automatically discovered. For example, given the name of digital camera (e.g. “FinePix”) as the query, the system finds its competitors (e.g. “IXY DIGITAL” and “EXILIM” etc.) which have the same features (e.g. price, design, or weights etc). Comparative relations among the collected entities including the query (queries) are then extracted. (The definition of “comparative relation” is described in Sec. 3.1.)

The second step generates a graph wherein each node corresponds to an entity, and each edge expresses the relation between two entities is generated. The relation between entities(nodes) is induced by the collection of the comparative relations extracted in the first step. The graph structure models the behavior of the “Potential Customer” who shops around for a better entity. Finally, centrality values for a given generation graph are then computed.

3.1 The Definition

A comparative sentence expresses an ordering relation between two entities. According to related work[7][8], English comparative sentence can be categorized into the four types; (1)*Non-Equal Gradable* (2)*Equative* (3) *Superlative* (4)*Non-Gradable*. Our research focus only on (1), they express a total ordering of some entities with regard to certain features, because it typically expresses user preference and represents important information in measuring the value of an entity. Type (1) comparative sentences are expressed by the comparative relation: $R = \{\text{entity1}, \text{entity2}, \text{feature}, \text{opinion}\}$. An entity can be a product name, a person name, etc. Two entities are ranked according to a feature. An opinion word is a expression of the user’s evaluation. The target of the opinion is entity1

and the other is entity2. For example, “The quality of A is clearer than that of B” is expressed by $R=\{A,B,quality,clear\}$. This paper proposes a method for extracting entity1, entity2 and opinion; future work includes the extraction of features.

3.2 Comparative Relation Extraction

A method for extracting the comparative non-equal gradable relations as defined in the previous section is described. In English linguistics, comparative sentences are expressed with specialized morphemes (e.g. “more”, “most”, “-er”, “-est”, or etc.). Likewise, Japanese comparative sentences use special language phrases (e.g. “yori”, “ni kurabe”, or etc.). The most typical Japanese comparative sentence has entity1 and entity2 appearing in the special pattern (e.g. “X **no hou ga** Y **yori yoi** (X is better than Y)”). These patterns are useful in identifying comparative sentences and the components of the comparative relations. However, those special patterns, in most cases, are only used to express either entity1 or entity2; other entities appear only in broad coverage noisy patterns that extract both many correct and incorrect entities. For example, both entity1 and a feature can appear with the same particle “ha” or “ga” (e.g. “X ha Y yori yoi”). Accordingly, using these generic patterns increases system recall while reducing precision. The proposed algorithm takes as input a seed query (a few seed queries) and iteratively collects entities identified by reliable results (i.e. both entity1 and entity2 are extracted since they occur on one of the special language patterns of comparative sentences). The collected entities are then used to (1) extract the comparative relations among the competitors of the query (i.e. entity1 and entity2 must be competitors), and (2) offset the limited coverage of generic patterns.

Language Pattern Matching. Each document is first morphologically analysed (whether the word is a noun, verb, etc) and separated into sentence segments by ‘.’, ‘,’ , etc. Each sentence segment is converted into data as listed below:

$$t = [word_1/morpheme_1][word_2/morpheme_2]...[word_p/morpheme_p]$$

Language patterns are then used to match and identify entity1, entity2, and the opinion from the source data after morphological analysis. The language patterns used for extraction are manually prepared. For each pattern, words that match “*” are extracted. The patterns of entity1 and entity2 are categorized as listed below by their reliability in identifying each component of the relation.

- Special language patterns of comparative sentences that can identify each component of a comparative relation with high precision/ low recall.
- Ambiguous language patterns that identify each component of the comparative relation with low precision/ high recall.

For each component of the relation, two or more candidates can be extracted from a sentence. We denote the candidate words of entity1 by $x(t)=\{< x_1, F(x_1) > , ..., < x_n, F(x_n) >\}$, where x_i is a word. $F(x)$ is 1 if x is extracted by a special

pattern, otherwise $F(x)$ is 0. In the same way, the candidate words of entity2 are denoted by $y(t)=\{< y_1, F(y_1) >, ..., < y_m, F(y_m) >\}$; the candidate words of the opinion are denoted by $z(t)=\{z_1, ..., z_o\}$.

Finding the Competitive Entities. The competitors of the query are extracted using the results of pattern matching. Let Q_0 be a set of queries, Q be a set of words expanded from Q_0 in this step, and T be a set of the data after pattern matching. We first set query Q_0 to Q . The algorithm takes the seed and then iteratively expands it using the data after pattern matching, where both entity1 and entity2 are extracted by a special pattern. For each t in T , if either entity1 or entity2 is not in Q but the other is in it, an entity not in Q is added to Q because it is the competitor of the query with high probability. The scanning of T is repeated several times until no further entity is added to Q .

Example 1. We have $Q_0=\{A, B\}$ and the five pieces of data after pattern matching as listed below:

$$\begin{aligned} t_1: x(t_1) &= \{< C, 1 >, < F, 0 >\}, y(t_1) = \{< E, 1 >\}, \\ t_2: x(t_2) &= \{< H, 1 >, < D, 0 >\}, y(t_2) = \{< G, 1 >\}, \\ t_3: x(t_3) &= \{< A, 1 >\}, y(t_3) = \{< C, 1 >, < D, 0 >\}, \\ t_4: x(t_4) &= \{< E, 0 >\}, y(t_4) = \{< B, 1 >\}. \\ t_5: x(t_5) &= \{< E, 1 >\}, y(t_5) = \{< C, 0 >, < D, 0 >\}. \end{aligned}$$

Let Q_i be Q after the i th scan (Q_0 are seed entities). At the first scan, C is added to Q_0 (i.e. $Q_1 = \{A, B, C\}$) because t_3 meets the conditions; $F(A) = 1$, $F(C) = 1$, $A \in Q_0$, and $C \notin Q_0$. At the second scan, E is added to Q_1 (i.e. $Q_2 = \{A, B, C, E\}$) because t_1 meets the conditions; $F(C) = 1$, $F(E) = 1$, $C \in Q_1$, and $E \notin Q_1$. At the third scan, no other entity can be added to Q . The result of the example is $Q = \{A, B, C, E\}$.

Identifying Each Item of the Comparative Relation. We assume that a sentence has one comparative relation at most, and each comparative relation has one entity1, one entity2, and one opinion at most. For each t , the collection of Q is then applied to match $x(t)$ and $y(t)$ to choose one as the component of the comparative relation from several candidate words. For t_1 , t_2 and t_3 in **Example 1**, this matching process selects C in $x(t_1)$, E in $y(t_1)$, A in $y(t_3)$ and C in $y(t_3)$. For t_4 and t_5 , E in $x(t_4)$ and C in $y(t_5)$ are also selected although neither of them appear in the special patterns. This step increases the precision of the extracted comparative relation while keeping recall high.

Opinion is also chosen from $z(t)$ by matching words stored in the opinion dictionary whose entries are word, morpheme, and its PN (*positive*, *negative*, or *neutral*) (e.g. {good, adjective, positive}). The PN is added to each relation at the same time. If two or more opinion words are obtained by this matching process, the word closest to the end of the sentence is considered first. Thus the data after this step is a set of $\{x, y, z, PN\}$, where $x \in x(t)$, $y \in y(t)$, $z \in z(t)$, and PN is *positive*, *negative* or *neutral*.

3.3 Generation Graph Based on the Potential Customer Model

In this section, we create a graph structure, where each node represents an entity and each edge represents a relation between two entities. The following subsections describe each of these steps in more detail.

Summarizing the Results of Comparative Relations. Preprocessing of graph-based analysis, the results of comparative relations extracted in the previous section should be summarized. Let N be the total number of entities(nodes). S_{ij} is the total number of comparative relations that indicate that entity(node) V_i is superior to entity(node) V_j . G_i is the total number of entities(nodes) associated with V_i (i.e. the total number of neighbor nodes of V_i). When $S_{ij} > S_{ji}$, we consider that V_i defeats V_j . W_i is the total number of victories out of G_i ($W_i \leq G_i$). These values are calculated.

Modeling the Behavior of Potential Customer. The proposed graph models the behavior of the “Potential Customer” who shops around for the best entity by referring to the comparatives. The customer starts by looking at one product (entity) and then proceeds to look at its competitors (entities). The customer’s next choice depends on the comparative merit and demerit of the current entity and the other entity. The better the entity is compared to the current entity, the more likely it is that the customer will choose it. These transition probabilities define a Markov chain between the entities. Our graph also considers self transition. If there are few competitors that are superior to the current product, self transition probability is high (i.e. when there is a large number of products that are superior, self transition is low). We denote A_{ij} as the transition probability from node V_i to node V_j . Matrix A is calculated by the following equation.

$$A_{ij} = \frac{B_{ij}}{\sum_{k=1}^N B_{ik}}$$

$$B_{ij} = \begin{cases} \alpha \frac{S_{ji}}{S_{ji}+S_{ij}} & \text{if } i \neq j \\ \beta \frac{W(i)}{G(i)} & \text{if } i = j \\ 0 & \text{if } i \neq j \text{ and } S_{ij} = 0 \text{ and } S_{ji} = 0 \end{cases}$$

The probability of a transition to a neighbor node is calculated by the value of S (if $i \neq j$) and self transition is calculated by the value of W and G (if $i = j$). Parameters α and β assign a numeric weighting and they are used to control their contribution ratios. Matrix A is finally calculated after normalization of matrix B to ensure that the sum of probability from each node is 1.

Example 2. We have nodes $V=\{a,b,c,d\}$ and S as listed below;

$$S_{ab}=1, S_{ba}=3, S_{ac}=4, S_{ca}=2, S_{ad}=1, S_{da}=2$$

$S_{ab}=1$ indicates that the total number of comparative relations that express that “ a is superior to b ” ($a > b$) is 1. Fig. 1 shows how to calculate the transition probability from node A to a neighbor node, and Fig. 2 shows how to calculate the self transition probability of node a .

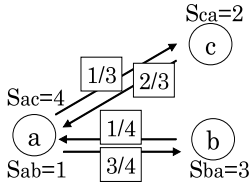


Fig. 1. How to calculate the probability of the transition to the neighbor node

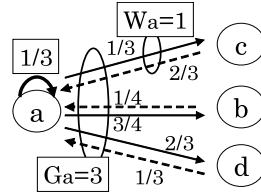


Fig. 2. How to calculate the probability of self transition

3.4 Computing Graph Centrality

The graph centrality score is calculated in order to measure the relative importance of an entity within its related entity cluster. The importance(score) of node V_j is determined by V_i of those links connected to V_j . An entity is important if it is superior to an other important entity. The equation is as follows.

$$S(V_j) = (1 - d) \times \sum (A_{ij} \times S(V_i)) + d$$

The definition is similar to PageRank[9], which uses a link analysis algorithm based on graph structure to find important web pages, in the following respect: it uses jumping factor d . The jumping factor is the probability that the user will stop visiting neighbor nodes and instead randomly visit any node in the graph. However, it differs from PageRank in the following respect: each edge in the graph has a weight. The calculated score of the entity is a probability distribution that represents the likelihood that a potential customer who shops around for a better product will arrive at any product.

4 Evaluation

We evaluated our proposed method using a prototype system based on the implementation described in the previous section. The system crawls the Web and collects blog entries or customer reviews. The text documents were morphologically analyzed using the JTAG morphological analyzer [10].

4.1 Evaluation of Comparative Relation Extraction

This section evaluates our proposed algorithm in extracting entity1 and entity2. We assumed that the user query, Q_0 , asked for 5 movies that were screened in October 2006 and collected 100 consumer reviews for each of the movies (500 movie reviews in total). The correct data sets were manually labeled by 3 human labelers. Conflicts were resolved by majority vote. 63 documents actually contained comparative sentences. The total number of entity1 and entity2 were 40 and 76 respectively. The comparative relations were extracted using the proposed algorithm. We then evaluated the results in terms of the precision, recall and F-Measure of entity1 and entity2, and observed how each step of the algorithms listed below improved the precision, recall and F-measure.

Table 1. The results of evaluation for extracting entity1 and entity2

		(1)	(1)+(2)	(1)+(3)	(1)+(2)+(3)
entity1	Precision	0.377	0.833	0.333	0.714
	Recall	0.725	0.500	0.100	0.625
	F-Measure	0.496	0.625	0.154	0.667
entity2	Precision	0.737	1.000	0.889	0.981
	Recall	0.737	0.592	0.105	0.684
	F-Measure	0.737	0.744	0.188	0.806

(1) Language pattern matching

(2) Identifying item using Q (Q is expanded from Q_0)

Tab. 1 shows the evaluation result. It shows that step (2) improved the F-measure of entity1 and entity2 because it increased the precision while keeping the recall relatively-high. In particular, the results of entity1 were improved dramatically (F-measure of entity1 increased from 0.496 to 0.625) because many ambiguous patterns(low precision/ high recall) identified entity1.

In most real world documents, the same entity was described in different words. For example, “M:i:3”, “M:I:3”, and “Mission Impossible 3” all refer to the same movie. Our method can identify these synonyms successfully. To prove this, we collected the formal movie names screened in Japan between January 1th 1995 and December 31 2006 and step(3) listed below is also shown.

(3) Identifying item using M (M is a collection of formal movie names)

The low recall of the combination of (1) and (3) indicates that the formal movie names are used infrequently while their synonyms are used frequently. The combination of (1), (2), and (3) yields the best F-measure because (3) improves the recall of (1)+(2) (although in many cases it is difficult to prepare them (e.g. person names, product names, etc.).

4.2 The Results of Movie Ranking and Visualization

We show an example of movie ranking and visualization based on the graph structure and centrality score. We collected blog entries entered between 1th May 2006 and 31th January 2007 using an existing blog search engine[11]. “The Da Vinci Code” which was screened in Japan from May 2006 was set as the query of our system. After finding the competitors, we obtained 179 titles in total and actually extracted their comparative relations(2326). The movie ranking results are indicated in Tab. 2. As shown, the Japanese movie “MEMORIES OF MATSUKO” had the top centrality score. The reason why the “MEMORIES OF MATSUKO” gets high score is that the score is propagated from entities with relatively high score such as the entity, “LIMIT OF LOVE -UMIZARU-” or etc. The total number of comparative relations that indicate “MEMORIES OF MATSUKO” is superior to “The Da Vinci Code” is 9 (out of the total of

Table 2. The results of movie ranking (query is “The Da Vinci Code”, a numeric weighting of transition possibility to other node is 1.0, self transition possibility is 1.0, jumping factor d is 0.3)

Rank	Movie Title	Centrality Score
1	MEMORIES OF MATSUKO	0.3572
2	Tales from Earthsea	0.3179
3	LIMIT OF LOVE -UMIZARU-	0.2782
4	Pirates of the Caribbean: Dead Man’s Chest	0.2646
5	DEATH NOTE	0.2520
6	M:i:III	0.1880
7	The Girl Who Leapt Through Time	0.1835
8	The Da Vinci Code	0.1742
9	The Devil Wears Prada	0.1733
10	Over The Hedge	0.1654

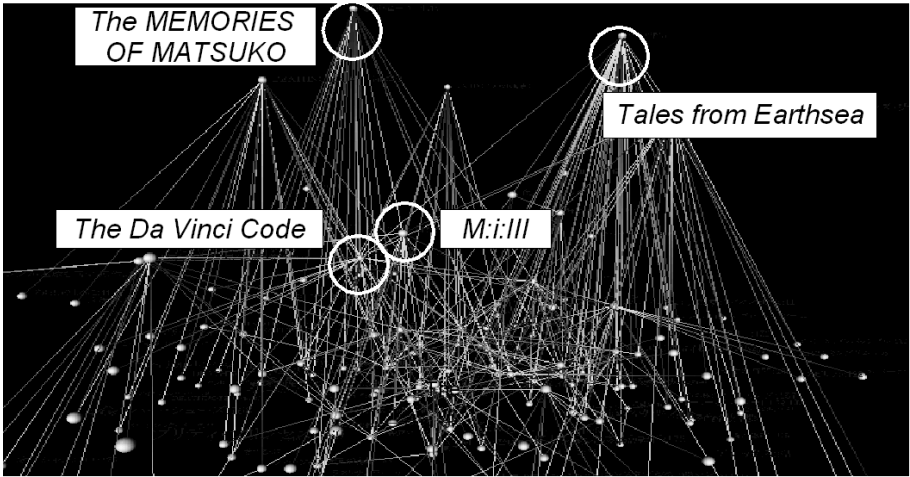


Fig. 3. The visualized results of relations among movies (a numeric weighting of transition possibility to other node is 1.0, self transition possibility is 1.0, jumping factor d is 0.3)

11 comparative relations). The movie titles which are ranked ahead of “The Da Vinci Code” tend to be the movies which are directly compared and superior to the query. Another reason is that “MEMORIES OF MATSUKO” has high self transition probability (its record is 17 victories and 6 defeats) while the query has low self transition probability(its record is 5 victories and 23 defeats).

Fig. 3 shows an example of visualization based on the graph structure and centrality score. In this visualization, we used Yamada’s method[12] and the centrality scores are plotted on the 3rd dimension. The white spheres in the graph correspond to nodes and the lines between them correspond to edges.

Edge direction is represented by color; the “from” side is light and the “to” side is dark. Two-way edges are white. This ranking and visualization enables the person to understand the relationships between “The Da Vinci Code” and its competitors, and the its relative importance.

5 Conclusion

This paper introduced a method for ranking: mining the importance of an entity (e.g. product, person, company, etc.) relative to its competitors by using comparative relations. There are two technical contributions:

- An algorithm is proposed that can accurately find competitive entities by using the results of reliable pattern matching.
- A graph structure is proposed to model the behavior of the “Potential Customer” who shops around for a better entity.

The results of experiments on actual Web-sourced data showed that the system can extract relevant comparative relations for various movies with relatively high precision. Our future work includes additional evaluations with other types of queries (person or company). Feature extraction is another research interest.

References

1. Liu, B., Hu, M., Cheng, J.: Opinion Observer: Analyzing and Comparing Opinions on the Web. In: Proc. of WWW 2005, pp. 342–351 (2005)
2. Hu., M., Liu, B.: Mining and Summarizing Customer Reviews. In: Proc. of KDD 2004, pp. 168–177 (2004)
3. Turney, P.D.: Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proc. of ACL 2002, pp. 417–424 (2002)
4. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification using Machine Learning Techniques. In: Proc. of EMNLP 2002, pp. 76–86 (2002)
5. Dave, K., Kawrence, S., Pennock, D.: Mining the Peanut Gallery: Opinion Extraction and Sentiment Classification of Product Reviews. In: Proc. of WWW 2003, pp. 519–528 (2003)
6. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. of VLDB 1994, pp. 487–499 (1994)
7. Jindal, N., Liu, B.: Mining Comparative Sentences and Relations. In: Proc. of AAAI 2006, pp. 1331–1336 (2006)
8. Jindal, N., Liu, B.: Identifying Comparative Sentences in Text Documents. In: Proc. of SIGIR 2006, pp. 244–251 (2006)
9. Brin, S., Page, L., Motwami, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web, Technical report, Computer Science Department, Stanford University (1998)
10. Fuchi, T., Takagi, S.: Japanese Morphological Analyzer using Word Co-occurrence-JTAG. In: Proc. of ACL-COLING 1998, pp. 409–413 (1998)
11. goo Blog, <http://blog.goo.ne.jp/>
12. Yamada, T., Saito, K., Ueda, N.: Cross-Entropy Directed Embedding of Network Data. In: Proc. of ICML 2003, pp. 832–839 (2003)

Query Recommendation Using Large-Scale Web Access Logs and Web Page Archive

Lin Li¹, Shingo Otsuka², and Masaru Kitsuregawa¹

¹ Dept. of Info. and Comm. Engineering, The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
{lilin, kitsure}@tkl.iis.u-tokyo.ac.jp

² National Institute for Materials Science
1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan
otsuka.shingo@nims.go.jp

Abstract. Query recommendation suggests related queries for search engine users when they are not satisfied with the results of an initial input query, thus assisting users in improving search quality. Conventional approaches to query recommendation have been focused on expanding a query by terms extracted from various information sources such as a thesaurus like WordNet¹, the top ranked documents and so on. In this paper, we argue that past queries stored in query logs can be a source of additional evidence to help future users. We present a query recommendation system based on large-scale Web access logs and Web page archive, and evaluate three query recommendation strategies based on different feature spaces (i.e., noun, URL, and Web community). The experimental results show that query logs are an effective source for query recommendation, and the Web community-based and noun-based strategies can extract more related search queries than the URL-based one.

1 Introduction

Keyword based queries supported by Web search engines help users conveniently find Web pages that match their information needs. A main problem, however, occurs for users: properly specifying their information needs through keyword-based queries. One reason is that queries submitted by users are usually very short [8]. The very small overlap of the query terms and the document terms in the desired documents will retrieve Web pages which are not what users are searching for. The other reason is that users might fail to choose terms at the appropriate level of representation for their information needs. Ambiguity of short queries and the limitation of user's representation give rise to the problem of phrasing satisfactory queries.

The utilization of query recommendation has been investigated to help users formulate satisfactory queries [1,3,4,8,9,10]. For example, new terms can be expanded to the existing list of terms in a query and users can also change some

¹ <http://wordnet.princeton.edu/>

or all the terms in a query. We summarize the main process of query recommendation in the following three steps:

1. choosing information sources where most relevant terms as recommendation candidates can be found, given a current query;
2. designing a measure to rank candidate terms in terms of relatedness;
3. utilizing the top ranked relevant terms to reformulate the current query.

The selection of information sources in the first step plays an important role for an effective query recommendation strategy. The general idea of existing query recommendation strategies [3,4,9,10] has focused on finding relevant terms from *documents* to existing terms in a query based on the hypothesis that a frequent term from the documents will tend to co-occur with all query terms. On the Web, this hypothesis is reasonable, but not always true since there exists a large gap between the Web document and query spaces, as indicated by [4] which have utilized query logs to bridge the query and Web document spaces. In this paper, different from the above researches, we think that *past queries* stored in the query logs may be a source of additional evidence to help future users. Some users who are not very familiar with a certain domain, can gradually refine their queries from related queries that have been searched by previous users, and hence get the Web pages they want.

In this paper, we present and evaluate a query recommendation system using past queries that provide a pool of relevant terms. To calculate the relatedness between two queries, we augment short Web queries by three feature spaces, i.e., noun space, URL space, and community (*community* means *Web community* in this paper) space respectively. The three feature spaces are based on Web access logs (the collected 10GB URL histories of Japanese users selected without static deviation) and a 4.5 million Japanese Web page archive. We propose a query recommendation strategy using a community feature space which is different from the noun and URL feature spaces commonly used in the literature [1,3,4,8,9]. The evaluation of query recommendation is labor intensive and it is not easy to construct an object test data set for it at current stage. An evaluation is carefully designed to make it clear that to which degree different feature based strategies add value to the query recommendation quality. We study this problem and provide some preliminary conclusions. The experimental results show that query logs are an effective source for query recommendation, and community-based and noun-based methods can extract more related search keywords than the URL-based one.

The rest of this paper is organized as follows. Firstly, we introduce three query recommendation strategies in Section 2. Then, we describe the details of experiment methodology and discuss the experimental results in Section 3 and Section 4 respectively. Lastly, we conclude our work in Section 5.

2 Query Recommendation Strategies

The goal our recommendation system is to find the related past queries to a current query input by a Web user, which means we need to measure the relatedness

between queries and then recommend the top ranked queries. Previous queries having common terms with the input query are naturally recommended. However, it is possible that queries can be phrased differently with different terms but for the same information needs while they can be identical but for the different information needs. To more accurately measure relatedness between two queries, most of existing strategies augment a query by terms from Web pages or search result URLs [1,5,8]. We think that the information related to the accessed Web pages by Web users are useful sources to augment original queries because the preferences of a user are reflected in form of her accesses. One important assumption behind this idea is that the accessed Web pages are *relevant* to the query. At the first glance, although the access information is not as accurate as explicit relevance judgment in the traditional relevance feedback, the user's choice does suggest a certain level of relevance. It is therefore reasonable to regard the accessed Web pages as relevant examples from a statistical viewpoint.

2.1 Three Feature Spaces for Query Recommendation

Given these accessed Web pages, we define three feature spaces as *noun space*, *URL space*, and *community space* to augment their corresponding Web queries and then estimate the relatedness between two augmented queries.

Noun Space. As we know, nouns in a document can more accurately represent the topic described by the document than others. Therefore, we enrich a query with the nouns extracted from the contents of its accessed Web page sets, which intends to find the topics hidden in the short query. Our noun space is created using ChaSen, a Japanese morphological analyzer². Since we have already crawled a 4.5 million Japanese Web page archive, we can complete the morphological analysis of all the Web pages in advance.

URL Space. The query recommendation strategy using the noun feature space is not applicable, at least in principle, in settings including: non-text pages like multimedia (image) files, documents in non-HTML file formats such as PDF and DOC documents, pages with limited access like sites that require registration and so on. In these cases, URLs of Web pages are an alternate source. Furthermore, because the URL space is insensitive to content, for online applications, this method is easier and faster to get recommendation lists of related past queries than the noun feature space. Our URL space consists of the hostnames of accessed URLs in our Web logs. The reason that we use hostnames of URLs will be explained in Section 3.2.

Community Space. The noun and URL spaces are straightforward and common sources to enhance a query. In this paper, we utilize the Web community information as another useful feature space since each URL can be clustered into its respective community. The technical detail of creating community is in our previous work [7] which created a web community chart based on the complete bipartite graphs, and extracted communities automatically from a large amount

² <http://chasen-legacy.sourceforge.jp/>

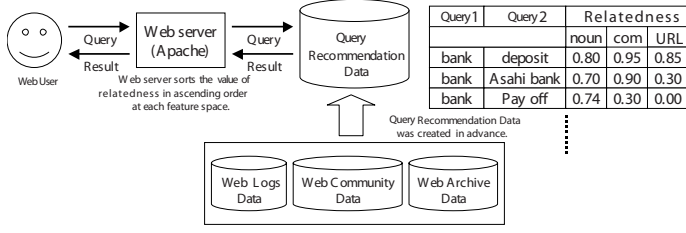


Fig. 1. The architecture of our system

of web pages. We labeled each Web page by a community ID. As thus, these community IDs constitute our community space.

2.2 Relatedness Definition

In this section we discuss how to calculate relatedness between two queries enriched by the three feature spaces. The relatedness is defined as

$$R_{q_x, q_y} = \frac{\sum_{e_i \in q_x \& e_i \in q_y} f_{q_x}(e_i) + \sum_{e_i \in q_x \& e_i \in q_y} f_{q_y}(e_i)}{2},$$

where q_x and q_y are queries, and R_{q_x, q_y} is the estimated relatedness score between q_x and q_y . Let $Q = \{q_1, q_2, \dots, q_x, \dots, q_n\}$ be a universal set of all search queries where n is the number of total search queries. We augment the query q_x by adding one of the three feature spaces (i.e., noun, URL, and community ID), denoted as: $q_x = \{e_1, e_2, \dots, e_x, \dots, e_m\}$ where e_x is an element of the used feature space and m is the total number of elements. The frequencies of elements in a query are denoted as $f_{q_x} = \{f_{e_1}, f_{e_2}, \dots, f_{e_x}, \dots, f_{e_m}\}$ for a single feature space. For URL space, we can get the frequencies of URLs visited by different users using access information in our Web logs.

In addition, we create an excluded set which stores the *highly frequent elements* of URLs, communities and nouns contained in accessed Web page sets. For example, the highly frequent elements of URLs are *Yahoo!*, *MSN*, *Google* and so on, and the highly frequent elements of nouns are *I*, *today*, *news* and so on. We exclude a highly frequent element e_h from the frequency space of f_{q_x} if the number of the test queries which feature spaces include e_h are more than half of the number of all the test queries used in our evaluation.

3 Experiment Methodology

3.1 Our System Overview

The goal of our query recommendation system is to find the related queries from past queries given an input query. Figure 1 shows a sketch of the system architecture consisting of two components. One is “Web Server(Apache)” where a

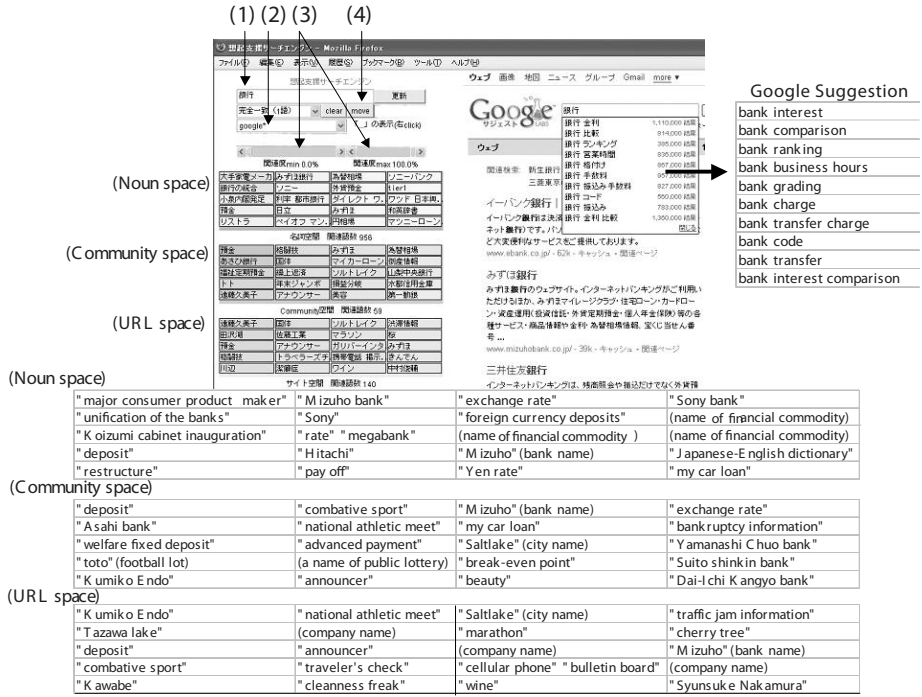


Fig. 2. The user interface of our system

user interactively communicates with our system. The user inputs a query to the Web server, and then the Web server returns the list of related past queries to her. The other is “Query Recommendation Data” storing the recommendation results of the three strategies discussed in Section 2. Our Web logs, Web community, and Web page archive data ensure the richness of information sources to augment Web queries. The interface of our query recommendation system is illustrated in Figure 2. A user can input a search query in Figure 2(1) while the related queries recommended by the component “Query Recommendation Data” are shown below and divided by using different feature spaces. Then, the user can choose one recommended query to add or replace the initial query and submit the reformulated query to a search engine selected from a dropdown list as shown in Figure 2(2). Finally, the search results retrieved by the selected search engine are listed in the right part. Furthermore, in Figure 2(3), there are two slide bars which can adjust the lower and upper bounds of relatedness scores. For each feature space, the maximal number of recommended queries is 20. If the user wants more hints, she can click a button shown in Figure 2(4) to get more recommended queries ordered by their relatedness scores with the initial query. The more the recommended query is related to the initial query, the query is displayed as a deeper red. Figure 2 presents recommendation of the query “Bank” as an example. Since in this study we utilize Japanese Web data,

UserID	AccessTime	RefSec	URL
1	2002/9/30 00:00:00	4	http://www.tkl.iis.u-tokyo.ac.jp/welcome_j.html
2	2002/9/30 00:00:00	6	http://www.jma.go.jp/JMA_HP/jma/index.html
3	2002/9/30 00:00:00	8	http://www.kantei.go.jp/
4	2002/9/30 00:00:00	15	http://www.google.co.jp/
1	2002/9/30 00:00:04	6	http://www.tkl.iis.u-tokyo.ac.jp/KIlab/Welcome.html
5	2002/9/30 00:00:04	3	http://www.yahoo.co.jp/
6	2002/9/30 00:00:05	54	http://weather.crc.co.jp/
2	2002/9/30 00:00:06	11	http://www.data.kishou.go.jp/majji/
3	2002/9/30 00:00:08	34	http://www.kantei.go.jp/new/kousiki/yotsei.html
5	2002/9/30 00:00:07	10	http://search.yahoo.co.jp/bin/search?p=%C5%B7%B5%A4
5	2002/9/30 00:00:10	300	http://www.tkl.iis.u-tokyo.ac.jp/KIlab/Members/members_j.html

Fig. 3. A part of our panel logs (Web access logs)

the corresponding English translation is in the bottom of this figure. If some queries are only available in Japanese, we give a brief English explanation. For example, the query “Mizuho” is a famous Japanese bank.

3.2 Data Sets

Web Access Logs. Our Web access logs, also called “*panel logs*” are provided by *Video Research Interactive Inc.* which is one of Internet rating companies. The collecting method and statistics of this panel logs are described in [6]. Here we give a brief description. The panel logs consist of *user ID*, *access time of Web page*, *reference seconds of Web page*, *URL of accessed Web page* and so on. The data size is 10GB and the number of users is about 10 thousand. Figure 3 shows the details of a part of our panel logs. In this study, we need to extract past queries and related information from the whole panel logs. We notice that the URL from a search engine (e.g., Yahoo!) records the query submitted by a user, as shown in Figure 3(a). We extract the query from the URL, and then the access logs followed this URL in a session are corresponding Web pages browsed by the user. The maximum interval to determine the session boundary is 30 minutes, a well-known threshold [2] such that two continuous accesses within 30 minutes interval are regarded as in a same session.

Japanese Web Page Archive. The large-scale snapshot of a Japanese Web page archive we used was built in February 2002. We crawled 4.5 million Web pages during the panel logs collection period and automatically created 17 hundred thousand communities from one million selected pages [7]. Since the time of crawling the Web pages for the Web communities is during the time of panel logs collection, there are some Web pages which are not covered by the crawling due to the change and deletion of pages accessed by the panels. We did a preliminary analysis that the full path URL overlap between the Web access logs and Web page archive is only 18.8%. Therefore, we chopped URLs to their hostnames and then the overlap increases to 65%.

3.3 Evaluation Method

We invited nine volunteers(users) to evaluate the query recommendation strategies using our system. They are our lab members who usually use search engines to meet their information needs. We also compare these strategies with the query

Table 1. Search queries for evaluation

Test Query	Accessed Web Pages	Group	Test Query	Accessed Web Pages	Group
lottery	891	A	bank	113	C
ring tone	446	B	fishing	64	A
movie	226	C	scholarship	56	B
hot spring	211	A	university	50	C
soccer	202	B			

Table 2. Evaluation results of our query recommendation system

Relevance of queries	Noun space	Community space	URL space
irrelevant	0.037	0.244	0.341
lowly relevant	0.043	0.089	0.107
relevant	0.135	0.131	0.106
highly relevant	0.707	0.480	0.339
relevant and highly relevant	0.843	0.611	0.444
un-judged	0.078	0.056	0.107

recommendation service supplied by Google search engine. Nine test queries used in our evaluation are listed in Table 1. The total number of queries evaluated by users are 540 because they evaluate the top 20 of recommended queries according to their relatedness values on each of the three feature spaces³. To alleviate the workload on an individual user, we divided the nine users to three group (i.e., A, B, and C) as shown in Table 1. We ask each group to give their relevance judgments on three queries. The relevance judgment has five levels, i.e., irrelevant, lowly relevant, relevant, highly relevant, and un-judged.

4 Evaluation Results and Discussions

4.1 Comparisons of Query Recommendation Strategies

The evaluation results of the recommended queries related with all search queries are shown in Table 2 where each value denotes the percentage of a relevance judgment level on each feature space. For example, The value with the “highly relevant” judgment using noun space is 0.707 which means the percentage of the “highly relevant” judgment chosen by the users in all recommended queries⁴ using the noun based strategy.

In Table 2, when the noun space based strategy is applied, there are more recommended queries evaluated “highly relevant” and fewer queries evaluated “irrelevant” than the other two feature spaces. Furthermore, if we combine the “highly relevant” judgments and the “relevant” judgments, we still gain the best results in the noun space while the percentage of the recommended queries judged as “irrelevant” using the URL space (i.e., 0.341) is higher than those using other two spaces. In the community space, there are many recommended queries evaluated “highly

³ 9 search queries * 20 recommended queries * 3 feature spaces = 540 evaluated queries.

⁴ 9 search queries * 20 recommended queries = 180 evaluated queries.

relevant” and “relevant”. Although the community based strategy produces less related queries than the noun based strategy, it is more than the URL based strategy. By using the URL space, the number of recommended queries judged as “irrelevant” is more than that of queries judged as “highly relevant” (e.g., “irrelevant” vs. “highly relevant” = 0.341 vs. 0.339 in Table 2). In general, the noun and community spaces can supply us with satisfactory recommendation while the URL space based strategy cannot stably produce satisfactory queries related to a query.

4.2 Case Study with “Google Suggestion”

We compare the result of our case study in Figure 2 with the query recommendation service provided by *Google Suggestion*. In Figure 2 our system presents some good recommended queries related to *bank* in the noun and community spaces such as “deposit”, “my car loan”, “toto” and so on that are not given by *Google Suggestion*.

5 Conclusions and Future Work

In this paper, we design a query recommendation system based on three feature spaces, i.e., noun space, URL space, and community space, by using large-scale Web access logs and Japanese Web page archive. The experimental results show that the community-based and noun-based strategies can extract more related search queries than the URL-based strategy. We are designing an optimization algorithm for incremental updates of our recommendation data.

References

1. Beeferman, D., Berger, A.L.: Agglomerative clustering of a search engine query log. In: KDD, pp. 407–416 (2000)
2. Catledge, L., Pitkow, J.: Characterizing browsing behaviors on the world-wide web. *Computer Networks and ISDN Systems* 27(6) (1995)
3. Chirita, P.-A., Firan, C.S., Nejdl, W.: Personalized query expansion for the web. In: SIGIR, pp. 7–14 (2007)
4. Cui, H., Wen, J.-R., Nie, J.-Y., Ma, W.-Y.: Query expansion by mining user logs. *IEEE Trans. Knowl. Data Eng.* 15(4), 829–839 (2003)
5. Glance, N.S.: Community search assistant. In: IUI, pp. 91–96 (2001)
6. Otsuka, S., Toyoda, M., Hirai, J., Kitsuregawa, M.: Extracting user behavior by web communities technology on global web logs. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 957–968. Springer, Heidelberg (2004)
7. Toyoda, M., Kitsuregawa, M.: Creating a web community chart for navigating related communities. In: HT, pp. 103–112 (2001)
8. Wen, J.-R., Nie, J.-Y., Zhang, H.: Query clustering using user logs. *ACM Trans. Inf. Syst.* 20(1), 59–81 (2002)
9. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. In: SIGIR, pp. 4–11 (1996)
10. Zhu, Y., Gruenwald, L.: Query expansion using web access log files. In: Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 686–695. Springer, Heidelberg (2005)

Description Logic to Model a Domain Specific Information Retrieval System

Saïd Radhouani¹, Gilles Falquet¹, and Jean-Pierre Chevalletinst²

¹ CUI, University of Geneva, Genève, Switzerland
{radhouani,falquet}@cui.unige.ch

² IPAL-CNRS, Institute for Infocomm Research, Singapore
viscjp@i2r.a-star.edu.sg

Abstract. In professional environments which are characterized by a domain (Medicine, Law, etc.), information retrieval systems must be able to process **precise queries**, mostly because of the use of a specific domain terminology, but also because the retrieved information is meant to be part of the professional task (a diagnosis, writing a law text, etc.). In this paper we address the problem of solving domain-specific precise queries. We present an information retrieval model based on description logics to represent external knowledge resources and provide expressive document indexing and querying.

1 Introduction

Information Retrieval Systems (IRS) are nowadays very popular, mainly due to the popularity of the Web. Most IRS on the Web (also called Search Engines) are general purpose, they don't take into account the specificities of the user domain of activity. We think there is a need for domain-adapted IRS: once the document domain is known, certain assumptions can be made, some specific knowledge can be used, and users may then ask much more precise queries than the usual small set of keywords in use for Web search engines.

In this work, we explore the modeling of precise search engines adapted to professional environments which are characterized by a domain: medicine, computer, law, etc. Each domain has its own terminology, i.e. its own set of terms that denote a unique concept in the domain. For example, in the medical domain, "X-ray" means an image obtained by the use of X-ray radiations, whereas in Physics domain, "X-ray" means the radiations only. In addition, users often have precise information needs that correspond to professional tasks such as writing medical reports, writing articles about specific events or situations, exploring a scientific question, etc.

In this context, the qualifier "precise" denotes a query that contains terms from a domain specific terminology and have a non trivial semantic structure. For example, a journalist would like to formulate the following query:

Query 1. Give me documents that deal with "*the US politician who won the 2007 peace Nobel prize*".

The journalist is looking for a *politician* whose *nationality* is *US*. A relevant document can for instance contain the name “Al Gore” without necessarily containing the terms “politician” and “US”. This document may not be found by a system merely based on terms matching. A possible solution is to specify that “politician” and “US” are not the terms the user is looking for, but rather a description of the element of interest. For solving this query, the system needs some **domain knowledge** in order to infer that “Al Gore” *is a* “Politician” and that his *nationality* is “US”. The underlying **query language** must also be able to allow the use of relationships for describing the user information need. Another particular case is the use of operators in the query:

Query 2. Give me “*images with a hip without any pathology*”.

A relevant answer to this query must contain the *hip* and must not contain any *pathology*¹ affecting it. A relevant document may contain a *hip* without pathology together with other parts of the human anatomy affected by pathologies. For this reason, the retrieval process must ensure, that only documents containing hip affected by pathologies are excluded. This can be expressed by using a semantic relationship between the query descriptors: “hip” *affected_by* “pathology”. We need domain **knowledge** during the indexing process to precisely describe the documents’ content and we also need a **document language** able to allow this kind of description.

Regarding the requirements we have presented, an IR model capable to solve precise queries must involve the following interdependent components:

External resource: Solving precise queries requires domain knowledge, notably its specialised terminology and its semantic relationships. This knowledge must be expressed in a knowledge representation language and stored in an external² resource such as an ontology;

Expressive document language: In order to allow retrieving documents for precise queries, we need an expressive document language which allows to incorporate semantic relationships and specialised terminology within their content description;

Expressive query language: The expression of precise queries requires a query language which allows the user to explicitly use: *i*) the specialized terminology of his domain of interest; *ii*) the semantic relationships between his query descriptors and *iii*) the desired operators.

The rest of this paper is structured as follows: In Section 2, we will present the most significant approaches that use domain knowledge for information retrieval (IR). Section 3 will be dedicated to the knowledge formalism we chose for our modelling. In Section 4, we will define our IR model presenting the document model and the query model in detail. Section 5 presents our conclusions and perspectives to develop the proposed approach.

¹ No dislocation, no fracture, etc.

² “External” because it models knowledge which are not present in the documents (queries) to be processed, at least in an explicit and complete form.

2 External Resource Based Information Retrieval

There are mainly two categories of approaches that use ERs for IR: *conceptual indexing* [1][2][3] and *query expansion* [4][5][6]. Both of them require a disambiguation step to identify, from the ER, the concepts denoted by the words within the document and the query [7][8].

The conceptual indexing consists in representing documents (queries) by concepts instead of words [9][10][11]. Thus, during the retrieval process, the matching between a query and a document is done based on a non-ambiguous vocabulary (concepts). So far, the approaches based on this technique have not shown significant improvement in terms of retrieval performance [9][12]. One of the factors on which depends the retrieval performance is the method used to “interpret” the semantic document (query) content. In existing approaches, once the concepts have been extracted, the documents (queries) are considered as “bags of concepts”. Therefore, the semantic relationships that may exist between the concepts they contain cannot be exploited. Consequently, the documents dealing with a subject close to that of the query could not be found with these approaches. Some works have shown interest in the representation of documents by semantic networks that connect the concepts of the same document. However, these networks are only used for disambiguation and not during the IR process [9]. The query expansion is a possible solution to this problem [5][6][13].

The idea behind query expansion is to use semantic relationships in order to enrich the query content by adding, from the ER, concepts that are semantically related to those of the query [5][6][13][14]. Several works analysed this aspect, but few have had positive results. In response to these failures, researchers proposed to extend the queries in a “careful” manner by selecting some specific relationships during the expansion process [4][9]. This manner allowed to improve the retrieval performance [9], but the extended queries are again considered as bags of concepts, and their structure is ignored during the retrieval process.

The existing approaches seem to be insufficient considering the requirements that we have presented. Indeed, they treat documents and queries as bags of concepts and do not sufficiently consider their structure. They are therefore incapable to solve precise queries which have complex semantic structures.

3 Formalism for Knowledge Representation

Several formalisms have been used in the IR modeling, notably Semantic Trees[15], Conceptual Graphs[16] and Description Logics (DLs)[17]. Taking into account our requirements, we found out that DLs are particularly appropriate for modeling in our context. Indeed, DLs allow to represent the three sources of knowledge (documents, queries and ER) with the same formalism, which ensures that all these sources can participate in the IR process in a uniform and effective way. This formalism provides also a high level of expressiveness, which is particularly suitable for the representation of precise information needs. Finally it offers a comparison operation that can implement the matching function of the IRS.

Description logics [18][19] form a family of knowledge representation formalisms based on logic. The basic notions of DL are atomic *concepts* and atomic *roles*. The concepts are interpreted as subsets of the individuals that constitute the domain to be modelled. The roles, are interpreted as binary relationships between individuals. Each DL is characterised by *constructors* provided for defining complex concepts (resp. roles) from atomic concepts (roles).

Semantics. An *interpretation* I of a DL vocabulary (a set of atomic concepts and atomic roles) is a pair $(\Delta^I, \cdot^I)$ where Δ^I is a non-empty set called the *domain of discourse* of I , and $\cdot^I$ is a function which associates to each concept C a set $C^I \subseteq \Delta^I$, and to each role R , a binary relationship $R^I \subseteq \Delta^I \times \Delta^I$.

According to our model's requirements, we chose from existing DLs the *Attributive Language with Complements and Qualified number restrictions* ($\mathcal{ALCQ}$) language. The syntax and the semantic of the $\mathcal{ALCQ}$ language are presented in table 1. Given an atomic concept c , an atomic role R and the concept descriptions C and D , the interpretation of a complex concept is defined in table 1.

Table 1. Syntax and semantic of the $\mathcal{ALCQ}$ language

Syntax	Semantic
c	c^I
$\top$	Δ^I
$\neg C$	$\neg C^I = \Delta^I \setminus C^I$
$\perp$	$\emptyset$
$C \sqcap D$	$C^I \cap D^I$
$C \sqcup D$	$C^I \cup D^I$
$\forall R.C$	$\{d \in \Delta^I \mid \forall e \in \Delta^I (R^I(d, e) \rightarrow e \in C^I)\}$
$\exists R.C$	$\{d \in \Delta^I \mid \exists e \in \Delta^I (R^I(d, e), e \in C^I)\}$
$\geq nR.C$	$\{d \in \Delta^I \mid \{e \in \Delta^I \mid R^I(d, e), e \in C^I\} \geq n\}$
$\leq nR.C$	$\{d \in \Delta^I \mid \{e \in \Delta^I \mid R^I(d, e), e \in C^I\} \leq n\}$

A DL knowledge base is comprised of a terminological component, the *TBox*, and an assertional component, the *ABox*. The *TBox* is made of *general concept inclusion (GCI) axioms* of the form $C \equiv D$ or $C \sqsubseteq D$ where C and D are two concept expressions. For instance,

$$Parent \equiv Person \sqcap \exists hasChild. Person.$$

The ABox contains assertions of the form $C(a)$ and $R(a, b)$ where C is a concept and a and b are individual identifiers. For instance

$$Person(Jacques), Person(Maria), hasChild(Jacques, Maria)$$

Subsumption. An interpretation I satisfies the GCI $C \sqsubseteq D$ if $C^I \subseteq D^I$. I satisfies the *TBox* T , if I satisfies all GCIs in T . In this case, I is called *model* of T . A concept D **subsumes** a concept C in T if $C \sqsubseteq D$ in every model I of T .

What makes many description logics particularly appealing is the decidability of the subsumption problem, i.e. the existence of algorithms that test if a concept subsumes another one.

4 Semantic Descriptors-Based Information Retrieval Model

We showed in Section 2 that approaches which consider documents (queries) as bags of concepts are insufficient to solve precise queries. Thus we propose to use DL expressions to represent documents and in particular the relationships that exist between the elements of a document.

4.1 The Semantic Descriptor: A New Indexing Unit

Any concept from the knowledge base may constitute a semantic descriptor when it is used withing a document (query). A semantic descriptor is an $\mathcal{ALCQ}$ expression which is intended to match as precisely as possible the concept to which it is referred to in the document (query). This expression is a conjunction of which at least one concept serves to identify the semantic descriptor. It can also contain other concepts which serve to “refine” the description of the semantic descriptor in question. Formally, a semantic descriptor S is of the form:

$$S \equiv d_{idf} \sqcap \exists \textit{described\_by}.C_1 \sqcap \dots \sqcap \exists \textit{described\_by}.C_n$$

where c_{idf} is the identifying concept and $C_1, \dots, C_n$ are the refining concepts.

The name *described_by* represents a generic relationship; in practical applications it will be replaced by a relationship (role) of the knowledge base.

Example: In a document containing “The Brazilian Minister of Sports Pelé”, the semantic descriptor is identified by “Pelé” and described by “Minister of Sports” and “Brazil”. Formally, this semantic descriptor is of the form:

$$S \equiv \textit{Pelé} \sqcap \exists \textit{Occupation.Minister\_of\_Sports} \sqcap \exists \textit{Nationality.Brazil}$$

4.2 Document and Query Representation

Each document doc (query q) is represented by a concept R_{doc} (R_q) defined by the conjunction of the semantic descriptors belonging to doc (q). In order to represent the documents and the queries using semantic descriptors, we propose to use the role *indexed_by*, which allows to associate a semantic descriptor S to a given document (query) doc (q) to be indexed (solved). Formally, the representation R of a given document or query containing the semantic descriptors $\{S_1 \dots S_n\}$ is an $\mathcal{ALCQ}$ expression of the form:

$$R \equiv \exists \textit{indexed\_by}.S_1 \sqcap \dots \sqcap \exists \textit{indexed\_by}.S_n$$

After the indexing process, the documents index is comprised of the original *TBox* extended by the R_{doc} concepts. During the querying process, the *TBox* is extended by the concept R_q .

Examples: Query 2 (Section 1) contains two semantic descriptors (*hip*, *pathology affecting a hip*) and a negation (*without*). It is represented by:

$$R_{Q2} \equiv \exists indexed\_by.Hip \sqcap \neg \exists indexed\_by.(Pathology \sqcap \exists affect.Hip)$$

The query “Give me an image containing Zidane **alone**” can be represented by

$$R_{Q3} \equiv \exists indexed\_by.Martin\_Luther\_King \sqcap = 1 indexed\_by.Person$$

Retrieval Process: The retrieval process consists in selecting the documents that satisfy the query requirements. In DL terms, the retrieval process can be seen as a task to retrieve those documents represented by concepts that are subsumed by the concept representing the corresponding query. Thus, the matching between a query q and a document doc is done by verifying that $R_{doc} \sqsubseteq R_q$ is true within the knowledge base. Finally, the set of relevant documents for a given query q is $\{doc \mid R_{doc} \sqsubseteq_T R_q\}$.

The design of the used ER has a major impact on search result. Indeed, the matching function based on the calculation of the subsumption can be very beneficial when the ER is rich in terms of *is-a* relationship. Indeed, through the algorithm that computes the subsumption, the use of DL offers a capacity of reasoning that can deduce implicit knowledge from those given explicitly in the TBox, and therefore help to retrieve relevant documents for a given query even if they do not share any words with it. However, using only the subsumption has some limits. Indeed, depending on the domain, the ER may be organized according to different semantic hierarchies. For instance, in the geographic domain, the geometric containment is probably one of the most important hierarchical relationship. The same is true for human anatomy. For example, if a user looks for a *fracture in the leg*, he or she will certainly consider a document dealing with a *pathology of the tibia* as relevant. Thus the retrieval system must take into account the *part_of* hierarchy that exists within the human anatomy. One way to solve this problem is to twist the subsumption relation and to represent the *part_of* hierarchy as a subsumption hierarchy. Thus implicitly stating, for instance, that *a tibia is a leg*. In this approach, a query

$$R_q \equiv \exists indexed\_by.(Fracture \sqcap \exists location.Leg)$$

will correctly retrieve a document described by

$$R_{doc} \equiv \exists indexed\_by.(Fracture \sqcap \exists location.Tibia)$$

because $R_{doc} \sqsubseteq R_q$ if $Tibia \sqsubseteq Leg$.

Using subsumption to mimic another relation may lead, in certain circumstances, to unexpected and counter-intuitive deductions. A “cleaner” and semantically safer approach consists in defining transitive properties to represent the

various types of hierarchies that may exist in a given domain. The above example would then lead to the following descriptors:

$$R_q \equiv \exists \textit{indexed\_by} . (\textit{Fracture} \sqcap \exists \textit{location} . (\exists \textit{part\_of} . \textit{Leg}))$$

$$R_{doc} \equiv \exists \textit{indexed\_by} . (\textit{Fracture} \sqcap \exists \textit{location} . \textit{Tibia})$$

If an axiom specifies that *part_of* is transitive and the definition of *Tibia* is of the form “... $\sqcap \exists \textit{part\_of} . \textit{Leg}$ ”, then the reasoner will infer that $R_{doc} \sqsubseteq R_q$.

5 Conclusion

In order to solve precise queries, we proposed an information retrieval model based on a new indexing unit: the *semantic descriptor*. A semantic descriptor is defined by concepts and relationships, and serves to describe the semantic documents and queries content. We defined our model using the *Description Logic*, which allows a uniform precise representation of documents and queries.

In order to assess the feasibility of our approach, we conducted some experiences (not described here) on a medical document collection. The obtained results are very promising and confirmed that the use of DL has a very good impact on the retrieval performance. Indeed, the DL offers the opportunity to use background knowledge about a specific domain. Thus, during the querying process we can benefit from the powerful reasoning capabilities a reasoner offers, notably the capacity to deduce the implicit knowledge from knowledge explicitly given in the *TBox*.

It is obvious that using DL reasoners to perform IR tasks leads to performances that are several orders of magnitude slower than classical index-based IRS. Nevertheless, several issues could be worth studying to improve the DL approach performances: *i*) document descriptors are generally simple (limited to $\sqcap$ and $\exists$ constructors), thus we could devise simpler reasoning algorithms, *ii*) when queries are simple, reasoning becomes even simpler and *iii*) the document corpus is generally stable and could be pre-processed in some way to facilitate the reasoner’s work.

Acknowledgments. The authors would like to thank Mathieu Vonlanthen for fruitful discussions about the use of DL reasoners to implement subsumption-based information retrieval.

References

1. Biemann, C.: Semantic indexing with typed terms using rapid annotation. In: Proceedings of the TKE 2005-Workshop on Methods and Applications of Semantic Indexing, Copenhagen (2005)
2. Mihalcea, R., Moldovan, D.: Semantic indexing using wordnet senses. In: Proceedings of the ACL-2000 workshop on Recent advances in natural language processing and information retrieval, Morristown, NJ, USA, pp. 35–45. Association for Computational Linguistics (2000)

3. Vallet, D., Fernández, M., Castells, P.: An ontology-based information retrieval model. In: Gómez-Pérez, A., Euzenat, J. (eds.) *ESWC 2005*. LNCS, vol. 3532, pp. 455–470. Springer, Heidelberg (2005)
4. Qiu, Y., Frei, H.P.: Concept based query expansion. In: Korfhage, R., Rasmussen, E.M., Willett, P. (eds.) *SIGIR*, pp. 160–169. ACM, New York (1993)
5. Voorhees, E.M.: Query expansion using lexical-semantic relations. In: *SIGIR 1994: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 61–69. Springer, Heidelberg (1994)
6. Baziz, M., Aussenac-Gilles, N., Boughanem, M.: Désambiguisation et Expansion de Requêtes dans un SRI, Etude de l'apport des liens sémantiques. *Revue des Sciences et Technologies de l'Information (RSTI) série ISI* 8, 113–136 (2003)
7. Krovetz, R., Croft, W.B.: Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems* 10, 115–141 (1992)
8. Sanderson, M.: Word Sense Disambiguation and Information Retrieval. Ph.d. thesis, University of Glasgow, Glasgow G12 8QQ, UK (1997)
9. Baziz, M.: Indexation conceptuelle guidée par ontologie pour la recherche d'information. Thèse de doctorat, Université Paul Sabatier, Toulouse, France (2005)
10. Smeaton, A., Quigley, I.: Experiments on using semantic distances between words in image caption retrieval. In: *Proc. of 19th International Conference on Research and Development in Information Retrieval*, Zurich, Switzerland (1996)
11. Uzuner, ö., Katz, B., Yuret, D.: Word sense disambiguation for information retrieval. In: *AAAI/IAAI*, p. 985 (1999)
12. Voorhees, E.M.: Natural language processing and information retrieval. In: Pazienza, M.T. (ed.) *SCIE 1999*. LNCS (LNAI), vol. 1714, pp. 32–48. Springer, Heidelberg (1999)
13. Mihalcea, R., Moldovan, D.I.: An iterative approach to word sense disambiguation. In: *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference*, pp. 219–223. AAAI Press, Menlo Park (2000)
14. Baziz, M., Boughanem, M., Aussenac-Gilles, N., Chrisment, C.: Semantic cores for representing documents in ir. In: *SAC 2005: Proceedings of the 2005 ACM symposium on Applied computing*, pp. 1011–1017. ACM, New York (2005)
15. Berrut, C.: Une méthode d'indexation fondée sur l'analyse sémantique de documents spécialisés. Le prototype RIME et son application à un corpus médical. Thèse de doctorat, Université Joseph Fourier, Grenoble, France (1988)
16. Chevallet, J.P.: Un Modèle Logique de Recherche d'Informations appliqué au formalisme des Graphes Conceptuels. Le prototype ELEN et son expérimentation sur un corpus de composants logiciels. PhD thesis, Université Joseph Fourier, Grenoble (1992)
17. Meghini, C., Sebastiani, F., Straccia, U., Thanos, C.: A model of information retrieval based on a terminological logic. In: *SIGIR 1993: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 298–307. ACM, New York (1993)
18. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York (2003)
19. Brachman, R.J., Schmolze, J.G.: An overview of the kl-one knowledge representation system. In: Mylopoulos, J., Brodie, M.L. (eds.) *Artificial Intelligence & Databases*, pp. 207–230. Kaufmann Publishers, San Mateo (1989)

Extending the Edit Distance Using Frequencies of Common Characters

Muhammad Marwan Muhammad Fuad and Pierre-François Marteau

VALORIA, Université de Bretagne Sud

BP. 573, 56017 Vannes, France

{marwan.fuad, pierre-francois.marteau}@univ-ubs.fr

Abstract. Similarity search of time series has attracted many researchers recently. In this scope, reducing the dimensionality of data is required to scale up the similarity search. Symbolic representation is a promising technique of dimensionality reduction, since it allows researchers to benefit from the richness of algorithms used for textual databases. To improve the effectiveness of similarity search we propose in this paper an extension to the edit distance that we call the extended edit distance. This new distance is applied to symbolic sequential data objects, and we test it on time series data bases in classification task experiments. We also prove that our distance is a metric.

Keywords: Time Series, Symbolic Representation, the Edit Distance.

1 Introduction

Similarity search is an important problem in computer science, and it has a large number of applications. Research in this area has focused on its different aspects. One of these aspects is the distance metric used to measure the similarity between two data objects. Another aspect of this problem is the so called “dimensionality curse”. One of the best solutions to deal with dimensionality curse is to utilize a dimensionality reduction technique to reduce dimensionality, then to utilize a suitable indexing structure on the reduced data objects. There have been different suggestions to represent time series, to mention a few; *DFT* [1] and [2], *DWT* [03], *SVD* [7], *APCA* [6], *PAA* [5] and [11], *PLA* [9]...etc.. However, among dimensionality reduction techniques, symbolic representation has many interesting advantages; it allows using text-retrieval algorithms and techniques [6]. In the beginning distance measures available for symbolic data processing were restricted to data structures whose representation is naturally symbolic (DNA and protein sequences, textual data...etc). But later these symbolic measures were also applied to other data structures that can be transformed into strings by using some symbolic representation techniques. There are quite a few distance metrics that apply to symbolically represented data. One of these measures is the edit distance (*ED*) [10], which is defined as the minimum number of delete, insert, and substitute (change) operations needed to transform string *S* into string *T*. Other measures for sequence alignment were proposed. The edit distance has a main drawback; it penalizes all change operations in the same way, without taking into account

the character that is used in the change operation. In order to overcome this drawback we could predefine cost functions that gave the costs of all possible change operations. But this approach is inflexible and highly dependent on the application and corresponding alphabet.

In this paper we propose a new general distance metric that applies to strings. We call it “The Extended Edit Distance” (*EED*). This distance adds new features to the well-known edit distance by adding an additional term to it. The new distance has a main advantage over the edit distance in that it deals with the above mentioned problem straightforwardly, since there is no need to predefine a cost function for the change operation. This distance can, by itself, detect if the change operations use characters that are “familiar” or “unfamiliar” to the two strings concerned.

The rest of this paper is organized as follows: in section 2 we present a motivating example followed by the *EED*. Section 3 contains the experiments that we conducted, we discuss the results in section 4, and conclude in section 5 with some perspectives.

2 EED

2.1 Motivating Example

Given the string $S_1 = \text{marwan}$, by performing two change operations in the first and fifth positions we obtain the string $S_2 = \text{aarwin}$. By calculating their edit distance we get; $ED(S_1, S_2) = 2$. Let NDC be the number of distinct characters that two strings contain, i.e. $NDC(S_1, S_2) = |\{ch(S_1)\} \cup \{ch(S_2)\}|$, where $ch(\)$ is the set of characters that a string consists of. In our example we have; $NDC(S_1, S_2) = 6$. Now if we change the same positions in S_1 with different characters b, e we obtain the string: $S_3 = \text{barwen}$. By calculating the edit distance we get; $ED(S_1, S_3) = 2$ (which is the same as $ED(S_1, S_2)$). But we notice that $NDC(S_1, S_3) = 7$. This means that one change operation used a character that is more “familiar” to the two strings in the first case than in the second case, in other words, S_2 is closer to S_1 than S_3 . But the edit distance couldn’t recognize this, since the edit distance was the same in both cases.

2.2 Definition-The Extended Edit Distance

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ . Let $f_a^{(S)}, f_a^{(T)}$ be the frequency of the character a in S and T , respectively. Where S, T are two strings in Σ^* . The extended edit distance (*EED*) is defined as;

$$EED(S, T) = ED(S, T) + \lambda \left[|S| + |T| - 2 \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(T)}) \right]$$

Where $|S|, |T|$ are the lengths of the two strings S, T respectively, and where $\lambda \geq 0$ ($\lambda \in R$). We call λ the co-occurrence frequency factor.

Revisiting the example presented in section 2.1 we see that $EED(S_1, S_2) = 4$, $EED(S_1, S_3) = 6$; which is what we expected, since, according to the concept of similarity we presented in section 2.1, S_2 is more similar to S_1 than S_3 .

2.3 Proposition (P1): EED Is a Distance Metric

Let D be a set of objects. A function $d : D \times D \rightarrow \mathbb{R}^+$ is called a distance metric if the following holds $\forall x, y, z \in D$:

$$(p1) d(x, y) = d(y, x), (p2) x = y \Leftrightarrow d(x, y) = 0, (p3) d(x, z) \leq d(x, y) + d(y, z).$$

We prove below that EED is a metric.

(p1): $EED(S, T) = EED(T, S)$ (this is obvious).

(p2): Since for all S in Σ^* we have $|S| = \sum_{a \in \Sigma} f_a^{(S)}$ we can easily verify that:

$$\lambda \left[|S| + |T| - 2 \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(T)}) \right] \geq 0 \quad \forall S, T \quad (1)$$

Let's prove first that $EED(S, T) = 0 \Rightarrow S = T$:

If $EED(S, T) = 0$, and taking into account (1), we get the two following relations:

$$\lambda \left[|S| + |T| - 2 \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(T)}) \right] = 0 \quad (2)$$

$$ED(S, T) = 0 \quad (3)$$

From (3), and since ED is metric we get: $S = T$. The backward proposition $S = T \Rightarrow EED(S, T) = 0$ is obvious.

(p3): $EED(S, T) \leq EED(S, R) + EED(R, T)$

$\forall S, T, R$ in Σ^* . Since ED is metric, we have: $ED(S, T) \leq ED(S, R) + ED(R, T)$ (4)

Let $D(S, T) = |S| + |T| - 2 \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(T)})$. We have to show that for all S, T, R in Σ^* :

$$\lambda \cdot D(S, T) \leq \lambda \cdot D(S, R) + \lambda \cdot D(R, T) \quad (5)$$

First, we note that the following equivalences hold:

$$\begin{aligned} \lambda \cdot D(S, T) &\leq \lambda \cdot D(S, R) + \lambda \cdot D(R, T) \Leftrightarrow |S| + |T| - 2 \cdot \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(T)}) \\ &\leq |S| + |R| - 2 \cdot \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(R)}) + |R| + |T| - 2 \cdot \sum_{a \in \Sigma} \min(f_a^{(R)}, f_a^{(T)}) \\ &\Leftrightarrow \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(R)}) + \sum_{a \in \Sigma} \min(f_a^{(R)}, f_a^{(T)}) \leq |R| + \sum_{a \in \Sigma} \min(f_a^{(S)}, f_a^{(T)}) \end{aligned}$$

Since $|R| = \sum_{a \in \Sigma} f_a^{(R)}$, proving (5) is equivalent to proving (6):

$$\sum_{a \in \Sigma} \text{Min}(f_a^{(S)}, f_a^{(R)}) + \sum_{a \in \Sigma} \text{Min}(f_a^{(R)}, f_a^{(T)}) \leq \sum_{a \in \Sigma} f_a^{(R)} + \sum_{a \in \Sigma} \text{Min}(f_a^{(S)}, f_a^{(T)}) \quad (6)$$

Second, we note that for all a in Σ we have:

$$\text{Min}(f_a^{(S)}, f_a^{(R)}) \leq f_a^{(R)} \quad \text{and} \quad \text{Min}(f_a^{(T)}, f_a^{(R)}) \leq f_a^{(R)}$$

Furthermore, for all a in Σ we have: Either

$$f_a^{(R)} = \text{Min}(f_a^{(R)}, f_a^{(S)}, f_a^{(T)}) \Rightarrow \text{Min}(f_a^{(R)}, f_a^{(T)}) \leq \text{Min}(f_a^{(S)}, f_a^{(T)}) \Rightarrow$$

$$\text{Min}(f_a^{(S)}, f_a^{(R)}) + \text{Min}(f_a^{(R)}, f_a^{(T)}) \leq f_a^{(R)} + \text{Min}(f_a^{(S)}, f_a^{(T)})$$

$$\text{Or } f_a^{(S)} = \text{Min}(f_a^{(R)}, f_a^{(S)}, f_a^{(T)}) \Rightarrow \text{Min}(f_a^{(S)}, f_a^{(R)}) = \text{Min}(f_a^{(S)}, f_a^{(T)}) \Rightarrow$$

$$\text{Min}(f_a^{(S)}, f_a^{(R)}) + \text{Min}(f_a^{(R)}, f_a^{(T)}) \leq f_a^{(R)} + \text{Min}(f_a^{(S)}, f_a^{(T)})$$

$$\text{Or } f_a^{(T)} = \text{Min}(f_a^{(R)}, f_a^{(S)}, f_a^{(T)}) \Rightarrow \text{Min}(f_a^{(R)}, f_a^{(T)}) = \text{Min}(f_a^{(S)}, f_a^{(T)}) \Rightarrow$$

$$\text{Min}(f_a^{(S)}, f_a^{(R)}) + \text{Min}(f_a^{(R)}, f_a^{(T)}) \leq f_a^{(R)} + \text{Min}(f_a^{(S)}, f_a^{(T)})$$

This shows that, for all a in Σ , the following inequality holds:

$$\text{Min}(f_a^{(S)}, f_a^{(R)}) + \text{Min}(f_a^{(R)}, f_a^{(T)}) \leq f_a^{(R)} + \text{Min}(f_a^{(S)}, f_a^{(T)})$$

Summing over all a in Σ we get a proof for proposition (6) and consequently a proof for proposition (5). Adding (4) and (5) side to side we get (p3): $EED(S, T) \leq EED(S, R) + EED(R, T)$. From (p1), (p2), and (p3) we get (P1) and conclude that EED is a metric.

3 Empirical Evaluation

We conducted four experiments of times series classification task based on the 1-NN rule on the datasets available at *UCR* [12]. We used leaving-one-out cross validation. As mentioned earlier, our new distance is applied to data structures which are represented symbolically. Time series are not naturally represented symbolically, but more and more studies focus on symbolic representation of time series. One of the most famous methods in the literature is *SAX* [4]. *SAX*, in simple words, consists of three steps; 1-Reducing the dimensionality of the time series by using piecewise aggregate approximation *PAA* 2-Discretization the *PAA* to get a discrete representation of the times series 3-Using the *MINDIST* measure. To test EED (or ED) we proceeded in the same way for steps 1 and 2 above to get a symbolic representation of time series, then in step 3 we compared EED with ED and with the distance measure defined in *SAX*, all applied to the resulting strings.

3.1 The First Experiment

The aim of the this experiment is to make a direct comparison among ED , EED and *SAX*. For this experiment, we used the same compression ratio that was used to test *SAX* (i.e. 1 to 4). We also used the same range of alphabet size (3-10).

For each dataset we tune the parameters on the training set to get the optimal values of these parameters; i.e. the values that minimize the error. Then we utilize these optimal values on the testing set to get the error rate for each method and for each dataset. As for parameter λ , for simplicity, and in all the experiments we conducted, we optimized it in the interval $[0, 1]$ only ($step=0.25$), except in the cases where there was strong evidence that the error was decreasing monotonously as λ increased.

For this experiment, we chose, at random, 4 datasets from the 20 datasets of UCR [12]. The chosen datasets were *CBF*, *Trace*, *Two_Patterns*, *Yoga*. After optimizing the parameters on the training sets, we used these parameters on the testing sets of these datasets; we got the results shown in Table. 1. (The best method is highlighted)

Table 1. The error rate of *ED*, *EED*, *SAX* on the testing sets of *CBF*, *Trace*, *Two Patterns*, and *Yoga*. The parameters used in the calculations are those that give optimal results on the training sets, the alphabet size was chosen from the interval $[3, 10]$. The compression ratio is (1:4).

	The Edit Distance (ED)	The Extended Edit Distance (EED)	SAX
CBF	0.029 $\alpha^*=10$	0.026 $\alpha=3, \lambda=0.75$	0.104 $\alpha=10$
Trace	0.11 $\alpha=10$	0.07 $\alpha=6, \lambda \geq 1.25$	0.42 $\alpha=10$
Two_Patterns	0.015 $\alpha=3$	0.015 $\alpha=3, \lambda=0$	0.081 $\alpha=10$
Yoga	0.155 $\alpha=7$	0.155 $\alpha=7, \lambda=0$	0.199 $\alpha=10$
MEAN	0.077	0.066	0.201
STD	0.067	0.064	0.155

(*: α is the alphabet size)

The results obtained show that *EED* was always better, or equal, to the other methods. Its average error is the smallest. The results also show that of all the three tested methods *EED* has the minimum standard deviation

3.2 The Second Experiment

This experiment is an extension of the first experiment; we didn't compare our new distance with *ED* and *SAX* only, but we also compared it with other distances that are applied for non-compressed time series. We chose the two most famous distances; *Dynamic Time Warping (DTW)* [7] and *Euclidean distance*. We chose randomly 4 datasets of the remaining datasets in UCR [12]. These were *Gun_Point*, *OSU Leaf*, *50words*, and *Fish*. We used the same compression ratio and the same range of alphabet size that we used with in the first experiment. We proceeded in the same way. We obtained the results shown in Table. 2.

Table 2. The error rate of *ED*, *EED*, *SAX*, *DTW* together with the *Euclidean distance* on the testing sets of *Gun_Point*, *OSU Leaf*, *50words*, and *Fish*. The alphabet size was chosen from the interval $[3, 10]$. The compression ratio is $(1:4)$.

	Euclidean Distance	DTW	ED	EED	SAX
Gun-Point	0.087	0.093	0.073 $\alpha=4$	0.06 $\alpha=4, \lambda=0.25$	0.233 $\alpha=10$
OSULeaf	0.483	0.409	0.318 $\alpha=5$	0.293 $\alpha=5, \lambda=0.75$	0.475 $\alpha=9$
50words	0.369	0.310	0.266 $\alpha=7$	0.266 $\alpha=7, \lambda=0$	0.327 $\alpha=9$
Fish	0.217	0.267	0.149 $\alpha=10$	0.149 $\alpha=10, \lambda=0$	0.514 $\alpha=10$
MEAN	0.289	0.270	0.201	0.192	0.387
STD	0.173	0.132	0.111	0.108	0.131

The results of this experiment show that *EED* is superior to the other distances.

3.3 The Third Experiment

This experiment aims at studying the impact of using a wider range of alphabet size; $[3, 20]$, we proceed in the same way we did before; we randomly chose 7 datasets of the remaining datasets. The 7 chosen datasets were *Coffee*, *Beef*, *Adiac*, *ECG200*, *Wafer*, *Swedish Leaf*, *Face (all)*. The compression ratio is the same as before $(1:4)$.

Table 3. The error rate of *ED*, *EED*, *SAX* on the testing sets of *Coffee*, *Beef*, *Adiac*, *ECG200*, *Wafer*, *Swedish Leaf*, and *Face (all)*. The alphabet size was chosen from the interval $[3, 20]$. The compression ratio is $(1:4)$.

	The Edit Distance (ED)	The Extended Edit Distance (EED)	SAX
Coffee	0.071 $\alpha=12, 13$	0.0 $\alpha=14, \lambda=0.25$	0.143 $\alpha=20$
Beef	0.467 $\alpha=17$	0.4 $\alpha=4, \lambda=0.75$	0.433 $\alpha=20$
Adiac	0.555 $\alpha=18$	0.524 $\alpha=19, \lambda=1$	0.867 $\alpha=18$
ECG200	0.23 $\alpha=13$	0.19 $\alpha=5, \lambda=0.25$	0.13 $\alpha=16$
Wafer	0.008 $\alpha=4$	0.008 $\alpha=4, \lambda=0$	0.004 $\alpha=19$
Swedish Leaf	0.344 $\alpha=4$	0.365 $\alpha=7, \lambda=0.25$	0.253 $\alpha=20$
Face (all)	0.324 $\alpha=7$	0.324 $\alpha=7, \lambda=0$	0.305 $\alpha=19$
MEAN	0.286	0.257	0.305
STD	0.199	0.200	0.284

EED was compared with *ED* and *SAX*. The final results on the testing sets are shown in Table. 3.

The results of this experiment show that for this range of alphabet size, the average error of the *EED* is the smallest. The standard deviation for *ED* for this range size is the smallest. However, it's very close to that of *EED*.

3.4 The Fourth Experiment

This experiment is designated to study the impact of using a different compression ratio. We conducted it on the rest of the datasets in *UCR* [12]. The compression ratio of this experiment is (1:5). The alphabet range is [3, 10]. After proceeding in the same way that we used for the other experiments we got the results shown in Table. 4

Table 4. The error rate of *ED*, *EED*, *SAX* on the testing sets of *Lighting2*, *Lighting7*, *Synthetic Control*, *Face (four)*, *Trace*, and *Olive Oil* the alphabet size was chosen from the interval [3,10]. The compression ratio is (1:5)

	The Edit Distance (ED)	The Extended Edit Distance (EED)	SAX
Lighting2	0.311 $\alpha = 5$	0.311 $\alpha = 5, \lambda = 0.75$	0.377 $\alpha = 3$
Lighting7	0.247 $\alpha = 5$	0.247 $\alpha = 5, \lambda = 0$	0.479 $\alpha = 7$
Trace	0.11 $\alpha = 10$	0.09 $\alpha = 8, \lambda = 0.75$	0.36 $\alpha = 10$
Synthetic Control	0.077 $\alpha = 8$	0.05 $\alpha = 6, \lambda = 0.25$	0.03 $\alpha = 10$
Face (four)	0.045 $\alpha = 5, 6$	0.045 $\alpha = 5, 6, \lambda = 0$	0.182 $\alpha = 9$
Olive Oil	0.267 $\alpha = 7$	0.267 $\alpha = 7, \lambda = 0, \dots, 1$	0.833 $\forall \alpha$
MEAN	0.176	0.168	0.377
STD	0.112	0.120	0.275

The results obtained show that *EED* was the best in almost all the datasets used in this experiment. The average error of *EED* is the smallest. However, the standard deviation for *ED* for this compression ratio is the smallest.

4 Discussion

In the experiments we conducted we had to use time series of equal lengths for comparison reasons only, since *SAX* can be applied only to strings of equal lengths. But *EED* (and *ED*, too) can be applied to strings of different lengths. We also didn't conduct experiments for alphabet size=2 because *SAX* is not applicable in this case (when *alphabet size* =2 then the distance between any two strings will be zero with *SAX*, and for any dataset). However, it's important to mention that comparing *EED* or *ED*, with *SAX* was only used as an indicator of performance. In fact, *SAX* is faster than any of

EED or *ED*, even though the error it produces is greater in most cases than that of *EED* or *ED*.

In order to represent the time series symbolically, we had to use a technique prepared for *SAX* for comparison purposes. Nonetheless, a representation technique prepared specifically for *EED* may even give better results.

The main property of the *EED* over *ED* is that it is more precise, since it considers a global level of similarity that *ED* doesn't consider

5 Conclusion and Perspectives

In this paper we presented a new distance metric applied to strings. The main feature of this distance is that it considers the frequency of characters, which is something other distance measures do not consider. Another important feature of this distance is that it's metric. We tested this new distance on a time series classification task, and we compared it to other distances. We showed that our distance gave better results in most cases. The main drawback of this distance is that it uses the parameter λ , which is heuristic, it also increases the training phase. The future work concerns the elimination of this parameter.

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Proceedings of the 4th Conf. on Foundations of Data Organization and Algorithms (1993)
2. 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)
3. 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, 1999, pp. 126–133 (1999)
4. Lin, J., Keogh, E.J., Lonardi, S., Chiu, B.Y.-c.: A symbolic representation of time series, with implications for streaming algorithms. *DMKD* 2003, 2–11 (2003)
5. 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)
6. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra: Locally adaptive dimensionality reduction for similarity search in large time series databases. *SIGMOD*, 151–162 (2001)
7. Keogh, E.: Exact indexing of dynamic time warping. In: Proc. 28th Int. Conf. on Very Large Data Bases, pp. 406–417 (2002)
8. Kom, 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)
9. 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 PacificAisa Conf. on Knowledge Discovery and Data Mining, pp. 51–60 (2001)
10. Wagner, R.A., Fischer, M.J.: The String-to-String Correction Problem. *Journal of the Association for Computing Machinery* 21(1), 168–173 (1974)
11. Yi, B., K.: Fast time sequence indexing for arbitrary L_p norms. In: Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt (2000)
12. UCR Time Series datasets,
http://www.cs.ucr.edu/~eamonn/time_series_data/

Tracking Moving Objects in Anonymized Trajectories

Nikolay Vyahhi¹, Spiridon Bakiras², Panos Kalnis³, and Gabriel Ghinita³

¹ Dept. of Computer Science, St. Petersburg State University, St. Petersburg, Russia
vyahhi@gmail.com

² Dept. of Mathematics and Computer Science, John Jay College, City University of New York
sbakiras@jjay.cuny.edu

³ Dept. of Computer Science, National University of Singapore, 117590 Singapore
{kalnis, ghinita}@comp.nus.edu.sg

Abstract. Multiple target tracking (MTT) is a well-studied technique in the field of radar technology, which associates anonymized measurements with the appropriate object trajectories. This technique, however, suffers from combinatorial explosion, since each new measurement may potentially be associated with any of the existing tracks. Consequently, the complexity of existing MTT algorithms grows exponentially with the number of objects, rendering them inapplicable to large databases. In this paper, we investigate the feasibility of applying the MTT framework in the context of large trajectory databases. Given a history of object movements, where the corresponding object *ids* have been removed, our goal is to track the trajectory of every object in the database in successive timestamps. Our main contribution lies in the transition from an exponential solution to a polynomial one. We introduce a novel method that transforms the tracking problem into a min-cost max-flow problem. We then utilize well-known graph algorithms that work in polynomial time with respect to the number of objects. The experimental results indicate that the proposed methods produce high quality results that are comparable with the state-of-the-art MTT algorithms. In addition, our methods reduce significantly the computational cost and scale to a large number of objects.

1 Introduction

Recent advances in wireless communications and positioning devices have generated significant interest in the collection of spatio-temporal (i.e., trajectory) data from moving objects. Any GPS-enabled mobile device with sufficient storage and computational capabilities can benefit from a wide variety of location-based services. Such services maintain (at a centralized server) the locations of a large number of moving objects over a long period of time. As an example, consider a traffic monitoring system where each car periodically transmits its exact location to a database server. The resulting trajectories can be queried by a user to retrieve important information regarding current or predicted traffic conditions at various parts of the road network.

Nevertheless, the availability of such data at a centralized location raises concerns regarding the privacy of the mobile clients, especially if the data is distributed to other parties. A simple solution that partially solves this problem is to anonymize the trajectory data, by not publishing the user *id*¹. In the traffic monitoring system, for instance,

¹ Assigning a fake *id* does not guarantee anonymity, since a user may be linked to a specific trajectory using background knowledge (e.g., known home address as starting point).

the *ids* of the individual users are not essential for measuring the traffic level on a road segment. Therefore, the mobile users may not be willing to identify themselves, and may choose to transmit only their location, but not their *id*. Furthermore, anonymous data collection may be the only option in certain environments. For instance, in the traffic monitoring system the trajectory data may be collected by sensors that are deployed throughout a city. In this scenario, every vehicle that passes in front of a sensor automatically generates a measurement that contains no information regarding its identity.

Even though anonymization is important for protecting the privacy of mobile users, detailed trajectory data (i.e., coupled with object identifiers) are valuable in numerous situations. For example, a law enforcement agency trying to track a suspect that was seen in a car at a specific time, can certainly benefit from stored trajectory information. In this scenario, anonymization severely hinders the tracking process, since there is no information to link successive measurements to the same trajectory. A straightforward solution, given the similarity of the two problems, is to leverage existing methods that are used in radar tracking applications. Multiple target tracking (MTT) [1,2] is a well-studied technique in the field of radar technology, which associates anonymized measurements with the appropriate object trajectories. This technique, however, is not practical. The reason is that every possible combination of measurements must be considered, in order to minimize the overall error across all trajectories. Consequently, the complexity of existing MTT algorithms grows exponentially with the number of objects, rendering them inapplicable to large databases.

In this paper, we investigate the feasibility of applying the MTT framework in the context of large trajectory databases. Given a history of object movements, where the corresponding object *ids* have been removed, our goal is to track the trajectory of every object in the database in successive timestamps. Our main contribution lies in the transition from an exponential solution to a polynomial one. To this end, we introduce a novel method that transforms the tracking problem into a min-cost max-flow problem. We then utilize well-known graph algorithms that work in polynomial time with respect to the number of objects. To further reduce the computational cost, we also implement a pruning step prior to the construction of the flow network. The objective is to remove all the measurement associations that are not feasible (e.g., due to a maximum velocity constraint). We perform an extensive experimental evaluation of our approach, and show that the proposed methods produce high quality results that are comparable with the state-of-the-art MTT algorithms. In addition, our methods reduce significantly the computational cost and scale well to a large object population.

The rest of the paper is organized as follows: Section 2 defines formally the problem, whereas Section 3 surveys the related work. A detailed description and analysis of our algorithm is given in Section 4. In Section 5 we evaluate experimentally our method. Finally, Section 6 summarizes the results and presents directions for future work.

2 Problem Formulation

Let $H = \{S_1, S_2, \dots, S_M\}$ be a long, timestamped history. A *snapshot* S_i of H is a set of locations (measurements) at time t_i ; the time difference $t_{i+1} - t_i$ between consecutive timestamps is not constant. Each snapshot contains measurements from

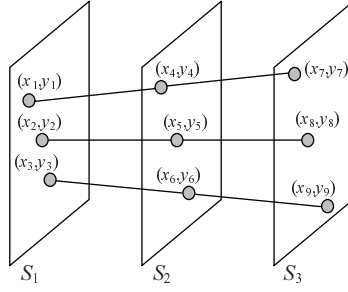


Fig. 1. Multiple target tracking (MTT) example

exactly N objects, i.e., we assume that (1) an existing object may not disappear and new objects may not appear during the interval $[t_1, t_M]$ and (2) the measurements are complete (there are no missing values). These assumptions may not hold in some cases, but our goal in this paper is to solve a restricted version of the problem. We plan to relax these constraints as part of our future work. Finally, we assume that the locations are anonymized, meaning that there is no object *id* that matches a certain location; any location measurement may correspond to any of the N objects.

Given N objects and a history H spanning M timestamps, an MTT query returns a set of N trajectories, where each trajectory i has the form $\{(x_{i_1}, y_{i_1}, t_1), (x_{i_2}, y_{i_2}, t_2), \dots, (x_{i_M}, y_{i_M}, t_M)\}$. Each triple in the above set corresponds to the location of the object at each of the M timestamps. To illustrate the significance of this result, consider the following scenario: A suspect was seen driving in the vicinity of his home address at time t_1 . What a data analyst may want to do, is issue a range query and retrieve a set of points (i.e., measurements) that may be associated with the suspect (at time t_1). After the MTT query is resolved, each of these points will be the source of a unique trajectory that will identify possible locations of the suspect at subsequent timestamps.

Figure 1 shows an example MTT query with $M = N = 3$. Each line connecting two measurements in successive timestamps indicates that the two measurements belong to the same trajectory. The three trajectories are disjoint and are formed such that the overall error is minimized (the details of the error function are discussed in Section 4). Given the illustrated associations in Figure 1, the topmost trajectory is represented as $\{(x_1, y_1, t_1), (x_4, y_4, t_2), (x_7, y_7, t_3)\}$.

3 Related Work

Multiple target tracking has been studied extensively for several decades, and a variety of algorithms have been proposed that offer different levels of complexity and tracking quality. They can be classified into three major categories: nearest neighbor (*NN*), joint probabilistic data association (*JPDA*), and multiple hypotheses tracking (*MHT*).

NN techniques [2] require a single scan of the dataset; for every set of measurements (i.e., from one timestamp), each sample is associated with a single track. The objective is to minimize the sum of all distances, where the distance is defined as a function

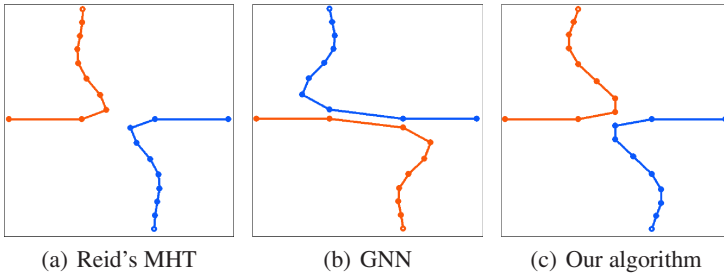


Fig. 2. Trajectory reconstruction for different methods

of the difference between the actual and predicted values. Among existing NN algorithms, the best is the global nearest neighbor (*GNN*) approach [3]. JPDA algorithms [1] also require a single scan and, for every pair of measurement-track, the probability of their association is calculated as the sum of the probabilities of all joint events. An experimental evaluation of several NN and JPDA algorithms can be found in [3]. Even though some of these methods run in polynomial time (due to their greedy nature that minimizes the error at each timestamp independently), their tracking quality is not good, leading to many false associations.

Reid's algorithm [4] is the most representative of the MHT methods. Instead of associating each measurement with a single track, multiple hypotheses are maintained, whose joint probabilities are calculated recursively when new measurements are received. Consequently, each measurement is associated with its source based on both previous and subsequent data (multiple scans). During this process unfeasible hypotheses are eliminated and similar ones are combined. Reid's algorithm produces high quality results, but its complexity grows exponentially with the number of measurements.

An example that illustrates the superiority of multiple scan techniques over their single scan counterparts is presented in Figure 2. In this example, two objects move towards each other, until they “meet”; then, they suddenly change their trajectories and move at opposite directions. GNN makes the wrong track assignments when the objects are close to each other, just because these assignments happened to minimize the error at some particular timestamp. On the other hand, Reid's algorithm tracks the two objects successfully, since it minimizes the error across all timestamps. This figure also shows the output of our method, which exhibits an accuracy that is similar to Reid's algorithm but is able to run in polynomial time (as we will illustrate in the following sections). The slight differences in the output between Reid's algorithm and ours, are due to the filters that are used to smooth the trajectories (Kalman filter for Reid, as opposed to a simpler filter for our method).

To reduce the complexity of the tracking process, [5,6] employ clustering. They group the set of measurements before forming the candidate tree, in order to remove unlikely associations. In this way, the problem is partitioned into smaller sub-problems that are solved more efficiently. Although this approach reduces the complexity, it still utilizes single scan techniques that are not accurate.

Another interesting application of multiple target tracking is investigated in [7], where the objective is to discover associations among asteroid observations that

correspond to the same asteroid. The authors introduce an efficient tree-based algorithm, which utilizes a pruning methodology that reduces significantly the search space. However, their problem settings are different from ours, since (1) they assume that there is a given motion model that has to be obeyed, and (2) they are interested in returning those sets of observations that conform to the motion model.

Finally, the idea of applying MTT techniques for the reconstruction of trajectories from anonymized data, was introduced in [8]. The authors use five real paths and show that Reid's algorithm is able to associate the majority of the measurements with the correct objects. However, their objective is not how to efficiently track multiple targets, but rather how to enhance the privacy of the users through path perturbation. In particular, they modify the original dataset in such a way that Reid's algorithm is confused.

4 Tracking Algorithm

This section discusses the details of our MTT algorithm. First, we present a brief overview of the min-cost max-flow problem. Then, we explain how to construct the graph from the history of location measurements and present a pruning mechanism that reduces significantly the graph size. Finally, we discuss the implementation details of our algorithm and analyze its computational complexity.

4.1 Preliminaries

A *flow network* [9] is a directed graph $G = (V, E)$, where V is a set of vertices, E is a set of edges, and each edge $(u, v) \in E$ has a capacity $c(u, v) \geq 0$. If $(u, v) \notin E$, it is assumed that $c(u, v) = 0$. There are two special vertices in a flow network: a source s and a destination t . A *flow* in G is a real-valued function $f : V \times V \rightarrow \mathbf{R}$, satisfying the following properties:

1. **Capacity constraint:** For all $u, v \in V$, we require $f(u, v) \leq c(u, v)$.
2. **Skew symmetry:** For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.
3. **Flow conservation:** For all $u \in V \setminus \{s, t\}$, we require $\sum_{v \in V} f(u, v) = 0$. In other words, only s can produce units of flow, and only t can consume them.

The *max-flow* problem is formulated as follows: given a flow network G , find a flow of maximum value between s and t .

The *min-cost max-flow* problem is a generalization of max-flow, where:

1. For every $u, v \in V$ the edge (u, v) has a cost $w(u, v)$, and we require $w(u, v) = -w(v, u)$.
2. The flow conservation property of the flow network is replaced by the following *balance constraint* property: For all $u \in V$, $b(u) = \sum_{v \in V} f(u, v)$. Note that, $b(u)$ may have non-zero values for vertices other than the source or the sink. In other words, every node in the network may be a producer or consumer of flow units, as long as the following flow conservation condition is satisfied: $\sum_{u \in V} b(u) = 0$.

The cost of a flow f is defined as

$$\text{cost}(f) = \sum_{(u,v) \in E} w(u, v) f(u, v)$$

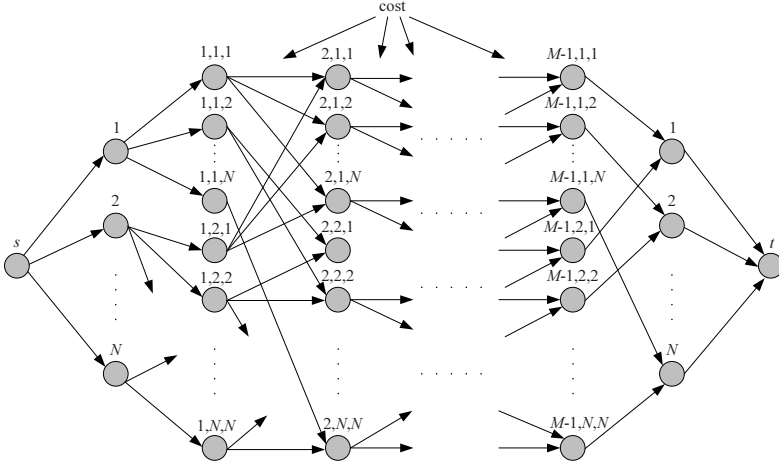


Fig. 3. Multi-target tracking (MTT) flow network

and the objective of the min-cost max-flow problem is to find the max-flow with the minimum cost.

4.2 Problem Transformation

A straightforward transformation of the MTT problem into a flow network is shown in Figure 3. Flow units are produced at the source s and consumed at the sink t ; our objective is to send a total of N flow units from s to t , each one identifying a single object trajectory. All edges have capacity 1 in the forward direction, and 0 in the reverse direction. Also, every edge (u, v) in the middle of the network (as shown in the figure) has cost value $w(u, v)$ in the forward direction, and $-w(u, v)$ in the reverse direction. The rest of the edges have zero cost.

The N vertices that are directly connected to s correspond to the first snapshot of measurements (one vertex for each location). Following these vertices are series of columns containing N^2 nodes each. Every node in these columns is identified by a triplet (t_i, p_i, p_j) , which has the following meaning: if a positive amount of flow runs through this node, then the underlying object moves from location p_i in timestamp t_i to location p_j in timestamp t_{i+1} . Consequently, edge $(t_i, p_i, p_j) \rightarrow (t_{i+1}, p_j, p_k)$ represents a partial trajectory from three consecutive timestamps ($p_i \rightarrow p_j \rightarrow p_k$), where $p_i, p_j, p_k \in [1..N]$.

The cost for the aforementioned edge is equal to the association error of the third measurement. As shown in Figure 4, if the first two measurements (p_i and p_j) belong to the same track, their values can be used to predict the next location of the object, based on the assumption that objects move on a straight line with constant speed. Therefore, for every possible location p_k , we can calculate the error of associating this measurement with any of the existing tracks. This definition of error is also used in [4]. Note that our method minimizes the sum of errors across all trajectories (similar to multiple hypotheses tracking), as opposed to methods that work in a single scan. Finally, the N

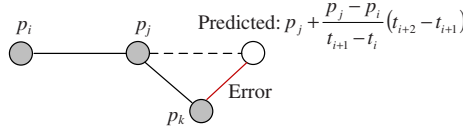


Fig. 4. Association error

nodes connected to the sink t correspond to the last set of measurements, and indicate the final positions of the moving objects.

Observe that the above flow network may lead to incorrect trajectories, by associating a single measurement with multiple tracks. For instance, if in the final solution we allow a positive amount of flow through edges $(1, 1, 1) \rightarrow (2, 1, 1)$ and $(1, 2, 1) \rightarrow (2, 1, 2)$ (Figure 3), then location 1 in timestamp 2 belongs to two different trajectories. One way to overcome this limitation is to create a bottleneck edge (with capacity 1) for each measurement that only allows a single unit of flow (i.e., track) to go through. We call this structure a *block*. Figure 5(a) illustrates the (m, k) -block, i.e., the block associated with the k_{th} measurement of the m_{th} timestamp. Let us use the notation $p_{m,k}$ to identify that particular point location. Then, this block represents all partial tracks $p_{m-1,i} \rightarrow p_{m,k} \rightarrow p_{m+1,j}$, $\forall i, j \in [1..N]$. Since the capacity of the middle edge is equal to 1, only one of these tracks can be selected.

Every (m, k) -block, where $1 < m < M$ and $1 \leq k \leq N$, is characterized by the following matrix:

$$C = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ c_{N,1} & c_{N,2} & \cdots & c_{N,N} \end{pmatrix}$$

where $c_{i,j}$ is the error in track $p_{m-1,i} \rightarrow p_{m,k} \rightarrow p_{m+1,j}$, i.e., the distance between the predicted location (based on the values of $p_{m-1,i}$ and $p_{m,k}$), and $p_{m+1,j}$. However, the block structure consists of only $(2N + 1)$ edges, which are not sufficient to represent the N^2 error values that are included in matrix C . Therefore, we modify the aforementioned block structure, and replace the middle part of the block with $2N$ vertices and N^2 edges. The result is shown in Figure 5(b). The N^2 edges connecting the two middle columns have the cost values associated with matrix C , while the remaining edges have cost equal to zero, i.e., they do not affect the process of the min-cost max-flow calculation.

The difference of the modified block structure compared to the rest of the flow network, is that we need to manually route the flows inside the block in order to guarantee that only one flow unit goes through. Specifically, when a positive amount of flow runs through a certain block, that block is automatically marked as *active* and the identifier of the edge occupying the block is recorded. An active block may only output a single flow unit, so an additional incoming flow has to be redirected backward in order to cancel the existing flow (hence the negative weight values on the reverse edges). In particular, a new flow is forced back through the reverse path of the existing flow, in order to select a new location in the previous timestamp. This is depicted in Figure 6(a), where the block is occupied by the flow with cost $c_{1,1}$. When a new flow enters from vertex $(2, 2, 1)$, it

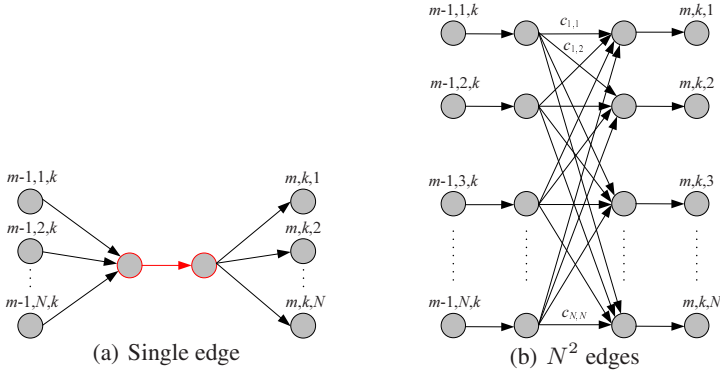


Fig. 5. Block structure for measurement $p_{m,k}$

is only allowed to follow the path indicated by the arrows, which takes the flow in the reverse direction towards vertex $(2, 1, 1)$ and cancels the original flow. Next, as shown in Figure 6(b), the incoming flow enters the block of the previous timestamp, where it also cancels the flow with cost $c_{1,1}$ and then follows the path to vertex $(2, 1, 2)$. Consequently, it selects measurement 2 at timestamp 3 (instead of measurement 1), which results in two distinct trajectories.

There are N blocks in every timestamp, each one contributing $O(N)$ vertices and $O(N^2)$ edges to the overall network. Therefore, the total number of vertices in the flow network is $|V| = O(MN^2)$, whereas the total number of edges is $|E| = O(MN^3)$.

4.3 Improving the Running Time

Solving the min-cost max-flow problem requires multiple shortest path calculations on the MTT flow network (discussed in the next section). Therefore, the size of the network is crucial for maintaining a reasonable running time. In its current form, however, the size of the flow network becomes prohibitively large when the number of measurements increases. To this end, we propose a pruning technique that may reduce significantly the size of the network. Observe that any object can travel at most $R_{\max}$ distance between two consecutive timestamps. The actual value of $R_{\max}$ depends on (i) the max-

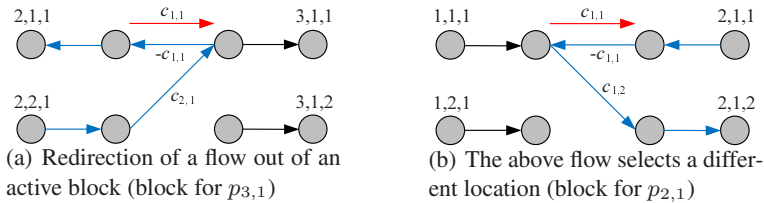


Fig. 6. Functionality of new block structure

imum speed of the objects and (ii) the time interval between the two timestamps. Consequently, every measurement $p_{m,k}$ can only be associated with those measurements $p_{m+1,i}$, $\forall i \in [1..N]$, such that the distance between the two points is less than $R_{\max}$. We can leverage this constraint in order to reduce the number of vertices and edges inside each block. Specifically, if we assume that there are, on average, K feasible associations for any measurement $p_{m,k}$, the number of vertices in the flow network is reduced to $|V| = O(MNK)$, while the total number of edges is reduced to $|E| = O(MNK^2)$. This may result in significant savings when $K \ll N$.

4.4 The MTT Algorithm

We have a single source s that needs to send N units of flow towards the destination t . Among all feasible max-flows, we are interested in finding the one with the minimum cost. A very efficient method for solving the min-cost max-flow problem is the Successive Shortest Path Algorithm [10]. It leverages the Ford-Fulkerson algorithm [9] that solves the max-flow version of the problem. The Ford-Fulkerson algorithm starts with $f(u, v) = 0$ for all $u, v \in V$, and works iteratively by finding an *augmenting path* where more flow can be sent. The augmenting paths are derived from the *residual network* G_f that is constructed during each iteration. Formally, $G_f = (V, E_f)$, where $E_f = \{u, v \in V : c_f(u, v) > 0\}$. $c_f(u, v)$ is called the residual capacity and is equal to $c(u, v) - f(u, v)$. Note that when an edge (u, v) carries a positive amount of flow in the flow network, it will be replaced by edge (v, u) in the residual network (as shown in Figure 6). This means that the residual network may contain edges with negative weights, since $w(v, u) = -w(u, v)$.

In the Successive Shortest Path Algorithm, instead of finding an augmenting path, we find the path with the minimum cost (given the weight values of the edges on the residual graph). Since the flow network may contain weights with negative values, we need to utilize the Bellman-Ford algorithm [11] for the shortest path calculations. This is not very efficient, as the Bellman-Ford algorithm has worst-case complexity $O(|V| \cdot |E|)$. In our MTT network, this translates to $O(M^2 N^2 K^3)$.

Instead, we use a well-known technique called *vertex potentials*, which transforms the network into one with non-negative costs (provided that there are no negative cost cycles). For every edge $(u, v) \in E$, where vertices u, v have potential $p(u)$ and $p(v)$, respectively, the *reduced cost* of the edge is given by: $w_p(u, v) = w(u, v) + p(u) - p(v) \geq 0$. It can be proved that the min-cost max-flow problems with edge costs $w(u, v)$ or $w_p(u, v)$ have the same optimal solutions. Therefore, by updating the node potentials, we can utilize a more efficient shortest-path algorithm during the iterations of the Ford-Fulkerson algorithm. Node potentials are initially set to zero², and are updated as follows: after the calculation of the shortest path, for every $u \in V$, $p(u) = p(u) + d(s, u)$, where $d(s, u)$ is the length of the shortest path from s to u .

The pseudo-code of our MTT algorithm is shown in Figure 7. It begins by constructing the flow network (as explained in Sections 4.2 and 4.3) from the history of measurements H . Then (lines 2-6), it initializes the flows and node potentials. At each iteration

² If prior to the first iteration of the algorithm there exist negative costs, Bellman-Ford must be invoked to remove them. In our case, however, we do not have negative costs before the first iteration, since the total flow inside the network is zero.

Algorithm MTT(H, M, N)

-
1. Construct flow network from H
 2. **for** each $(u, v) \in E$ // Initialize flows
 3. $f(u, v) = 0$
 4. $f(v, u) = 0$
 5. **for** each $u \in V$ // Initialize node potentials
 6. $p(u) = 0$
 7. **for** $i = 1$ to N
 8. Find shortest path p from s to t in G_f
 9. **for** each $u \in V$ // Update node potentials
 10. $p(u) = p(u) + d(s, u)$
 11. **for** each $(u, v) \in p$ // Augment flow across path p
 12. $f(u, v) = f(u, v) + 1$
 13. $f(v, u) = -f(u, v)$
 14. **return** N trajectories
-

Fig. 7. The MTT algorithm

of the Successive Shortest Path Algorithm (lines 8-13), a single unit of flow is added to the network; the algorithm terminates after N iterations. The resulting trajectories are returned by following each flow unit from s to t through the flow network.

Before analyzing the computational complexity of our algorithm, we should briefly discuss a common problem that may occur in min-cost max-flow calculations. Due to the negative weights of some edges in the residual network, there is a possibility that negative cost cycles exist (we actually encountered this problem in our experiments). In this case, the shortest-path calculations can not be performed and the algorithm fails. Instead of terminating the algorithm when a negative cost cycle is detected, we implement a greedy approach that may generate non-optimal solutions. In particular, we (1) output all the tracks that are discovered so far (which might not be optimal), (2) remove all the vertices and edges associated with these tracks from the flow network, and (3) start a new min-cost max flow calculation on the reduced graph.

4.5 Complexity

The computational complexity of the MTT algorithm shown in Figure 7, is directly related to the complexity of the underlying shortest-path algorithm³. Theoretically, the fastest running time is achieved with Dijkstra's algorithm [12], using a Fibonacci heap implementation for the priority queue. The complexity of Dijkstra's algorithm is $O(|V| \log |V| + |E|) = O(MNK \log(MNK) + MNK^2)$. Thus, the total running time (due to N iterations) is $O(MN^2K(\log(MNK) + K))$. This corresponds to the main contribution of our work, i.e., a multiple hypotheses tracking algorithm that works in polynomial time, instead of exponential.

We have also experimented with other implementations of shortest-path algorithms, which produced similar, and in some cases better, running times compared to the aforementioned method. For instance, the computational complexity of the Fibonacci heap

³ The complexity of graph construction is $O(MN^2 + MNK^2)$ and can be ignored.

structure has a large hidden constant; therefore, a simple binary heap is often more efficient. The overall complexity is $O(N(|V| + |E|) \log |V|) \approx O(MN^2K^2 \log(MNK))$. An interesting approach, which works surprisingly well, is to utilize Bellman-Ford’s algorithm for finding the shortest paths. Even though the complexity of Bellman-Ford is $O(M^2N^2K^3)$, in practice it runs much faster for our flow network due to the “left-to-right” structure of the graph⁴. Furthermore, Bellman-Ford’s algorithm works with negative costs as well, meaning that we do not have to maintain node potentials.

The space complexity of our method is dominated by the amount of storage required to store the $|E|$ edges of the flow network (around 20 bytes for each edge). Therefore, the worst-case space complexity of our MTT algorithm is $O(MNK^2)$.

5 Experimental Evaluation

In this section, we evaluate the performance of the proposed MTT algorithm, and compare it with a GNN implementation (using clustering) that is described in [6]. This approach works in low polynomial time with a complexity of $O(MNC^2)$ (where C is the average cluster size), and was shown to have the best performance among other MTT techniques in the detailed experimental evaluation of [3]. We do not include Reid’s MHT algorithm [4] in this comparison, since it could not produce any results within a reasonable time limit. In the following plots, we use “GNN” to label the curves corresponding to the GNN approach, and “MCMF” to label our own algorithm.

We experimented on a real road map of the city of San Francisco [13], containing 174,956 nodes and 223,001 edges⁵. The original map was scaled to fit in a $[0, 10000]^2$ workspace. The trajectories are generated as follows: (1) We randomly select a starting node and a destination node (from the map) for each object. (2) Each object then travels on the shortest-path between the two points. At the first timestamp, the distance d_i covered by each object i is randomly selected between 0 and $R_{\max}$ (as defined in Section 4.3). At subsequent timestamps, the distance is adjusted randomly by $\pm 10\% \cdot R_{\max}$, while ensuring that it neither becomes negative nor exceeds $R_{\max}$. (3) Upon reaching the endpoint, a new random destination is selected and the same process is repeated.

In each experiment we generate N random trajectories that are sampled for a period of M timestamps. We then run the corresponding MTT algorithms (without the object *ids*) and collect the resulting trajectories. These trajectories are compared to the original ones, where we measure the *success rate*, i.e., the percentage of successive triplets (as shown in Figure 4) that are associated with the correct trajectory. We use the CPU time and the success rate as the performance metrics. Table 1 summarizes the parameters under investigation, along with their ranges. Their default values are typeset in boldface. In each experiment we vary a single parameter, while setting the remaining ones to their default values. The total number of measurements varies from 50,000 to 500,000.

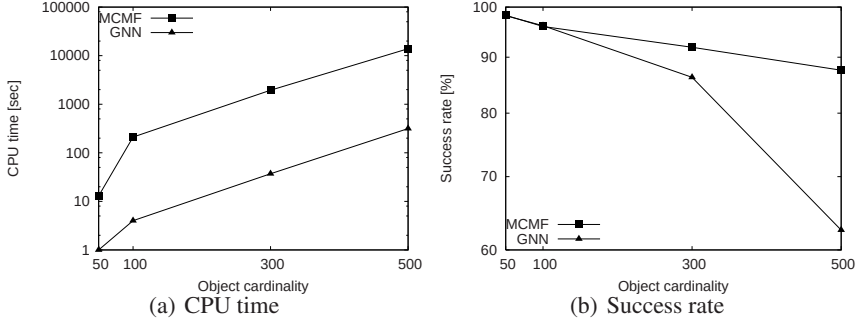
Figure 8(a) shows the running time of the two methods as a function of the object cardinality. As expected, MCMF is slower than GNN, but it improves considerably over

⁴ Actually, we also enhanced the functionality of Bellman-Ford’s algorithm with a processing queue (for vertices), which reduces the $O(|V| \cdot |E|)$ complexity.

⁵ This “network” corresponded to the map topology on which the objects move, and it has nothing to do with the “flow network” of our algorithm.

Table 1. System parameters

Parameter	Range
Number of objects (N)	50, 100 , 300, 500
Number of timestamps (M)	500, 1000 , 1500, 2000
Object speed ($R_{\max}$)	20, 40 , 80, 160

**Fig. 8.** Performance vs. object cardinality

Reid’s algorithm, which is exponential to the size of the input and fails to terminate even in the simplest of cases. We expect that by employing “divide-and-conquer” techniques (e.g., by forming clusters that are solved independently of each other, similar to the methods used in [5,6]) our algorithm will scale to much larger datasets.

The main advantage of our approach over single scan methods is depicted in Figure 8(b). This plot shows the accuracy of the trajectory reconstruction process, in terms of the percentage of correct associations. Even though GNN achieves lower running time, its accuracy deteriorates rapidly with increasing number of objects. This is due to the fact that more objects exhibit crossing trajectories, which confuses GNN (as shown in Figure 2). Therefore, the results of GNN may be of little value in practice. On the other hand, MCMF is very accurate and maintains a success rate of over 87%.

Figure 9 shows the CPU time for GNN and MCMF, as a function of the history length M . GNN scales linearly with M , while the slope of the curve for MCMF exhibits some variations. This behavior can be explained by the approximation that is discussed in the last paragraph of Section 4.4. When negative cost cycles are detected, the size of the graph is reduced and subsequent iterations are executed faster. Consequently, the running time of our algorithm is also affected by the appearance of negative cost cycles. Note that the complexity analysis in Section 4.5 corresponds to the worst-case, i.e., when negative cost cycles never form. Regarding accuracy, both algorithms are unaffected by the number of timestamps.

Next, we investigate the effect of the object speed on the CPU time. As shown in Figure 10(a), both algorithms become slower as the speed increases. For GNN, this is due to the fact that clustering is less effective when the objects move faster. MCMF is also affected by the object speed, since the average number of feasible associations K for each measurement increases. Finally, Figure 10(b) depicts accuracy as a function

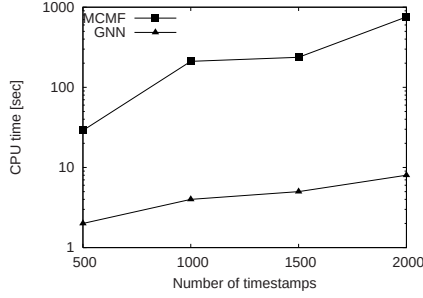


Fig. 9. CPU time vs. number of timestamps

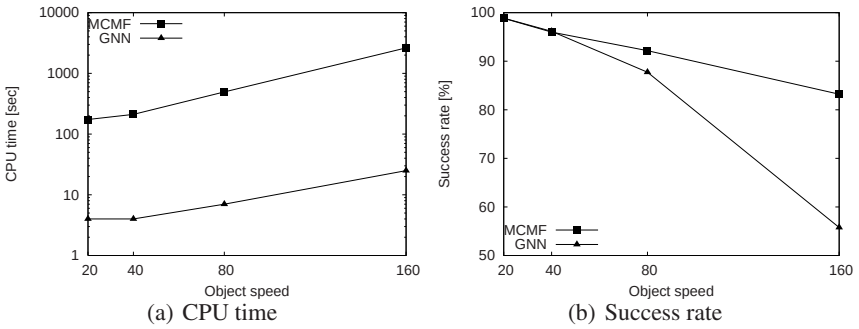


Fig. 10. Performance vs. object speed

of the speed of the moving objects. As the speed of an object increases, the successive locations of its trajectory move further apart from each other. Therefore, within a snapshot there may be many measurements that are closer to the object's previous location than the correct one. The greedy nature of GNN is not able to deal with that and, for high speeds, only 55% of the associations are correct. MCMF, on the other hand, is clearly superior; its success rate is always over 83%.

6 Conclusions

In this paper, we investigate the feasibility of applying multiple target tracking techniques in the context of anonymized trajectory databases. Existing methods are either very slow (i.e., the complexity is exponential to the number of measurements), or very inaccurate. The main contribution of our work lies in the novel transformation of the MTT problem into an instance of the min-cost max-flow problem. This transformation allows for a polynomial time solution in $O(MN^2K(\log(MNK) + K))$, where M is the number of timestamps, N is the number of measurements in each timestamp, and K is the average number of feasible associations for each measurement. Our initial results indicate that the proposed method produces very accurate results.

In the future, we plan to extend our work in a number of directions. First, we will investigate the feasibility of our method in complex scenarios where (1) new tracks may be initiated at random timestamps, and (2) location measurements may be lost due to errors on the wireless channel. Second, we will combine our methods with clustering, in order to further reduce the computational and space complexity. Specifically, through clustering, we will partition the tracking problem into a number of smaller sub-problems that can be solved more efficiently.

References

1. Bar-Shalom, Y., Fortmann, T.E.: Tracking and Data Association. Academic Press, London (1988)
2. Blackman, S.S.: Multiple-Target Tracking with Radar Applications. Artech House (1986)
3. Leung, H., Hu, Z., Blanchette, M.: Evaluation of multiple radar target trackers in stressful environments. *IEEE Trans. on Aerospace and Electronic Systems* 35(2), 663–674 (1999)
4. Reid, D.B.: An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control* 24(6), 843–854 (1979)
5. Chummun, M., Kirubarajan, T., Pattipati, K., Bar-Shalom, Y.: Fast data association using multidimensional assignment with clustering. *IEEE Trans. on Aerospace and Electronic Systems* 37(3), 898–913 (2001)
6. Konstantinova, P., Nikolov, M., Semerdjiev, T.: A study of clustering applied to multiple target tracking algorithm. In: *Proc. International Conference on Computer Systems and Technologies (Comp.Sys.Tech.)*, pp. 1–6 (2004)
7. Kubica, J., Moore, A.W., Connolly, A., Jedicke, R.: A multiple tree algorithm for the efficient association of asteroid observations. In: *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 138–146 (2005)
8. Hoh, B., Gruteser, M.: Protecting location privacy through path confusion. In: *IEEE International Conference on Security and Privacy in Communication Networks (Secure Comm)*, pp. 194–205 (2005)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. The MIT Press, Cambridge (2001)
10. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs (1993)
11. Bellman, R.: On a routing problem. *Quarterly of Applied Mathematics* 16(1), 87–90 (1958)
12. Dijkstra, E.: A note on two problems in connection with graphs. *Numerische Mathematik* 1, 269–271 (1959)
13. Brinkhoff, T.: A framework for generating network-based moving objects. *GeoInformatica* 6(2), 153–180 (2002)

REALM: Replication of Data for a Logical Group Based MANET Database

Anita Vallur, Le Gruenwald*, and Nick Hunter

The University of Oklahoma
School of Computer Science
Norman, Oklahoma 73019 USA
ggruenwald@ou.edu

Abstract. Mobile Ad-Hoc Networks, or MANETs, provide communication between free-roaming mobile hosts without a fixed infrastructure. These MANETs operate under conditions of limited battery power and may also experience frequent network partitioning. This paper introduces a data replication technique called REALM (REplication of data for A Logical group based MANET database). REALM takes advantage of application semantics when determining where to place replicas of data items. It also tries to predict network partitioning and disconnection ahead of time and creates copies of data items accordingly. Experiments show that REALM increases the percentage of successful transactions with or without the presence of network disconnection and partitioning.

Keywords: Replication, data replication, algorithm, mobile databases, mobile ad hoc network databases.

1 Introduction

A mobile ad hoc network is a collection of wireless enabled mobile devices called mobile hosts. Every mobile host performs the functions of a router to enable communication between those mobile hosts that are not directly within each other's communication range. The unpredictable movement of mobile hosts and the absence of a fixed network infrastructure make MANET suitable for applications such as battlefield, rescue operations and conferences.

Since power restrictions and network partitioning in MANETs cause data to become unavailable, multiple copies of the same data are often created. This process of creating multiple copies of the same data item is called data replication. Data replication is more complex in MANETs than in conventional distributed database systems. This is because replication for MANETs should address the issues of power restrictions, mobility of clients and servers, network disconnection and partitioning, as well as real-time requirements since many MANET applications are time-critical [12]. In addition to these concerns, in practice, a MANET data replication scheme also needs to deal with the following limitations:

* Contact author.

1. The data replication scheme should be able to work in the absence of a global node positioning/location management system. This is because the use of such devices/systems increases the power consumption of the mobile hosts in the network. In addition, it is unrealistic to impose the use of these systems on applications which may otherwise have no use for them.
2. The absence of motion parameters of mobile hosts like speed and direction should not cause the data replication scheme to fail. It is not realistic to assume that information about the movement of users is always known ahead of time, especially in MANET applications like rescue operations.

MANET replication is a widely researched area. However, to the best of our knowledge, all existing MANET data replication techniques assume the availability of a GPS and/or that of the motion parameters of mobile hosts. This paper attempts to propose a MANET data replication technique, called REALM (Replication of data for A Logical group based MANET database) that does not make such assumptions.

Mobile hosts in typical MANET applications belong in groups based on the functions they perform in the applications. Since all mobile hosts in a group perform the same set of operations, they access the same set of data items. Members of every group are spread across the network, and perform their functions in their locations. Due to this many tend to move within a limited distance of one or two hops. They collaborate with members of other groups in their locations to achieve the goals of the MANET application. REALM capitalizes on this common MANET application semantic by grouping mobile hosts into logical groups. Each logical group is comprised of mobile hosts that will need to access the same data items.

REALM aims at increasing the percentage of successful transactions while reducing the energy consumption of mobile hosts, as well as balancing the energy consumption of servers. By considering such groups of mobile hosts, REALM will be able to replicate data items closer to clients that will need to access the data items more often. This will reduce the execution time of transactions, and hence, will result in an increased percentage of successful transactions.

The rest of this paper is organized as follows: Section 2 presents some representative MANET applications and identifies their semantics; Section 3 reviews some existing works in MANET data replication; Section 4 describes the proposed replication technique (REALM); Section 5 introduces the prototype used to evaluate the performance of REALM; Section 6 presents the results of the experiments performed; and finally, Section 6 provides conclusions and future research directions.

2 MANET Applications

Since MANETs do not require a fixed infrastructure, they find use in a range of applications such as disaster rescue, personal area network [14] and one laptop per child project [9]. In this section, we briefly describe some of these applications and identify the semantics that should be incorporated into the design of a data replication technique.

1. *Disaster Rescue Application*: A disaster rescue application involves functions like identifying injured people, getting medical aid to the injured, and identifying the dead. Since each of these functions is performed by a group of personnel, they will access the same set of data items. Members of different groups collaborate with each other to achieve the overall goals of the application. For example, a member of the group that rescues people from the disaster will need to collaborate with members of the first aid team. Members of the same group are spread out, and they provide their services in the rescue site at their allotted location. They may also move around to provide their services at other locations.
2. *One Laptop per Child Project Application*: This non-profit project aims to sell cheap and durable laptops to governments such as Africa. The laptops are a special prototype designed by the project and serve as learning tools for their recipients. One particular aspect of the design is that the computers work in an ad-hoc network mode and can keep an Internet connection concurrently. The computers are created with a long battery life, but are limited by their low processing capability and storage space. In a school environment, groups of students require data for their different classes. Students will also move from class to class where different data may be needed.
3. *Personal Area Network (PAN)*: PAN is a very popular application of MANET. It allows users to set up their own network to be able to access their information on different systems like laptops, desktops, pocket PCs and cell phones. In a household or small office environment, different groups of people can use the PAN to access information pertaining to their interest. For example, in an office environment, the accounts team may need to access the revenue and expenditure of the business, while the investment team may need access to sales statistics and fund details. Since the network is set up for a small office, the users move around the boundaries of the office.

The example MANET applications described above have the following common characteristics:

1. Users belonging to the same group need to perform the same set of tasks. Hence they access the same data items. For the purposes of data replication, such users who need access to the same data item can be considered to be in the same logical group. Knowledge of the existence of logical groups will allow every server in the network to identify data items that will be highly accessed by clients that may submit their transactions to it.
2. Users move randomly; however, their movement is mostly within a finite distance. On average, the range of an 802.11b wireless card is about 100 meters. Hence, from the above description, it is realistic to say that the movement of the users is within one or two hop communication distance of an 802.11b wireless card.
3. Members of one group tend to communicate and collaborate with members of other groups in order to achieve the objectives of the application.

The above identified common characteristics of MANET applications can be used to improve data replication, as we will show in Section 4 where our proposed algorithm, REALM, is presented.

3 Literature Review

While MANET replication is a widely researched area, there exists a limited number of data replication techniques designed for MANET database systems. [4] provides three models for replicating data in a distributed database. Static Access Frequency allocates data on each server until storage is exhausted. Dynamic Access Frequency and Neighborhood, introduces additional functionality to remove duplication between neighboring hosts. The Dynamic Connectivity Based Grouping method improves replication by removing redundancy between groups of mobile hosts. [5] improves on these techniques by considering the occurrence of data update.

[6] uses the link strength between hosts when deciding which data item replicas to eliminate. However, [6] relies on the availability of the movement information of mobile hosts to calculate the link strength between them. These works also assume that the access frequencies of data items are known ahead of time and do not change. [7] improvises on these techniques by considering correlation between data items in the calculation of data access frequencies.

[12] weighs data access frequencies based on the remaining energy level of the host and the link strength of connections to other hosts. It considers server workload and energy consumption when accessing replicas. It relies on the availability of GPS as well as the movement information of the mobile hosts to calculate link strength as well as to predict network partitioning.

As discussed in Section 1, the availability of a GPS or movement information of mobile hosts is not always realistic in MANET applications. In addition, data access frequency calculated by a server may change as move near and away from the server. Hence, it is not realistic to assume that access frequencies are known ahead of time and are constant. Apart from these limitations, no existing technique has explored the effect of application semantics on data access in MANET applications. From our study of MANET applications, we believe that knowledge of application semantics can help servers store data items closer to clients that access them more frequently. The next section describes REALM (REplication of data for A Logical group based MANET database). REALM tries to incorporate application semantics in data replication to overcome the shortcomings of existing MANET replication works.

4 Proposed Replication Technique (REALM)

The main goal of REALM is to increase the percentage of successful transactions. It also attempts to reduce the power consumed by the mobile hosts, and balance the power consumed by servers in the network.

Unlike existing MANET works, REALM takes advantage of the characteristics of typical MANET applications (presented in Section 2). REALM groups mobile hosts based on the data items they will need to access. Mobile hosts that access the same set of data items belong in the same logical group. Group membership of mobile hosts helps in identifying the data items that any mobile host in the network will need to access, as well as to identify the most frequently accessed data item on every server.

REALM does not require the use of GPS or the knowledge of movement information of the mobile hosts in the network.

REALM consists of four algorithmic components, namely *MANET Partition Prediction Algorithm*, *Replica Allocation Algorithm*, *Replica Access Algorithm* and *Replica Synchronization Algorithm*. Before describing these components, Sections 4.1 and 4.2 describe the transaction management architecture and data and transaction types considered by REALM, respectively.

4.1 Transaction Management Architecture

Clients in a MANET application submit their transaction to any server in the network. The coordinator is the server that receives the transaction from the client. The coordinator selects servers to access the data items required to execute the transaction. After the execution of the transaction, the coordinator collects the results and forwards them to the client. As described in Section 2, group members in MANET applications are spread out in the area of the application. Due to this, selecting participant servers based on group membership will incur increased transaction execution times. Hence, REALM relies on a decentralized MANET architecture in which data is accessed from servers regardless of their group membership.

4.2 Data and Transaction Types

REALM considers both read only and read write data. Apart from real-time transaction types, firm and soft, REALM also considers the following transaction types:

1. *Accurate Value Transactions*: These transactions need the most recent value of the data to succeed. A transaction initiated to increase the death toll by 1 is an example of an Accurate Value transaction.
2. *Approximate Value Transactions*: These transactions can tolerate outdated values of data items. An example of this type of transaction is one that tries to access weather information.

4.3 MANET Partition Prediction Algorithm

The MANET partition prediction algorithm is based on the Optimized Link State Routing Protocol (OLSR) [11]. It is used by all servers in the network to predict the occurrence of network disconnection/partitioning.

The MANET partition prediction algorithm capitalizes on the hello packets that are used by OLSR to implement neighbor discovery. OLSR transmits one hello packet every x seconds. In addition, OLSR calculates the strength of every link once every y seconds (x and y are configurable). The value of y should be chosen after considering the tradeoff between keeping the servers updated with changes in link quality and the energy consumption overhead that it may pose. This ensures that the routing neighbor tables are kept updated with minimal overhead of messages. Using the number of hello packets that were expected and the number of hello packets that were actually received, mobile hosts S_1 and S_2 at either side of a link calculate the link quality (LQ) using the following formula [11]:

$$LQ = \frac{\text{No. of hello packets received in the last } y \text{ seconds}}{\text{No. of hello packets expected in the last } y \text{ seconds}} \quad (1)$$

Then, S_1 and S_2 exchange their calculated values of LQ through a ‘Topology Control’ message. At either host, the value of LQ received from the other end is called Neighbor Link Quality (NLQ). Using LQ and NLQ, S_1 and S_2 calculate the bidirectional link quality as follows [11]:

$$\text{Bidirectional link quality of the link connecting } S_1 \text{ and } S_2 = LQ * NLQ \quad (2)$$

When the bidirectional link quality decreases beyond a preset threshold accompanied by an increase in the transmission time, S_1 and S_2 predict their disconnection. Then they identify other the mobile hosts that will become unreachable due to their disconnection. When the total number of servers that will be disconnected from S_1 (S_2) increases beyond a preset threshold, S_1 (S_2) triggers replication.

4.4 Replica Allocation Algorithm

The replica allocation algorithm is executed by every server in the network to determine the data items that must be stored on it. It assumes that the group definitions and group memberships of every mobile host are known by all servers. In addition, the location of the original copy of every data item is stored in the replica table of every server. Every server S triggers the execution of this algorithm when any one or more of the following conditions is true:

1. The percentage of successfully executed transactions at S falls below a preset percentage. This ensures improved data availability in the network.
2. The occurrence of network partitioning is predicted by S .
3. The percentage of transactions executed on S that access data on other servers is greater than those that access data on S . This is done to minimize the number of remote transactions since such transactions consume higher server energy.
4. The energy level of S falls below the average energy level of the servers that are one or two hops away from S . This is done for the server S to reduce the energy consumption of S .

To illustrate the working of the replica allocation algorithm, consider the example MANET topology in Fig. 1 where C_i and S_j denote a client and a server, respectively. The ovals around C_i and S_j denote the communication range of C_i and S_j . This allows the figure to represent the number of communication hops between C_i and S_j . Mobile hosts with overlapping communication ranges can communicate directly. Mobile hosts also act as routers and relay communication between unconnected mobile hosts. For example, S_1 and C_1 communicate directly, while C_1 and S_2 communicate through S_1 . Let S_1 , S_2 and S_3 store the original copies of data items D_1 , D_2 and D_3 . Let the original copies of D_4 , D_5 and D_6 be disconnected from the network. This might be due to loss of energy on these servers or their movement into a different partition. Tables 1 and 2 represent the group membership of the mobile hosts in the network and the access log maintained at server S_3 respectively. At the time when replication is triggered by S_3 , let C_6 be disconnected from S_3 .

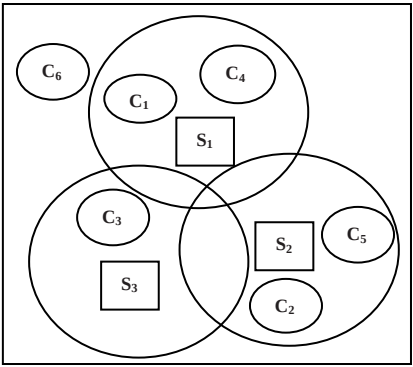


Fig. 1. Example MANET Topology

Table 1. Group Membership of Mobile Hosts

Group Name	Group Members	Data Items of Interest to Group Members
G ₁	C ₁ , C ₄	D ₁ , D ₃
G ₂	C ₃ , C ₅	D ₂ , D ₅
G ₃	C ₂ , C ₆ , S ₂	D ₂ , D ₄ , D ₆

Table 2. Access Log at Server S₃

Data Items	Mobile Hosts	Number of Accesses of the Data Item
D ₄	C ₃	10
D ₅	C ₃	12
	S ₂	20
D ₂	C ₃	25
	C ₆	8

When replication is triggered on S₃, the following steps are performed:

1. Calculation of access frequencies: The access frequency of a data item D on server S (AF^D_S) is calculated as the number of transactions which were submitted to S by clients that are currently within two hops from it.
At S₃, the access frequencies of the various data items are calculated as follows:
 $AF^{D_4}_{S_3}$ = Number of accesses made by C₃ = 10
 $AF^{D_5}_{S_3}$ = Number of accesses made by C₃ + Number of accesses made by S₂:
 $\phantom{AF^{D_5}_{S_3}} = 12 + 20 = 32$
 $AF^{D_2}_{S_3}$ = Number of accesses made by C₃ = 25
The number of accesses for D₆ made by C₆ is not considered in the calculation of the access frequency of D₂ since C₆ is not a one or two hop neighbor of S₃.
2. Arrange the data items in the descending order of their calculated access frequency in a list L . At S₃, L consists of data items D₅, D₂ and D₄, after arranging the data items in the descending order of their access frequencies.

3. For each mobile host within two hops from S , determine the data items of interest to the mobile host based on its group membership. If any of these data items are not already present in L , add it to L . At S_3 , D_6 is added to L since it is one of the data items of interest to group G_3 , to which S_2 belongs. Hence, L at S_3 consists of D_5 , D_2 and D_4 and D_6 .
4. For each data item D in L , determine whether the original copy server of D or a replica of D is connected to S . If so, eliminate D from L . In the example, since S_2 stores the original copy of D_2 , S_3 eliminates D_2 from L . Hence, L at S_3 consists of D_5 , D_4 and D_6 .
5. For every remaining data item D in L , if the mobile host that has accessed D maximum number of times is a server, request that server to store a replica of D . If the request is accepted, eliminate D from L . At S_3 , S_2 is the mobile host which accesses D_5 . Hence S_3 requests S_2 to store a copy of D_5 . Once S_2 accepts this request, L at S_3 then consists of D_4 and D_6 .
6. Store the remaining data items in L on S until S runs out of free space. S_3 stores D_4 and D_6 .

4.5 Replica Access Algorithm

The aim of the replica access algorithm is to help increase the percentage of successful transactions while also conserving the power consumption of mobile hosts. The data access is based on the data and transaction type.

Since firm transactions have only one deadline, they are executed at the nearest server that stores the required data. Soft transactions, on the other hand, have a higher probability of execution compared to firm transactions. Hence, they are executed at the nearest server with highest energy that stores the required data.

Accurate value transactions are executed at the server that has the most recent copy of the data items. This is determined by the coordinator of the transaction by requesting the most recent timestamp of the required data item from servers that store copies of it. Approximate value transactions are executed at the most updated server within two communication hops of the coordinator.

Read only data items can be accessed from any server on which they are stored. Hence, they are executed based on the transaction type. Read write data items are accessed from any server they are stored in, if the energy level of the server is greater than or equal to the average energy level of the servers in the network. This is possible because the replica synchronization algorithm of REALM ensures that servers with energy level at least equal to the average server energy level in the network are updated at all times.

4.6 Replica Synchronization Algorithm

The replica synchronization algorithm ensures that replicas of every data item in the network are in sync. It does this by propagating updates to servers that store the updated data item. However, since such an approach may lead to high server power consumption, the update operation and the most recent update timestamp of the data item is propagated only to servers with energy level at least equal to the average power level of the servers in the network.

Once a server S receives a propagated update for a data item D it stores, it compares the received timestamp with the last updated timestamp of its copy of D . If the timestamps match, the two copies of D are in sync with each other. So S executes the received update operation on D . If the timestamps differ, S requests a copy of the most recent value of D .

5 Prototype Model

REALM has been implemented on an existing MANET prototype ([3]). This prototype consists of servers (laptops) on Fedora Core Linux operating system running the MySQL database [10], and clients (PDAs) running the Familiar Linux operating system. Both servers and clients in the network run the Optimized Link State Routing Protocol (OLSR) with its link quality extension enabled. Clients in the network generate real-time firm and soft transactions.

Apart from REALM, two other replication models, namely, the No Replication model and the Hara model ([5], [6]) have been implemented on the prototype for purposes of performance comparison. The No Replication model allows us to compare the performance of REALM with that of a system that does not store multiple copies of data items. REALM and the Hara model are similar in that both assume a decentralized MANET architecture and consider update transactions. REALM replicates data items differently from the Hara model by allowing application semantics to influence data replication. In addition, REALM also tries to predict the occurrence of network partitioning and replicates data items as necessary.

We created a sample database based on the requirements of a fire rescue scenario. The experimental prototype consisted of five servers and eight clients. Four logical groups of mobile hosts were defined. At the start of every experiment, only the original copy of every data item was created. Each client initiated a total of 1000 transactions for every run of the experiment. A comparison between the performances of REALM and the Hara model, as presented in Section 5, allows us to gauge the effectiveness and overhead incurred by REALM.

6 Experimental Results

Experiments were performed using the prototype described in Section 5, varying the dynamic parameters namely, Percentage of Firm transactions, Percentage of Accurate transactions, Access Frequency, Frequency of Network Partitioning, Transaction Inter Arrival Time and Read/Write Ratio. Due to space constraints, we present only some of the graphs obtained from the experiments performed.

6.1 Effects of Firm Transactions

Fig. 2 shows the effect of the percentage of firm transactions on the percentage of successful transactions and the server energy consumption. As the percentage of firm transactions increases, the percentage of successful transactions decreases. REALM executes a higher percentage of transactions successfully as it continually monitors

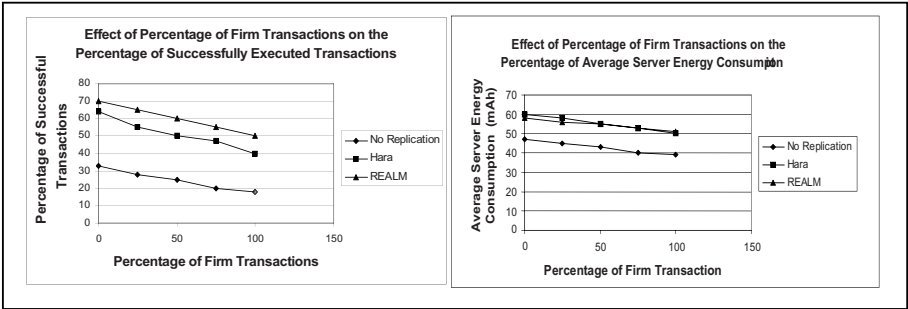


Fig. 2. Effect of Percentage of Firm Transactions

the percentage of successful transactions and replicates data items to improve this metric. As a consequence of this, however, REALM incurs high server energy consumption. The Hara model also incurs comparable server energy consumption as it broadcasts data items in order to synchronize its replicas.

6.2 Effects of Accurate Value Transactions

Fig. 3 shows the effect of percentage of accurate value transactions on the percentage of transactions executed and the average difference in energy consumption of two servers.

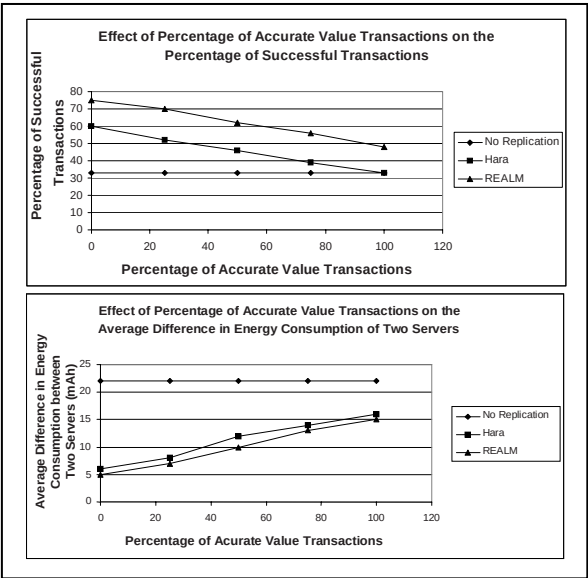


Fig. 3. Effect of Percentage of Accurate Value Transactions

The percentage of transactions successfully executed by the No Replication model is independent of the percentage of accurate value transactions since it does not maintain copies of data items. REALM propagates updates as soon as they are processed; increasing the number of updates replicas. Hence, it executes the highest percentage of successful transactions among the models being studied. With increase in the number of accurate value transactions, the workload of those servers with the most up-dated copy of data items increases. This causes the increase in imbalance of server energy consumption of both the Hara model and REALM. REALM incurs the least imbalance among the three models. This is because it replicates and accesses data items considering the energy levels of servers.

6.3 Effect of Data Access Frequency

The effect of data access frequencies on the percentage of successful is shown in Fig. 4. Data access frequency is represented using the zipf parameter θ . As θ increases, the number of data items being accessed frequently decreases. Due to this, Hara model as well as REALM increases the number of replicas in the network. Hence, the percentage of successful transactions yielded by both models increases with increase of θ . Unlike the Hara model, REALM considers the requirements of the clients in addition to the access frequencies. Due to this, it is able to cater to the needs of the clients better than the Hara model. This allows REALM to yield higher percentage of successful transactions even as the value of θ increases.

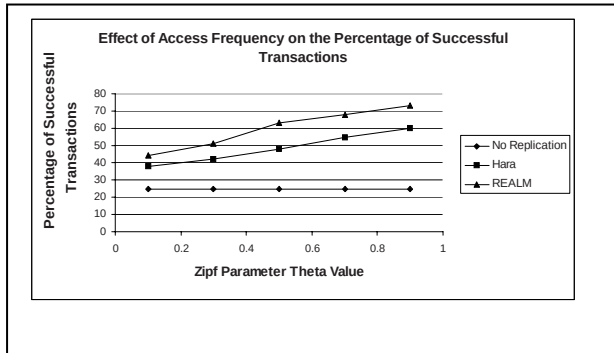


Fig. 4. Effect of Data Access Frequency

6.4 Effects of Network Partitioning Frequency

Fig. 5 shows the effect of frequency of network partitioning on the percentage of successful transactions and the average server energy consumption. With increase in the frequency of network partitioning, servers move into different partitions, increasing the unavailability of data. REALM yields the highest percentage of successful transactions among the models studied. This is because REALM tries to predict network partitioning and triggers replication ahead of its occurrence. However, this also

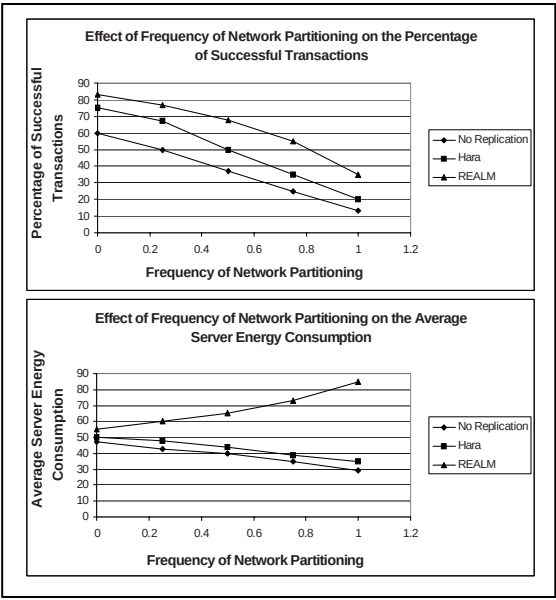


Fig. 5. Effect of Frequency of Network Partitioning

causes its server energy consumption to rise. This is because, with every partition prediction, REALM replicates data items. As the frequency of partitioning increases, the number of times that replication is triggered increases.

6.5 Effects of Transaction Inter-arrival Time

Fig. 6 shows the effect of transaction inter arrival time (IAT) on the percentage of successful transactions. The percentage of successful transactions increases with increase in IAT. This is due to decrease in the time rate of server workload.

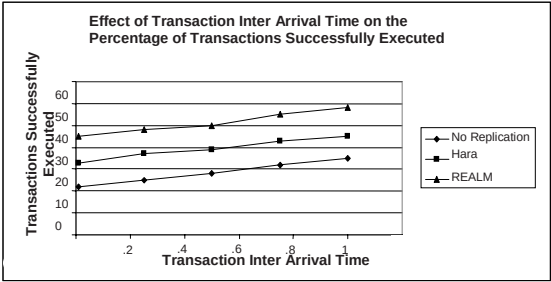


Fig. 6. Effect of Transaction Inter Arrival Time

The No Replication model queues up transactions that access the same data item on a single server because each data item is only available in a single location. Unlike the Hara model, REALM reduces the total execution time of every transaction considering communication distance as a factor for choosing participants. Hence, it yields the highest percentage of successful transactions among the models studied.

7 Conclusions and Future Work

This paper presented a data replication technique called REALM for logically group based MANET real-time databases. By considering groups of mobile hosts which access the same set of data items, REALM is able to take advantage of the needs of the clients in the network at the time of replication. REALM yields the best performance in terms of percentage of successful transactions and balance in server energy consumption when compared to the existing techniques.

REALM can be further strengthened by developing a rigorous, power-efficient partition prediction scheme. Clients can maintain replica tables to be able to make a more informed choice of the transaction coordinator. The conditions for triggering replication can be checked after a preset number of transactions have been executed. Guidelines to choose this value can be established. In addition, REALM can also be extended to tolerate overlapping and variable group definitions. This will involve a scheme to update the group membership of mobile hosts.

References

1. Bellavista, P., Corradi, A., Magistretti, E.: REDMAN: A Decentralized Middleware Solution for Cooperative Replication in Dense MANETs. In: International Conference on Pervasive Computing and Communications Workshops, pp. 158–162 (2005)
2. Chen, K., Nahrstedt, K.: An integrated data lookup and replication scheme in mobile ad hoc networks. In: SPIE International Symposium on the Convergence of Information Technologies and Communications, pp. 1–8 (2001)
3. Gruenwald, L., Bernedo, P., Padbmanabhan, P.: PETRANET: A Power Transaction Management Technique for Real Time Mobile Ad-Hoc Network Databases. In: IEEE International Conference on Data Engineering (ICDE), p. 172 (2006)
4. Hara, T.: Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In: IEEE INFOCOM, pp. 1568–1576 (2001)
5. Hara, T.: Replica Allocation Methods in Ad Hoc Networks with Data Update. ACM-Kluwer Journal on Mobile Networks and Applications, 343–354 (2003)
6. Hara, T., Loh, Y.H., Nishio, S.: Data Replication Methods Based on the Stability of Radio Links in Ad Hoc Networks. In: The 14th International Workshop on Database and Expert Systems Applications, pp. 969–973 (2003)
7. Hara, T., Murakami, N., Nishio, S.: Replica Allocation for Correlated Data Items in Ad Hoc Sensor Networks. ACM SIGMOD RECORD, 38–43 (2004)
8. Hauspie, M., Simplot, D., Carle, J.: Replication decision algorithm based on link evaluation services in MANET. CNRS UPRESA 8022 – LIFL Univ. Lille (2002)
9. One Laptop Per Child (last accessed, April 2007), <http://www.laptop.org>
10. MySQL (last accessed, February 2007), <http://www.mysql.com>

11. OLSR (last accessed, February 2007), <http://olsr.org>
12. Padmanabhan, P., Gruenwald, L.: DREAM: Data Replication in Ad Hoc Mobile Network Databases. In: IEEE International Conference on Data Engineering (April 2006)
13. Ratner, D., Reiher, P., Popek, G.J.: Roam: A Scalable Replication System for Mobility. *Mobile Networks and Applications* 9(5), 537–544 (2004)
14. Sun, J.: Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing. In: Proc. of ICII, Beijing, China, pp. 316–321 (2001)

A Cache Management Method for the Mobile Music Delivery System: JAMS

Hiroaki Shibata, Satoshi Tomisawa, Hiroki Endo, and Yuka Kato

School of Industrial Technology, Advanced Institute of Industrial Technology,
1-10-40 Higashi-Ohi, Shinagawa-Ku Tokyo 1400011, Japan

Abstract. The authors have proposed a music delivery system JAMS (JAMais vu System), which uses ad-hoc networks with mobile devices for the localization services. In this paper, they focus on a cache management scheme for JAMS to deliver music files efficiently. The proposed scheme makes it possible to manage cache files robustly by using adaptive cache delivery according to each node request. In addition, the authors conduct a simulation experiment and obtain access failure ratio of music files under various system conditions. The simulation results indicate that the proposed scheme is suitable for the system using localization services.

Keywords: Ad-Hoc Networks, Music Delivery Services, P2P, Cache Management.

1 Introduction

With the recent increased availability of high performance mobile data terminals and broadband Internet access via mobile networks, mobile network services are widely noticed. From these viewpoints, the authors have proposed music delivery system JAMS (JAMais vu¹ System)[1][2], which uses ad-hoc networks with mobile devices for localization services. The design concept is to give users a surprise and a pleasure in their daily lives. Recently, context awareness technologies, which provide services according to user conditions and user features, are widely noticed[3][4]. The standpoint of JAMS is opposite to them, and it actively uses accidental transitions of situations.

JAMS users have stored music files in their own mobile terminals in advance, and share the music files among other users staying in the same space, such as a commuter train, a cafeteria, a classroom and so on. At that time, since the space in which a user is staying changes every moment, JAMS users staying in the same space also change and sharing music files change. As a result, service contents vary according to users' staying spaces, and JAMS makes it possible to provide unexpected and subconscious services.

For such mobile music delivery services in ad-hoc networks, effective music delivery schemes, which use cache data, are needed in order to avoid load

¹ The feeling that you are a stranger to something which is actually familiar to you. This is the antonym of “*deja vu*,” which is the feeling that you have previously experienced something which is happening to you now.

processing concentration. Several studies have examined cache delivery methods for sharing files. As for an effective cache management in P2P networks, a cache management method of Winny is proposed[5]. Since this method needs some stable nodes with enough resources and does not consider node mobility well, it is difficult to apply it to JAMS. Effective cache delivery methods in ad-hoc networks have been also proposed[6][7]. However, these methods deliver cache data to networks beforehand according to prediction of frequency in use of a file, and it is difficult to use them for JAMS (because of localization of contents).

In order to design a cache management method for JAMS, the following two problems must be solved.

- The problem on localization of contents:
Because the targets of JAMS are local ad-hoc networks, existing files in a “Space” change every moment. For this reason, it is necessary to design a robust cache delivery method which is not affected by inequality of contents.
- The problem on mobile terminals:
Mobile terminals have many restrictions of resources, such as CPU power, memory size, disk size and power consumption. Therefore, it is necessary to deliver caches effectively with a simple method as much as possible. Load balance is also needed.

To solve these problems, this paper proposes an adaptive cache delivery method according to requests from nodes. This method does not optimize cache delivery process. The proposed method makes it possible to manage cache data robustly for JAMS. In addition, by conducting simulation experiments, this paper confirms the effectiveness of the proposed method.

2 Overview of JAMS

This section describes the overview of JAMS. JAMS provides services by using localization of contents, so that the authors explain the differences of contents in different “Spaces”. Then, system architecture and system functions of JAMS are shown.

2.1 Localization of Contents

In this paper, localization of contents is expressed as the word of “Space”. An image of “Space” is shown in Fig. 1. In this figure, nodes within a specific range construct a network (localization), and the network is defined as a “Space”. As for JAMS, the localization is made by using ad-hoc networks. In a space within a specific range (e.g. in the train, in the classroom etc.), an ad-hoc network is constructed by mobile terminals which users in this space have, and music files are shared among users in the network.

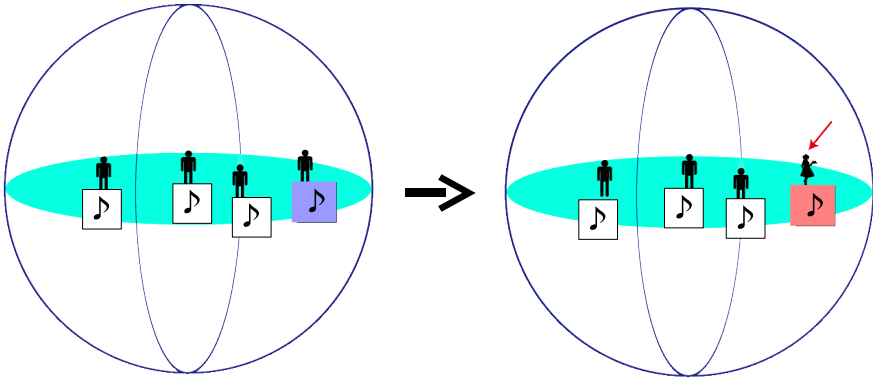


Fig. 1. An image of “Space”

2.2 System Architecture

System architecture of JAMS is shown in Fig. 2. JAMS is a pure P2P type network service. The same application software is implemented on all of terminal nodes, and those nodes communicate with each other on an equal position. Therefore, both of sending and receiving functions of music data (the streaming function) are implemented on one terminal. In addition, JAMS has the function which calculates the node feature value by using music file characters of the node and indicates the value to the user, and the function which calculates the “Space” feature value by using feature values of nodes in the “Space”. Each node also has the function for those (the feature value calculation function).

Moreover, the cache management method proposed in this paper has the mechanism which deliver caches of music files to nodes in ad-hoc networks. For that, a management function for nodes delivered cache files (the delivery DB) and a management function for storing cache file (the cache DB) are implemented on each node.

3 Cache Management Method

3.1 Design Concept

A characteristic of JAMS is to deliver caches of music data to ad-hoc networks in order to deliver music files effectively in networks with mobile data terminals. The authors determined the following three concepts for design of the cache management method.

- Each node manages cache files autonomously.
- Cache replacement occurs according to changes in “Space”.
- An adaptive method is used instead of a statistical method.

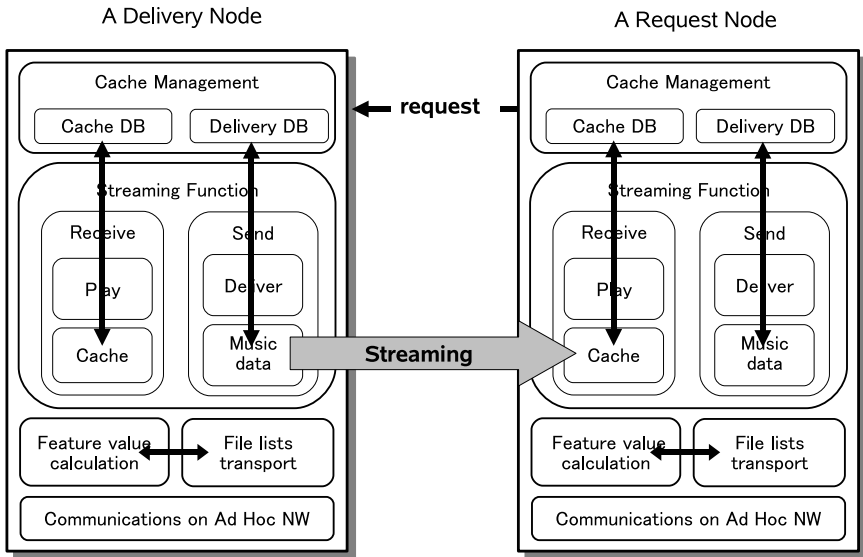


Fig. 2. System architecture

For the first concept, mobile terminals have many restrictions of resources, so that it is necessary to adopt mechanisms with a light load to terminals. In particular, since JAMS is a pure P2P type network service and all nodes are on an equal level, it is impossible to use some stable nodes for special roles. Therefore, distributed management of cache data are needed in order to avoid load processing concentration. For the second concept, it is necessary to adopt a cache delivery method which does not affect localization of contents. JAMS is a system which provides different services according to users' staying "Spaces." Consequently, the method which affects the feature of "Space" cannot be adopted. For the third concept, it is necessary to design an effective cache delivery method which is not affected by inequality of contents. Existing file types in a "Space" change every moment. It is impossible to predict frequency in use of a file beforehand and to use a statistical method.

3.2 Features of the Method

The authors designed the cache management method of JAMS based on the concepts mentioned above. There are two features of the method. One is to conduct distributed management of caches on terminal nodes, and the other is to remove cache data by using time-out. The concrete features are described as follows.

– Cache Delivery:

A passive delivery method (an adaptive method) is used instead of an active delivery one (an statistical method). Music data are stored in a node as

a cache file only when the node receives the music file via streaming from another node.

- Cache Removal:
If there are no delivery requests of a cache until the time-out determined beforehand, the cache is removed from the node automatically.
- Cache Replacement:
LRU (Least Recently Used) is used for replacement. Caches are replaced in the order of their residual time to time-out, shortest first.
- Cache Management:
The source node of a cache manages the destination node of the cache. Hence, relationship between nodes with the cache of a music file can be represented by a tree structure whose root is the node with the original file. Cache search is conducted recursively along the tree.
- Cache selection:
If the number of cache delivery requests to a node exceeds the threshold (ex. 3 for JAMS), the node informs the request node of a managed cache, and distributes cache accesses. Caches are used in order of the layer number from the root of the cache management tree, smallest first. If there are caches in the same layer, those are used in order of their residual time to time-out, shortest first.

As to the cache management, since cache delivery does not occur unless a node receives music data, explicit process for joining new nodes to the ad-hoc network is not needed. On the other hand, when cache removal occurs by exceeding time-out, leaving nodes from the ad-hoc networks and so on, parts of the cache management tree structure collapse. However, the starting point of cache search is the node with the original file, therefore the tree can be constructed again. When the node with the original file leaves from the network, the corresponding file is removed from the search targets, and we just wait for removal of the existing cache files in the network by time-out.

3.3 Procedure of the Method

The procedure of the cache management method is shown in Fig. 3. In this figure, cache delivery process from the node A with the original file is represented. The upper part indicates file delivery process via streaming according to requests and cache storing process in the request nodes. The lower part indicates the structure of the cache management tree. In this figure, parent nodes manage the child nodes connected with them by lines. The authors explain the detail processes as follows.

1. Node B and node C store the cache of node A. At that time, node H requests music delivery to node A. Caches of node B and node C were delivered directly from node A, therefore node B and node C directly connect lines with node A on the cache management tree.

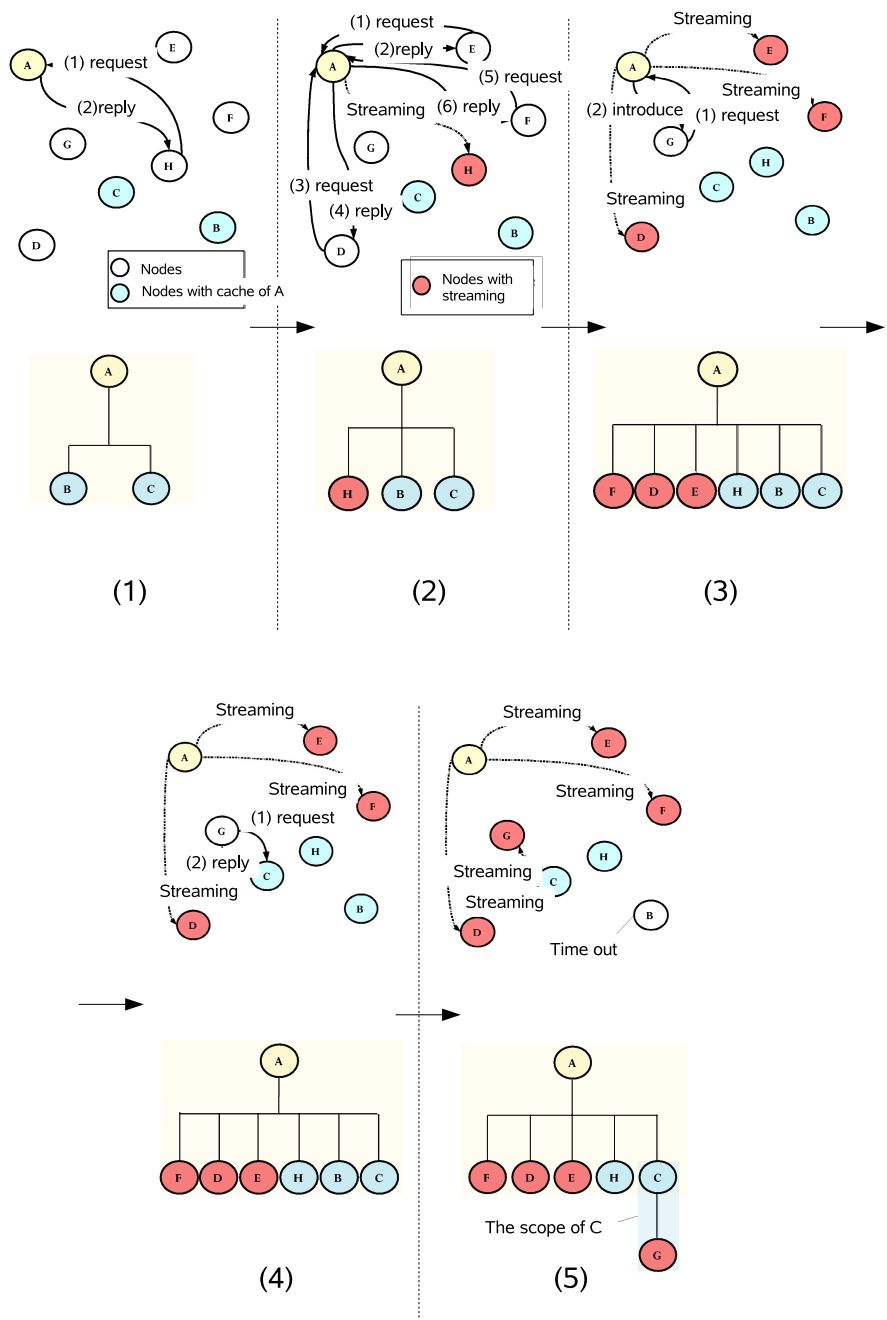


Fig. 3. Procedure of the method

2. Node A delivers music data via streaming to node H. At that time, node E, node D and node F request music delivery to node A. Since node H stores the cache file now, the node is added to the management tree.
3. According to delivery requests, node A delivers music data via streaming to node E, node D and node F. At that time, node G requests music delivery to node A. As for JAMS, since the number of streams delivered simultaneously from one node is three²node A cannot deliver the music data. Therefore, node A informs node G of node C which has the shortest residual time in the cache list of node A. Node D, node E and node F are added to the cache management tree.
4. Node A introduces node C to node G, and then, node G requests music delivery to node C. At that time, node C is available, and delivers music data via streaming to node G.
5. Node G stores the cache file delivered from node C. Since the cache data is managed by node C, node G is added to the cache management tree with direct connection to node C. As for node B, since there are no delivery requests of a cache until the time-out, the cache is automatically removed from the node and the cache management tree.

4 Performance Evaluation

In order to verify the effectiveness of the proposed method, the authors developed a simulator for the cache delivery function, and conducted performance evaluation. This section shows the experimental results.

4.1 Experimental Conditions

This paper confirms that the proposed method is not affected by the change of “Spaces”, and is a robust and effective cache delivery scheme under various system environments. For that purpose, several situations which have different conditions are provided, and performance data are measured under the conditions. The ratio of delivery requests to delivery failures is used for performance evaluation metrics. It is defined as follows.

$$\text{Failure ratio} = \frac{\text{The number of delivery failures}}{\text{The number of delivery requests}} \quad (1)$$

In this paper, this value was obtained for each node, and the average of them was calculated. As for JAMS, there is a limit of the number of streams delivered simultaneously from one node, so the failure ratio might become high unless effective cache delivery is made.

² The number of streams delivered simultaneously from one node depends on a performance characteristic of a mobile terminal, available bandwidth and so on. As for JAMS, the number is three, which was determined by the prior experiment using PCs. In the case of using cell phones, this value is also supposed.

The experimental condition is described in Tab. 1. Arrival rates use a Poisson distribution and sojourn time uses a normal distribution with the mean values indicated in the table. Nodes generating requests, requested nodes and requested music files in a node are determined randomly. For the comparison, the failure ratios were measured by using the following four methods.

Table 1. Experimental condition

No.	Situation	Arrival rate	Sojourn time	Feature
1	Cafeteria	0.0067/s	2700 s	A few nodes and long sojourn time.
2	Platform	0.5/s	300 s	A lot of nodes and short sojourn time.
3	Library	0.0017/s	5400 s	A few nodes and long sojourn time.
4	Train	0.167/s	1200 s	A lot of nodes and short sojourn time.

- Method A: This is the proposed method in this paper. It is the method that a cache is managed by the node which delivers the cache data (the parent node).
- Method B: This does not use cache mechanisms.
- Method C: This is the method that a cache is managed by the node which has the original file of the cache data (the original node).
- Method D: This is the method that cache data are delivered beforehand according to probability.

As for method D, the experiment classified nodes in the “Space” according to similarity of music files (in this case, music genres were used for the classification), and the source node of cache and the destination node were randomly selected in the class with the largest number of nodes. Then, suitable files to music features of the class were delivered as caches. This simulates the situation that files with the highest probability of selection are delivered as caches in advance.

4.2 Experimental Results

The simulation experiment corresponding to six hours was conducted under the situations in Tab. 1. Fig. 4 shows the experimental result. This figure indicates the average of the failure ratios for the situations (the average of the failure ratios of nodes) and the standard deviations of the averages for the situations (σ). As a result, the authors confirmed that failure ratio of the proposed method (method A) is the lowest for all situations and the method makes it possible to deliver caches effectively. Moreover, they found that changes in situations have a little effect on failure ratio for the method and it is a robust method for various situations.

The following are discussions on the experimental results. First, in the situations for the experiment, the failure ratios of situation 1 and situation 3 (sojourn times in “Spaces” are long, and the number of join and leave nodes is small) are

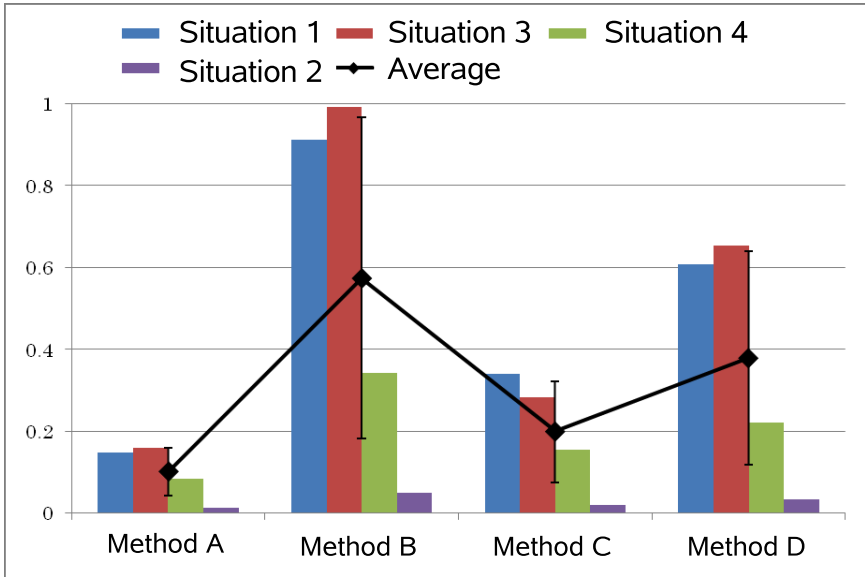


Fig. 4. Experimental results

high. This is because concentration of access to popular contents occurs after many nodes stay in a “Space” within a certain period, and cache does not work well with a lot of joining and leaving nodes. Second, the result on each method is discussed. For method B, which does not use cache mechanisms, the failure ratios of situation 1 and situation 3 are nearly 1. As a result, we found that it is necessary to implement cache mechanisms for effective music delivery. For method D, which delivers cache data to networks beforehand according to prediction of frequency in use of a file, there is not remarkable effect of cache delivery because situation change frequently occurs in JAMS (i.e. it is difficult to predict frequency in use of a file) though prediction accuracy greatly affects failure ratio on the method. We also found that failure ratios vary widely for situations. For method C, which manages a cache file in the original node, failure ratio is high because the next cache node is not introduced if access failure to a cache file occurs.

We conclude that the proposed method can deliver cache files effectively to a network under various conditions and can lower failure ratio of file access.

5 Conclusion

This paper proposed a effective cache management method for a music delivery system JAMS, which uses ad-hoc networks with mobile terminals for localization services. The proposed scheme makes it possible to manage cache files robustly by using adaptive cache delivery according to each node request. Moreover, by

conducting a simulation experiment and obtaining access failure ratio of music files under various system conditions, the authors verified the effectiveness of the proposed method.

References

1. Shibata, H., Tomisawa, S., Endo, H., Kato, Y.: A p2p network system for music delivery using localization of contents. In: IPSJ DPS Workshop 2007, pp. 37–42 (2007)
2. Shibata, H., Tomisawa, S., Endo, H., Watabe, T., Oshiro, H., Kato, Y.: A mobile music delivery system using field features of contents. IPSJ SIG-DPS DPS-133, 25–30 (2007)
3. Corbett, D.J., Cutting, D.: AD LOC: Collaborative Location-based Annotation. IPSJ Journal 48(6), 2052–2064 (2007)
4. Nakanishi, Y., Takahashi, K., Tsuji, T., Hakozaiki, K.: iCAMS: A mobile communication tool using location and schedule information. IEEE Pervasive Computing 3(1), 82–88 (2004)
5. Kaneko, I. (ed.): Technologies of Winny. ASCII (2005)
6. Hara, T.: Replica allocation for correlated data items in ad-hoc sensor networks. ACM SIGMOD 33(1), 38–43 (2004)
7. Yin, L., Cao, G.: Supporting cooperative caching in ad hoc networks. IEEE Trans. on Mobile Computing 5(1), 77–89 (2006)

EcoRare: An Economic Incentive Scheme for Efficient Rare Data Accessibility in Mobile-P2P Networks

Anirban Mondal¹, Sanjay Kumar Madria², and Masaru Kitsuregawa¹

¹ Institute of Industrial Science
University of Tokyo, Japan

{anirban,kitsure}@tkl.iis.u-tokyo.ac.jp

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

Abstract. We propose EcoRare, a novel economic incentive scheme for improving the availability of **rare data** in Mobile-P2P networks. EcoRare combats free-riding and effectively involves free-riders to improve the availability (and lifetimes) of rare data. EcoRare also facilitates the creation of multiple copies of rare items in the network since its novel selling mechanism allows a given data to have multiple owners. Our performance study demonstrates that EcoRare indeed improves query response times and availability of rare data items in Mobile-P2P networks.

1 Introduction

In a Mobile Ad-hoc Peer-to-Peer (M-P2P) network, mobile peers (MPs) interact with each other in a peer-to-peer (P2P) fashion. Proliferation of mobile devices (e.g., laptops, PDAs, mobile phones) coupled with the ever-increasing popularity of the P2P paradigm (e.g., Kazaa) strongly motivate M-P2P applications. M-P2P applications facilitate mobile users in sharing information *on-the-fly* in a P2P manner. Mobile devices with support for wireless device-to-device P2P communication are beginning to be deployed such as Microsoft's Zune [13].

This work focusses on improving the availability of *rare* data items. *Rare* items are those, which get sudden bursts in accesses based on *events*. In the absence of related events, rare items are generally of little interest to most M-P2P users, hence they are typically hosted by relatively few MPs, who obtain them from content distributors. For example, in case of an unexpected event such as a gas emission, several users would be interested in knowing where gas-masks are available. This would lead to sudden bursts in accesses to data concerning gas-masks, and users would *urgently* need this data in *real-time*. Peers, which host data concerning gas-masks, obtain such data as advertisements from content providers (e.g., shops selling gas-masks). Observe that the sudden bursts in accesses to rare items generally occurs with a given time-frame, before and after which rare items are accessed rarely. Similarly, in case of a sudden snowfall, many people would want to know in *real-time* where shovels are available, thereby resulting in

a sudden burst of accesses to data concerning shovels. Peers hosting data about shovels obtain such data as advertisements from shops that sell shovels. Such M-P2P interactions for effective sharing of rare data are currently not freely supported by existing wireless communication infrastructures.

In the same vein, during a sudden heavy snowfall in an area usually having negligible snowfall, many cars are likely to slip and require some repairs due to the snow since most cars in areas of negligible snowfall are not typically equipped with snow-related features such as snow tires. Thus, many drivers would require *real-time* information about car repairs. Peers hosting data about car repairs obtain them as advertisements from car-repair garages. Moreover, when an art exhibition takes place in a city, many people attending the exhibition may become interested in obtaining more knowledge about paintings, which would result in sudden bursts in accesses to data concerning paintings. Peers, who host data about rare paintings, may have obtained the data from art connoisseurs and shops selling art-related items. Observe how such sudden interest of several users in paintings or in car repairing arises due to *events*. Incidentally, our target applications mainly concern slow-moving objects e.g., people in art exhibitions.

Data availability in M-P2P networks is typically lower than in fixed networks due to frequent network partitioning arising from user movement and mobile devices switching ‘off’ when their generally limited energy is drained. Since a large percentage of MPs are typically free-riders [14] (i.e., they do not provide any data), the limited energy of a typically small percentage of MPs, which provide data, is quickly drained, thereby further reducing M-P2P data availability. Availability of rare data is further exacerbated since they are generally stored at relatively few MPs, which may be several hops away from query issuers. Notably, existing M-P2P incentive schemes [26] do not consider rare data. Hence, we propose **EcoRare**, a novel *economic incentive scheme*, in which MPs collaborate effectively with each other to improve the availability of rare data.

In EcoRare, each data item is associated with two (*virtual-currency*) prices, namely the *use-price* ρ_u and the *sell-price* ρ_s . Paying the *use-price* entitles the query issuing MP M_I to obtain some basic information about its queried data item d , but M_I does not obtain the ownership of d i.e., it cannot sell d . On the other hand, by paying the *sell-price*, M_I obtains more detailed information about d and it obtains ownership rights of d , hence it can sell d . Thus, in EcoRare, a given data can have multiple owners. Such multiple owners hosting the same data guard the data availability against the unavailability of some of the owners. Observe that if M_I pays the *use-price*, it cannot sell d because it does not have complete information concerning d . Intuitively, given a data d , the *sell-price* of d is always higher than its *use-price*.

To put things into perspective, let us refer to our application scenarios. When M_I pays the *use-price* for obtaining information about gas-masks and shovels, it obtains information about only a few shops selling these items and the respective prices of these items at those shops. However, when M_I pays the *sell-price*, it obtains additional information including complete catalogues of more shops selling these items, information about how to order these items (e.g., by phone)

and how to get these items delivered to M_I . Observe that in the above scenarios, the peer M_S that hosts the queried data essentially acts as an agent for the content-provider. Thus, the prices paid by M_I to M_S may be viewed as a *commission* for giving M_I information about the queried data. Similarly, M_S may be regarded as disseminating advertisements on behalf of the content-provider.

In case of paintings, paying the *use_price* entitles M_I to obtain some basic information concerning the art shops, where replicas of the paintings can be purchased. However, when M_I pays the *sell_price*, it obtains additional information such as how to order the paintings and get them delivered, digital images of collection of paintings by the same artist, educational documentary videos about the historical and cultural aspects of the paintings and so on. For the car-repairing application scenario, the *use_price* allows M_I to know only a few addresses of car repair garages, while the *sell_price* enables M_I to get additional information e.g., a video showing how to change a tire.

In essence, EcoRare requires a data-requestor to pay either the *use_price* or the *sell_price* to the data-provider. EcoRare also requires query issuing MPs to pay a constant relay cost to each MP in the successful query path to entice relay MPs in forwarding queries quickly. Thus, EcoRare effectively combats free-riding because free-riders would have to earn currency for issuing their own requests, and they can earn currency only by hosting items and relaying messages. As we shall see later, rare items are higher-priced than non-rare items in terms of both *use_price* and *sell_price*. This provides free-riders with higher incentive [23,8,20] to host rare items for maximizing their revenues. By enticing free-riders to pool in their energy and bandwidth resources to host rare items, EcoRare improves the availability and lifetimes of rare items due to multiple ownership and the creation of multiple copies. Incidentally, querying in EcoRare proceeds via broadcast which is limited to 8 hops to optimize communication overheads.

EcoRare considers both read-only and updatable data items. A rare item owner can send the subsequent updates under a contract for a period of time to the MPs, which have paid it the *sell price* for the data. For our application scenarios, in case gas-masks or shovels go out of stock at some of the shops, the previous owner of the data can send this information to the MP(s) that it had previously sold the data to. Updates can also be in the form of appending new information e.g., for the car-repair application scenario, the data owner could provide the addresses of more car garages, where a slot for car repairing is currently available. Moreover, the current buyer can also get the direct feed from the content provider (e.g., a car-repair garage).

Incidentally, virtual currency incentives are suitable for P2P environments due to the high transaction costs of real-currency micro-payments [24]. The works in [5,6,28] discuss how to ensure secure payments using a virtual currency. Notably, these secure payment schemes are complementary to our proposal, but they can be used in conjunction with our proposal.

The main contributions of EcoRare are three-fold:

1. It combats free-riding and effectively involves free-riders to improve the availability (and lifetimes) of rare data.

2. It facilitates the creation of multiple copies of rare items in the network since its selling mechanism allows a given data to have multiple owners.
3. It indeed improves query response times and availability of rare data items in M-P2P networks, as shown by a detailed performance evaluation.

To our knowledge, this is the first work to propose an M-P2P economic incentive scheme for rare items.

2 Related Work

Economic models for resource allocation in distributed systems [7,15] do not address unique M-P2P issues such as node mobility, free-riding and frequent network partitioning. Economic schemes for resource allocation in wireless ad hoc networks [17,27] do not consider free-riding. Schemes for combating free-riding in static P2P networks [9,11,14,16,18] are too static to be deployed in M-P2P networks as they assume peers' availability and fixed topology.

Schemes for improving data availability in mobile ad hoc networks (MANETs) (e.g., the **E-DCG+** approach [12]) primarily focus on replica allocation, but do not consider economic incentive schemes and M-P2P architecture. Interestingly, the proposals in [25,26] consider economic schemes in M-P2P networks. However, they focus on data dissemination with the aim of reaching as many peers as possible, while we consider on-demand services. Furthermore, the works in [25,26] do not consider free-riders and data item rarity issues.

The works in [2,28] use virtual currency to stimulate the cooperation of mobile nodes in forwarding messages, but they do not consider prices of data items and data item rarity issues. The proposals in [4,22] concentrate on compensating forwarding cost in terms of battery power, memory, CPU cycles, but they do not entice free riders to host data.

P2P replication suitable for mobile environments has been incorporated in systems such as ROAM [19], Clique [21] and Rumor [10]. However, these systems do not incorporate economic schemes. MoB [3] is an open market collaborative wide-area wireless data services architecture, which can be used by mobile users for opportunistically trading services with each other.

3 EcoRare: An M-P2P Economic Scheme for Rare Data

This section discusses the economic scheme of EcoRare. In particular, we present the formulae for *use_price*, *sell_price* and the revenue of an MP.

Computation of the *use_price* ρ_u

Table 1 summarizes the notations used in this paper. Using the notations in Table 1, the *use_price* ρ_u of a data item d is computed below:

$$\begin{aligned} \rho_u &= \int_{t_1}^{t_2} (\lambda \times e^{\tau_D/\tau_R} dt) \quad \text{if } \tau_R \leq \tau_D \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{1}$$

Table 1. Summary of Notations

Symbol	Significance
d	A given rare data item
ρ_u	The <i>use_price</i> of d
ρ_s	The <i>sell_price</i> of d
λ	Rarity score of a data item
τ_D	Time of query deadline
τ_R	Time of query response
n_{r_i}	Number of read access to d during the i^{th} time period
N_{Copies}	The total number of copies of a data item in the network

where $[t_2 - t_1]$ is a given time period. τ_R and τ_D are the query response times and the query deadlines respectively. The term e^{τ_D/τ_R} implies that faster response times (w.r.t. the query deadline) for queries on d command higher price, which is consistent with ephemeral M-P2P environments. For queries answered after the deadline τ_D , $\rho_u = 0$ as the query results may no longer be useful to the user.

Rarity score λ of a data item d quantifies its rarity. λ depends upon the number of MPs which host d , and the variability in access frequencies of d during the past N periods of time e.g., shovels for removing snow are heavily accessed only during a specific time-frame associated with the sudden snowfall, while at other times, they may not be accessed at all. λ is computed below:

$$\lambda = (\theta \times (max_{\eta_r} - min_{\eta_r}) \times NP) / (\sum_{i=1}^N \eta_i) \quad (2)$$

where N is the number of time periods over which λ is computed. $\theta = (max_{\eta_w} - min_{\eta_w} + 1)$, where max_{η_w} and min_{η_w} are the maximum and minimum number of write accesses to d during any of the last N time periods. Thus, for read-only items, $\theta = 1$. max_{η_r} and min_{η_r} are the maximum and minimum number of read-accesses to d during any of the past N time periods. NP is the ratio of the total number of MPs in the network to the number of MPs that host d . Observe that λ decreases with increasing number of MPs hosting d . This is because a data item becomes less rare when it is hosted at more MPs, thus its rarity score λ also decreases. Each MP knows the value of NP since every MP *periodically* broadcasts its list of data items so that each MP knows the data items that are hosted at other MPs. η_i is the total access frequency (i.e., total read accesses for read-only items and the sum of reads and writes for read-write items) during the i^{th} time period. Thus, the term $(\sum_{i=1}^N \eta_i)$ represents the total access frequency of d during the last N time periods.

Computation of the *sell_price* ρ_s

Using the notations in Table 1, the *sell_price* ρ_s of an item d is computed below:

$$\rho_s = \rho_u \times ((\sum_{i=1}^N n_{r_i}) / N_{Copies}) \quad (3)$$

Observe that ρ_s of a given item d depends upon the *use_price* ρ_u of d . This helps in ensuring that ρ_s always exceeds ρ_u , which is necessary in case of our application scenarios. Here, n_{r_i} is the number of read-accesses to d during the i^{th} time-period. Thus, the term $(\sum_{i=1}^N n_{r_i})$ represents the total number of reads to d during the last N time-periods. N_{copies} is the total number of copies of d in the network. Every MP knows the value of N_{copies} since every MP *periodically* broadcasts its list of data items so that each MP knows the data items that are hosted at other MPs. Notably, the number of read-accesses to d must always exceed the number of copies of d hosted in the network. This is because MPs have limited memory, hence they will not host data items, whose number of read accesses is relatively low. Due to memory space constraints, each MP tries to host data items with relatively higher number of read accesses to maximize its revenue from hosting the items. Thus, the term $(\sum_{i=1}^N n_{r_i}) / N_{copies}$ essentially represents a factor, which is always greater than 1. This guarantees that the *sell_price* ρ_s of any given item d always exceeds its *use_price* ρ_u .

Revenue of an MP

Revenue of an MP is defined as the difference between the amount of virtual currency that it earns (by hosting data items and providing relay services) and the amount that it spends (by requesting data items and paying relay costs to the MPs in the successful query path).

Suppose MP M hosts p data items, and for the i^{th} item, the number of queries (served by M) corresponding to *use_price* is h_{u_i} , while the i^{th} item's *use_price* is ρ_{u_i} . On the other hand, let the price of the i^{th} item and its access frequency corresponding to *sell_price* be ρ_{s_i} and h_{s_i} respectively. Now suppose the number of items in the network not hosted by M is q . M issues r_{u_i} queries corresponding to *use_price* for the i^{th} such item, while the i^{th} item's *use_price* is ρ_{u_i} . Moreover, let the price of the i^{th} item and its access frequency corresponding to *sell_price* be ρ_{s_i} and r_{s_i} respectively for the items requested by M .

Assume MP M relays m messages and requires to pay relay commissions for n messages in the course of issuing different queries. Thus, M 's earnings from relaying messages equals $(m \times K)$, while M 's spending on relay cost is $(n \times K)$. Here, K is a constant, which is a small percentage of the *use_price* ρ_u of the data item corresponding to the relay message. In this work, we consider K to be 5% of ρ_u . Observe that EcoRare provides higher incentive to MPs for hosting items than for relaying messages. This is in consonance with EcoRare's aim of improving the availability of rare items by encouraging MPs to host such items. Revenue ω of an MP is computed below:

$$\omega = \left(\sum_{i=1}^p (\rho_{u_i} \times h_{u_i}) + \sum_{i=1}^p (\rho_{s_i} \times h_{s_i}) \right) - \left(\sum_{i=1}^q (\rho_{u_i} \times r_{u_i}) + \sum_{i=1}^q (\rho_{s_i} \times r_{s_i}) \right) + ((m - n) \times K)$$

In the above equation, the first and second terms represent MP M 's earnings, while the third and fourth terms indicate M 's spending. The last term relates to the earnings and spending of M due to relay commissions.

4 Data Selling Mechanism of EcoRare

This section discusses the data selling mechanism of EcoRare such that multiple copies of the rare data are created, which improves accessibility to rare data.

In EcoRare, data selling works in the following two ways:

- *Query-based selling*: When a query-issuing MP M_I pays the *sell_price* of a data item d to the host MP M_S of d , M_I obtains a copy of d and also becomes an owner of d , thus M_I can now host and re-sell d .
- *Push/Pull-based selling*: When a data-providing MP's energy becomes low or when it is about to leave the network, it may decide to sell its rare data items to earn currency (i.e., a *push-based selling mechanism*). On the other hand, due to EcoRare's economic model, free-riders need to earn currency, without which they would not be able to issue any requests of their own. Hence, free-riders may want to buy one or more rare data items from the data-providing MPs so that they can earn currency by hosting these items (i.e., a *pull-based selling mechanism*). Periodically, data-providing MPs (including MPs that have obtained data via the selling and re-selling mechanisms) broadcast their list of rare items to facilitate both push and pull-based selling mechanisms.

Observe that both the *query-based* and the *push/pull-based* selling mechanisms are essentially equivalent to the data provider selling the data item d such that both become owners of d . Thus, EcoRare allows multiple owners for any given item d , and each owner is allowed to host and sell/re-sell d . Moreover, both these mechanisms effectively create multiple copies of a given item d , thereby improving the accessibility to rare items. In the absence of a selling mechanism, rare items would become inaccessible once their original providers run out of energy. Furthermore, multiple copies of rare items also facilitate shorter query paths, thereby further improving both query response times and data availability.

Given that both these selling mechanisms have many similarities, we present their algorithms in a unified manner. In these algorithms, we shall refer to the data-providing MPs as the **sellers**, while the query-issuing MPs (in case of the query-based selling mechanism) or the free-riders (in case of the push/pull-based mechanism) shall be referred to as the **buyers**.

Figure 1 depicts the algorithm for a seller-MP M_{Sell} . In Line 1 of Figure 1, $\phi = (\lambda_d \times \eta_d)$, where λ_d (computed by Equation 2) is the rarity score of item d and η_d is d 's access frequency. Observe that ϕ reflects the revenue-earning potential of d since d 's price increases with increase in both λ_d and η_d . Notably, items with relatively high update frequencies are not considered for sale. This is because the sale of frequently updated items would incur significant energy and bandwidth overheads for the seller MP due to continuous update transmissions. As Lines 1-3 indicate, M_{Sell} sells items with relatively higher values of ϕ to maximize

Algorithm *EcoRare_Seller_MP*

```

/*  $\phi$  is an item's revenue-earning potential */
(1) Sort all its data items in descending order of  $\phi$ 
(2) Compute the average value  $\phi_{avg}$  of all its data items
(3) Select items for which  $\phi$  exceeds  $\phi_{avg}$  into a list Sell
    /* Sell is the candidate set of items for selling */
(4) Broadcast the list Sell upto its  $n$ -hop neighbours

(5) for each data item  $d$  in Sell
(6)   Wait for replies from potential buyers
(7)   Receive replies from buyers into a list Sell_List
(8)   if Sell_List is non-empty
(9)     for each MP  $M$  in Sell_List
(10)      Sell  $d$  to  $M$  by sending  $d$  to  $M$  and granting  $M$  ownership of  $d$ 
(11)      Obtain the sell_price of  $d$  from  $M$ 
end

```

Fig. 1. EcoRare algorithm for seller-MP

its revenue from sales, while optimizing its energy consumption for transmitting sold items to buyers. In Line 4, M_{Sell} 's broadcast message contains not only the items for sale, but also the price, rarity score and recent access history of each item. Moreover, this broadcast is limited only upto n -hops to reduce communication overheads. For our application scenarios, we found a value of $n = 4$ to be reasonable. In Lines 5-11, M_{Sell} receives replies from potential buyers, and sells items to these buyers in lieu of the respective *sell_prices* of the items.

Figure 2 depicts the algorithm for a buyer-MP M_{Buy} . In Lines 1-4 of Figure 2, M_{Buy} receives broadcast messages from different seller-MPs, collates this broadcast information and sorts the items for sale in descending order of ϕ , which is the revenue-earning potential of an item. In Lines 5-10, observe how M_{Buy} prefers to select items with higher revenue-earning potential ϕ for hosting for maximizing its revenue per unit of its memory space since its memory space is limited. Thus, M_{Buy} greedily *simulates* the filling up of its memory space by items with higher value of ϕ . In Lines 11-13, having selected the items to buy, M_{Buy} obtains the items from their corresponding sellers and pays the *sell_price* of these items to the sellers. In case M_{Buy} does not have adequate currency to make the payment, it is allowed to make the payment after it has earned currency by hosting these items. This policy of allowing deferred payments allows free-riders, which may initially not have enough currency to buy items, to seamlessly integrate into participating in the network.

5 Performance Evaluation

This section reports our performance evaluation. MPs move according to the *Random Waypoint Model* [1] within a region of area 1000 metres $\times$ 1000 metres. The *Random Waypoint Model* is appropriate for our application scenarios,

Algorithm *EcoRare_Buyer_MP**Sell_i*: Candidate data items for sale from seller-MP *i**Spc*: Its available memory space

- (1) for each seller-MP *i*
 - (2) Receive broadcast message from *i* containing the set *Sell_i*
 - (3) Append all data items in *Sell_i* to a set *bigSell*
 - /* ϕ is an item's revenue-earning potential */
 - (4) Sort all data items in *bigSell* in descending order of ϕ
 - (5) for each data item *d* in *bigSell*
 - (6) while *Spc* > 0
 - (7) /* *size_d* is the size of *d* */
 - (8) if (*size_d* ≤ *Spc*)
 - (9) Add *d* to a set *Buy*
 - (10) *Spc* = *Spc* - *size_d*
 - (11) for each data item *d* in set *Buy*
 - (12) Send the *sell_{price}* of *d* as payment to the corresponding data-provider of *d*
 - (13) Receive *d* (including re-sell rights) for *d* from the corresponding data-provider of *d*
- end**

Fig. 2. EcoRare algorithm for buyer-MP

which involve random movement of users. A total of 100 MPs comprise 15 data-providers and 85 free-riders (which provide no data). Communication range of all MPs is a circle of 100 metre radius. A time-period is 180 seconds, and *periodically*, every 180 seconds, MPs exchange messages to inform each other concerning the items that they host. Table 2 summarizes our performance study parameters.

Table 2. Performance Study Parameters

Parameter	Default value
No. of MPs (N_{MP})	100
Zipf factor (ZF) for queries	0.9
Queries/second	20
Bandwidth between MPs	28 Kbps to 100 Kbps
Probability of MP availability	50% to 85%
Size of a data item	50 Kb to 350 Kb
Memory space of each MP	1 MB to 1.5 MB
Speed of an MP	1 metre/s to 10 metres/s
Size of message headers	220 bytes

Each data-provider owns 10 data items of varying sizes with different rarity scores. We create three classes of items, namely *rare*, *medium-rare* and *non-rare*. The number of items in each of these classes is decided using a highly skewed Zipf

distribution (zipf factor = 0.9) over three buckets to ensure that the majority of items in the network are *rare* in that they will be assigned high rarity scores.

We assign the rarity score λ_d to each item d as follows. For *rare* items, $0.7 \leq \lambda_d \leq 1$; for *medium-rare* items, $0.5 \leq \lambda_d < 0.7$; for *non-rare* items, $0 < \lambda_d < 0.5$. Hence, for each item d , we randomly assign the value of λ_d based on the lower-bounds and upper-bounds of d 's class. Among the 15 data-providers, each *rare* item is assigned randomly to only one MP, each *medium-rare* item is randomly assigned to 1-2 MPs, while each non-rare item is randomly assigned to 3-4 MPs. Notably, the actual value of λ_d used for computing item prices during the experiments is based on Equation 2, hence the above method is only used to ensure that the majority of items in the network are rare. Furthermore, 90% of the items are read-only, while the other 10% are read-write items. We decide which items are read-only based on a random number generator.

Each query is a request for a single data item. 20 queries/second are issued in the network. Items to be queried are randomly selected from all the items in the entire network. Number of queries directed to each class of items (i.e., *rare*, *medium-rare* and *non-rare*) is determined by a highly skewed Zipf distribution with Zipf factor of 0.9 to ensure that the vast majority of queries are directed towards *rare* items. This is consistent with our application scenarios, which involve sudden bursts in accesses to rare items. 90% of the queries are for read-only items. Although selling is also applicable to updatable items (albeit with low update frequencies), since most queries in our experiments are for read-only items, the selling mechanism impacts MP participation significantly. 70% of queries correspond to *sell_price*, while the rest correspond to *use_price*.

Performance metrics are **average response time (ART)** of a query and **data availability (DA)**. ART equals $((1/N_Q) \sum_{i=1}^{N_Q} (T_f - T_i))$, where T_i is the query issuing time, T_f is the time of the query result reaching the query issuing MP, and N_Q is the total number of queries. ART includes data download time, and is computed only for successful queries. (Unsuccessful queries die after TTL of 8 hops, hence they are not considered for ART computations.) *DA* equals $((N_S/N_Q) \times 100)$, where N_S is the number of successful queries and N_Q is the total number of queries. Queries can fail due to MPs being switched 'off' due to running out of energy or due to network partitioning.

Existing M-P2P economic incentive schemes do not consider data rarity issues. As reference, we adopt an economic scheme designated as **Econ**. Econ is an economic incentive-based model, which uses data item pricing formulae similar to that of EcoRare, the difference being that Econ's item pricing formulae do not consider rarity scores λ of data items, and Econ supports neither selling of items nor multiple ownership. Like EcoRare, Econ performs querying via broadcast.

Effect of Economic Incentives

To examine the effect of economic incentives, let us first compare EcoRare with a **non-economic non-incentive scheme (Non-Econ)**. Non-Econ provides no incentives for MP participation and it does not prefer rare items. It does not support selling of items. We define threshold revenue Rev_{TH} as 50% of the

average revenue of the 15 data-providers in the network. The results in Figure 3 indicate that when the number N_{TH} of MPs above Rev_{TH} increases, ART and DA improve for EcoRare. This is due to more free-riders providing service as MP revenues increase. To earn revenue for issuing their own requests, free-riders host items (by paying *sell_price* to data-providers) to increase their revenues.

Higher MP participation results in multiple paths to locate queried items. Hence, queries can be answered within lower number of hops, thereby decreasing ART. The existence of multiple query paths better preserves accessibility to rare items, thereby improving DA. In contrast, Non-Econ shows constant performance since it is independent of revenue, and it is outperformed by EcoRare because it does not provide any incentives for improving MP participation.

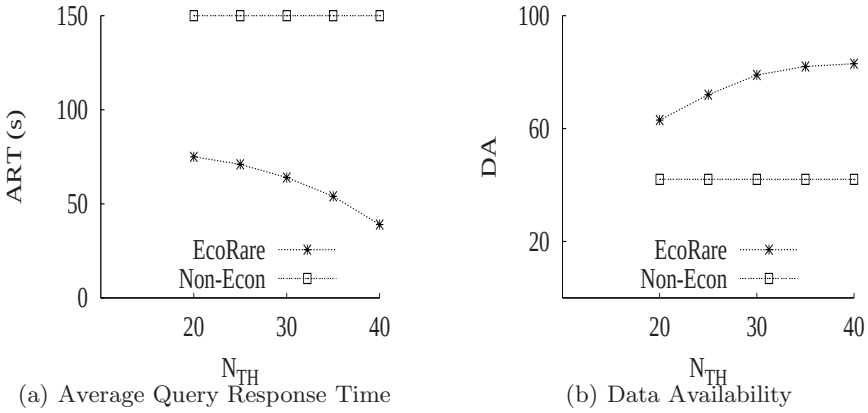


Fig. 3. Effect of economic incentives

Performance of EcoRare

Figure 4 depicts the performance of EcoRare using default values of the parameters in Table 2. As more queries are answered, the energy of MPs decreases and increasing number of MPs run out of energy, hence query paths to data items become longer or inaccessible. Hence, ART increases and DA decreases for both EcoRare and Econ. With increasing number of queries, network congestion and overloading of some of the data-providers also increases ART.

The performance of EcoRare degrades at a considerably slower rate than that of Econ because EcoRare's selling mechanism (via the *sell_price*) results in the creation of multiple copies of rare items, thereby ensuring that rare items are hosted at some MP. In contrast, since Econ does not consider any selling mechanism, rare items become inaccessible when their providers run out of energy.

MP Participation and Rare Item Lifetimes

Figure 5a depicts the percentage of host MPs in the M-P2P network over time. An MP is regarded as a host MP if it hosts a data item, which is accessed at

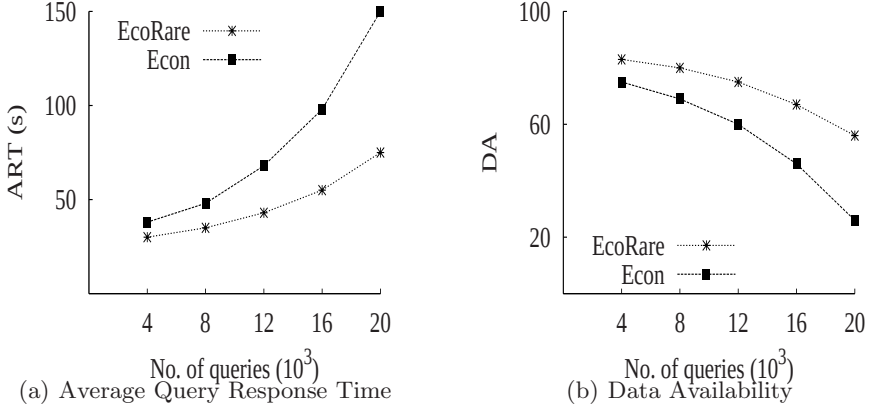


Fig. 4. Performance of EcoRare

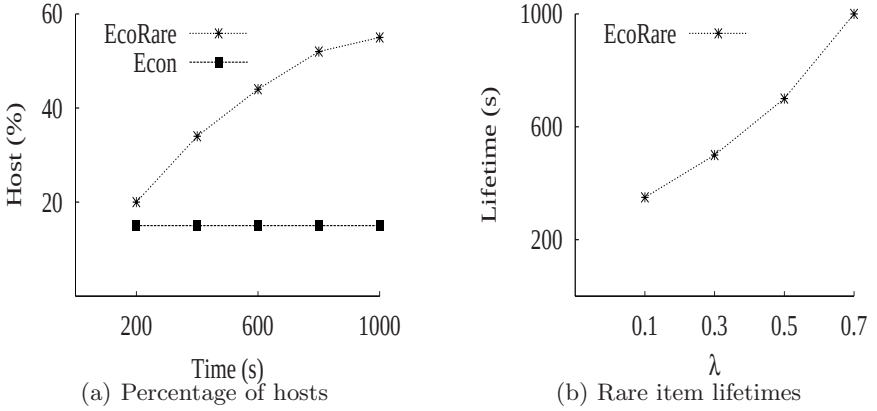


Fig. 5. MP participation and rare item lifetimes

least once (either via the *use_price* or the *sell_price* mechanism) during a given time period. In EcoRare, the selling mechanism entices the effective conversion of free-riders to host MPs. The percentage of host MPs plateaus over time due to the majority of the free-riders having been already converted to host-MPs and owing to some of the free-riders running out of energy. In contrast for Econ, the percentage of hosts remains relatively constant over time since only the data-providers host data due to the absence of a selling mechanism. In essence, EcoRare's selling mechanism increases MP participation levels upto the point where the majority of the MPs are providing service to the network.

Figure 5b depicts the results when the rarity score λ of data items is varied. For this experiment, we normalized the values of λ of different data items such that $0 < \lambda \leq 1$ for each item d . We selected five data items corresponding to each value of λ and computed the average of their lifetimes. Higher value of λ implies

higher rarity score of data item d . Observe that lifetimes of items increase with increasing value of λ because rare items are higher-priced, hence they are more likely to be hosted by free-riders for maximizing revenues. We do not present the results of Econ for this experiment as it does not consider item rarity.

6 Conclusion

We have proposed EcoRare, a novel economic incentive scheme for improving the availability of **rare data** in M-P2P networks. EcoRare combats free-riding and effectively involves free-riders to improve the availability (and lifetimes) of rare data. Moreover, EcoRare facilitates the creation of multiple copies of rare items in the network since its novel selling mechanism allows a given data to have multiple owners. Our performance study shows that EcoRare indeed improves query response times and availability of rare data items in Mobile-P2P networks.

References

1. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocol. In: Proc. MO-BICOM (1998)
2. Buttyan, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. ACM/Kluwer Mobile Networks and Applications 8(5) (2003)
3. Chakravorty, R., Agarwal, S., Banerjee, S., Pratt, I.: MoB: a mobile bazaar for wide-area wireless services. In: Proc. MobiCom (2005)
4. Crowcroft, J., Gibbens, R., Kelly, F., Ostring, S.: Modelling incentives for collaboration in mobile ad hoc networks. In: Proc. WiOpt (2003)
5. Daras, P., Palaka, D., Giagourta, V., Bechtsis, D.: A novel peer-to-peer payment protocol. In: Proc. IEEE EUROCON, vol. 1 (2003)
6. Elrufaie, E., Turner, D.: Bidding in P2P content distribution networks using the lightweight currency paradigm. In: Proc. ITCC (2004)
7. Ferguson, D.F., Yemini, Y., Nikolaou, C.: Microeconomic algorithms for load balancing in distributed computer systems. In: Proc. ICDCS, pp. 491–499 (1988)
8. Garyfalos, A., Almeroth, K.C.: Coupon based incentive systems and the implications of equilibrium theory. In: Proc. IEEE International Conference on E-Commerce Technology proceedings (2004)
9. Golle, P., Brown, K.L., Mironov, I.: Incentives for sharing in peer-to-peer networks. In: Proc. Electronic Commerce (2001)
10. Guy, R., Reiher, P., Ratner, D., Gunter, M., Ma, W., Popek, G.: Rumor: Mobile data access through optimistic peer-to-peer replication. In: Proc. ER Workshops (1998)
11. Ham, M., Agha, G.: ARA: A robust audit to prevent free-riding in P2P networks. In: Proc. P2P, pp. 125–132 (2005)
12. Hara, T., Madria, S.K.: Data replication for improving data accessibility in ad hoc networks. IEEE Transactions on Mobile Computing 5(11) (2006)
13. <http://www.microsoft.com/presspass/presskits/zune/default.mspx>
14. Kamvar, S., Schlosser, M., Garcia-Molina, H.: Incentives for combatting free-riding on P2P networks. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003. LNCS, vol. 2790, pp. 1273–1279. Springer, Heidelberg (2003)

15. Kurose, J.F., Simha, R.: A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Computers* 38(5), 705–717 (1989)
16. Liebau, N., Darlagiannis, V., Heckmann, O., Steinmetz, R.: Asymmetric incentives in peer-to-peer systems. In: *Proc. AMCIS* (2005)
17. Liu, J., Issarny, V.: Service allocation in selfish mobile ad hoc networks using vickrey auction. In: Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) *EDBT 2004. LNCS*, vol. 3268, pp. 385–394. Springer, Heidelberg (2004)
18. Ramaswamy, L., Liu, L.: Free riding: A new challenge to P2P file sharing systems. In: *Proc. HICSS*, p. 220 (2003)
19. Ratner, D., Reiher, P.L., Popek, G.J., Kuenning, G.H.: Replication requirements in mobile environments. *Mobile Networks and Applications* 6(6) (2001)
20. Ratsimor, O., Finin, T., Joshi, A., Yesha, Y.: eNcentive: A framework for intelligent marketing in mobile Peer-to-Peer environments. In: *Proc. ICEC* (2003)
21. Richard, B., Nioclais, D., Chalon, D.: Clique: A transparent, peer-to-peer replicated file system. In: Chen, M.-S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A. (eds.) *MDM 2003. LNCS*, vol. 2574, Springer, Heidelberg (2003)
22. Srinivasan, V., Nuggehalli, P., Chiasserini, C.F., Rao, R.R.: Cooperation in wireless ad hoc networks. In: *Proc. INFOCOM* (2003)
23. Straub, T., Heinemann, A.: An anonymous bonus point system for mobile commerce based on word-of-mouth recommendation. In: *Proc. ACM SAC* (2004)
24. Turner, D.A., Ross, K.W.: A lightweight currency paradigm for the P2P resource market. In: *Proc. Electronic Commerce Research* (2004)
25. Wolfson, O., Xu, B., Sistla, A.P.: An economic model for resource exchange in mobile Peer-to-Peer networks. In: *Proc. SSDBM* (2004)
26. Xu, B., Wolfson, O., Rishe, N.: Benefit and pricing of spatio-temporal information in Mobile Peer-to-Peer networks. In: *Proc. HICSS-39* (2006)
27. Xue, Y., Li, B., Nahrstedt, K.: Optimal resource allocation in wireless ad hoc networks: A price-based approach. *IEEE Transactions on Mobile Computing* (2005)
28. Zhong, S., Chen, J., Yang, Y.R.: Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In: *Proc. IEEE INFOCOM* (2003)

Identifying Similar Subsequences in Data Streams

Machiko Toyoda¹, Yasushi Sakurai², and Toshikazu Ichikawa¹

¹ NTT Information Sharing Platform laboratories, NTT Corporation
9–11, Midori-cho 3-Chome Musashino-shi, Tokyo, 180–8585 Japan
{toyoda.machiko,ichikawa.toshikazu}@lab.ntt.co.jp

² NTT Communication Science Laboratories, NTT Corporation
2–4, Hikaridai, Seika-cho, Keihanna Science City, Kyoto, 619–0237 Japan
yasushi.sakurai@acm.org

Abstract. Similarity search has been studied in a domain of time series data mining, and it is an important technique in stream mining. Since sampling rates of streams are frequently different, and their time period varies in practical situations, the method which deals with time warping such as Dynamic Time Warping (DTW) is suitable for measuring similarity. However, finding pairs of similar subsequences between co-evolving sequences is difficult due to increase of the complexity because DTW is a method for detecting sequences that are similar to a given query sequence.

In this paper, we focus on the problem of finding pairs of similar subsequences and periodicity over data streams. We propose a method to detect similar subsequences in streaming fashion. Our approach for measuring similarity relies on a proposed scoring function that incrementally updates a score, which is suitable for data stream processing. We also present an efficient algorithm based on the scoring function. Our experiments on real and synthetic data demonstrate that our method detects the pairs of qualifying subsequence correctly and that it is dramatically faster than the existing method.

1 Introduction

Data streams are becoming increasingly important in several domains such as finance, sensor network environment, manufacturing, and network monitoring. The processing and mining of data streams have attracted an increasing amount of interest recently. Storing all the historical data is impossible, since the data streams are unbounded in size and arrive online continuously. However, fast responses are required.

Management of data streams has appeared as one of the most challenging extensions of database technology. Similarity search is required to continuously monitor data streams. In the static case, this problem is treated as whole matching and subsequence matching. Dynamic Time Warping (DTW) has been widely used as a similarity measure, and many works have studied whole matching and subsequence matching. Since the sampling rates of streams are frequently different and

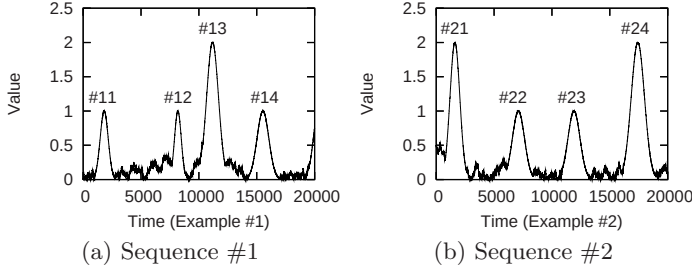


Fig. 1. Illustration of cross-similarity. These sequences have small and large spikes.

their time period varies in practical situations, the similarity measure which considers time scaling such as DTW is important in data stream processing.

In this paper, we focus on detecting similar subsequence pairs and periodicity from data streams. The problem we want to solve is as follows:

Problem 1 (Cross-similarity). Given two data streams, determine whether the pairs of similar subsequences and periodicity in the sense of DTW at any point in time.

The problem is illustrated in Fig. 1. Sequence #1 has three small spikes (#11, #12, #14) and sequence #2 has two small spikes (#22, #23). The amplitude of these spikes is almost the same, but the period is different. These sequences also have three large spikes (#13, #21, #24), not of the same period. Intuitively, cross-similarity means partial similarity between two sequences. For example, the subsequence pairs #11 and #22, #11 and #23, #13 and #21, and #13 and #24 show cross-similarity to sequences #1 and #2.

We propose a method for similarity matching on data streams, which visualizes the positions of cross-similarity. Let us consider two evolving sequences X of length n and Y of length m . As we show later, the naive solution requires $O(nm^2 + n^2m)$ time and space. Specifically, our method has the following nice characteristics:

- *High-speed processing:* our method discovers cross-similarity in $O(m + n)$ time instead of $O(nm^2 + n^2m)$ time the naive solution requires.
- *Low space consumption:* the method also requires $O(m + n)$ memory space for detecting cross-similarity.

The remainder of the paper is organized as follows. Section 2 gives related work on sequence matching and stream mining. Section 3 describes the problem definition, and Section 4 introduces our method. Section 5 reviews the results of the experiments, which clearly demonstrate the effectiveness of our method. In Section 6, we give the conclusion.

2 Related Work

The database community has been researching problems in similarity queries for time series databases for many years. Similarity matching of time-series data

was first examined by Agrawal et al [1]. They proposed an indexing mechanism called F-index, which uses the Discrete Fourier Transform (DFT) for feature extraction and indexes only on the first few DFT coefficients using R*-tree [2]. This method studied whole matching, and Faloutsos et al. [6] generalized this work for subsequence matching. Moreover, to improve the search performance, the method using Wavelet was proposed [4][14]. Indyk et al. [8] address the problem of finding representative trends from time series data. Representative trends are a period, which is the smallest distance between the first subsequence and other subsequences and a subsequence which is the smallest distance between it and all subsequences. In [5] and [18], Motif was defined as repeated subsequences. These works focus on static data sets.

In studies on data streams, Gao et al. [7] introduced a strategy of processing continuous queries with prediction and addressed the problem of finding similarity-based patterns from data streams. AWSOM [12] is one of the first streaming methods for forecasting and is intended to discover arbitrary periodicities in single time sequences. Sakurai et al. [15] applied DTW to data streams and proposed SPRING, efficiently monitors multiple numerical streams under the DTW. SPRING is useful for detecting subsequences, which are similar to a query sequence. Zhu et al. [20] focused on monitoring multiple streams in real time and proposed StatStream, which computes the pairwise correlations among all streams. SPRIT [13] addressed the problem of capturing correlations and finding hidden variables corresponding to trends in collections of co-evolving data streams. Braid [16] is a method to detect lag correlations between data streams and uses geometric probing and smoothing to approximate the exact correlation. Wang [19] combined the concepts of cross correlation and time warping and proposed TWStream which captures correlation in data streams with a much more robust similarity measure. However, none of the above methods satisfies the specifications that we listed in the introduction.

3 Problem Definition

In this section, first, we introduce Dynamic Time Warping (DTW), which is a similarity measure between sequences. Then, we define problems to study in creating our objective.

3.1 Dynamic Time Warping

DTW (Dynamic Time Warping) is one of the most useful distance measures [3][9][10][11]. Since DTW is a transformation that allows sequences to be stretched along the time axis to minimize the distance between the sequences, it can be used to calculate the distance between sequences whose lengths and/or sampling rates are different. DTW is used in case of searching for sequences that are similar to the query sequence. The DTW distance of two sequences is calculated by the dynamic programming and is the sum of tick-to-tick distances after the two sequences have been optimally warped to match each other. An

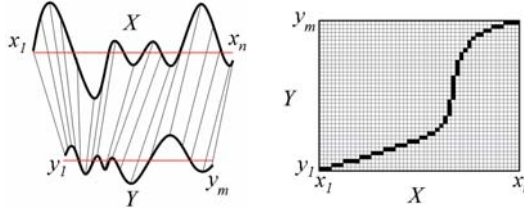


Fig. 2. Illustration of DTW. The left figure indicates the alignment of measurements by DTW. The right figure indicates the optimal warping path in the time warping matrix.

illustration of DTW is shown in Fig. 2. The left figure is the alignment of measurements by DTW for measuring the distance between two sequences. Even if the sequence length of two sequences is the same, DTW aligns properly between each element. The distance calculation uses the time warping matrix shown in the right figure. The warping path is the set of grid cells in the time warping matrix, which represents the alignment between the sequences and is shown as colored cell in the figure. Let us formally consider two sequences: $X = (x_1, x_2, \dots, x_n)$ of length n and $Y = (y_1, y_2, \dots, y_m)$ of length m . Their DTW distance $D(X, Y)$ is defined as:

$$\begin{aligned}
 D(X, Y) &= f(n, m) \\
 f(i, j) &= \|x_i - y_j\| + \min \begin{cases} f(i, j-1) \\ f(i-1, j) \\ f(i-1, j-1) \end{cases} \\
 f(0, 0) &= 0, \quad f(i, 0) = f(0, j) = \infty \\
 (i &= 1, \dots, n; \quad j = 1, \dots, m)
 \end{aligned} \tag{1}$$

where $\|x_i - y_j\| = (x_i - y_j)^2$ is the distance between two numerical values. Notice that any other choice (say, absolute difference: $\|x_i - y_j\| = |x_i - y_j|$) would be fine; our upcoming algorithms are completely independent of such choices.

3.2 Cross-Similarity

Let us consider the two sequences $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_m)$. $X[i_s : i_e]$ denotes the subsequence starting from time-tick i_s and ending at i_e , and $Y[j_s : j_e]$ denotes the subsequence starting from time-tick j_s and ending at j_e . $L_x = i_e - i_s + 1$ is the length of $X[i_s : i_e]$ and $L_y = j_e - j_s + 1$ is the length of $Y[j_s : j_e]$. L_w denotes the optimal warping path between $X[i_s : i_e]$ and $Y[j_s : j_e]$, and satisfies $L_x \leq L_w$, $L_y \leq L_w$, $L_w < L_x + L_y$. The problem we want to solve is defined as follows:

Definition 1 (Cross-similarity). *Given two sequences X and Y , a threshold ϵ , and a minimum length of similar subsequence ζ , we detect pairs of similar subsequences $X[i_s : i_e]$ and $Y[j_s : j_e]$ satisfying the following conditions:*

1. $D(X[i_s : i_e], Y[j_s : j_e]) \leq \varepsilon L_w$
2. $L_x \geq \zeta$ and $L_y \geq \zeta$

Our approach is different from the traditional time series analysis; our algorithm is designed to find partial similarity between multiple sequences in the sense of DTW. Thus, our algorithm is robust for sequences whose sampling rate is different or time period varies. Identifying partial similarity between sequences is called local alignment in the bioinformatics domain. We deal with local alignment of numeric sequences in this paper, while local alignment in the bioinformatics domain deals with symbol sequences.

An additional challenge is that we focus on data streams. Given two evolving sequences $X = (x_1, \dots, x_i, \dots, x_n, \dots)$ and $Y = (y_1, \dots, y_j, \dots, y_m, \dots)$, we want to find similar subsequence pairs between X and Y . Since data stream processing requires low memory and high speed processing, the problem we want to solve is more challenging. In the next section, we will give our approach to this problem.

4 Proposed Method

In this section, we propose a new method for cross-similarity on data streams. First, we present a method with DTW for cross-similarity, then we describe our method.

4.1 Naive Solution

For the cross-similarity problem, the most straightforward solution would be to compute the distance between all possible subsequences of X and all possible subsequences of Y . That is, the number of time warping matrices used for the computation increase significantly every new time tick. We refer to this method as *Naive*.

Let $d_{i,j}(k, l)$ be the distance of the element (k, l) in the time warping matrix of X and Y that starts from i on the x -axis and j on the y -axis. The distance of the subsequence matching between X and Y can be obtained as follows.

$$\begin{aligned}
 D(X[i_s : i_e], Y[j_s : j_e]) &= d_{i_s, j_s}(i_e - i_s + 1, j_e - j_s + 1) \\
 d_{i,j}(k, l) &= ||x_{i+k-1} - y_{j+l-1}|| + \min \begin{cases} d_{i,j}(k, l-1) \\ d_{i,j}(k-1, l) \\ d_{i,j}(k-1, l-1) \end{cases} \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 d_{i,j}(0, 0) &= 0, \quad d_{i,j}(k, 0) = d_{i,j}(0, l) = \infty \\
 (i &= 1, \dots, n; \quad k = 1, \dots, n - i + 1; \quad j = 1, \dots, m; \quad l = 1, \dots, m - j + 1)
 \end{aligned}$$

We compute the average distance d' to evaluate the similarity of $X[i_s : i_e]$ and $Y[j_s : j_e]$.

$$d' = \frac{d_{i_s, j_s}(i_e - i_s + 1, j_e - j_s + 1)}{L_w} \quad (3)$$

Algorithm

Input: a new value x_i at time-tick i

Output: qualifying subsequence pairs and distances

```

1. for  $i = 1$  to  $n$  do
2.   for  $j = 1$  to  $m$  do
3.     for  $l = 1$  to  $m - j + 1$  do
4.       Compute  $d_{i,j}(n - i + 1, l)$  by Equation (2);
5.       Compute  $d'$  by Equation (3);
6.       if  $d' \leq \varepsilon \wedge L_x \geq \zeta \wedge L_y \geq \zeta$  then
7.          $i_s := i$ ;  $i_e := n - i + 1$ ;
8.          $j_s := j$ ;  $j_e := j + l - 1$ ;
9.         Report  $(i_s, i_e, j_s, j_e, d')$ ;
10.      end if
11.    end for
12.  end for
13. end for

```

Fig. 3. Naive algorithm for detecting cross-similarity

where, again, L_w is the optimal warping path between $X[i_s : i_e]$ and $Y[j_s : j_e]$. We introduce an algorithm of the naive solution in Fig. 3. This algorithm updates the distance array of incoming x_i at time-tick i . The distance array of incoming y_j at time-tick j is also updated similarly by this algorithm. DTW is a sequence matching method for finding similar sequences to a fixed-length query sequence, thus the naive solution needs to create a new matrix for every time-tick. Since the naive solution requires $O(nm)$ matrices, $O(nm^2 + n^2m)$ values have to be updated.

4.2 Overview

As mentioned above, the naive solution requires $O(nm)$ matrices, which is infeasible to use for data stream processing. Unlike the naive solution, our solution requires a single matrix to detect cross-similarity. Our solution consists of (1) a new scoring function to evaluate cross-similarity, and (2) a streaming algorithm that uses the scoring function, each described next.

The scoring function provides a similarity measure for detecting cross-similarity. The similarity between subsequences is computed by assigning a score after the two sequences have been optimally warped to match each other. This function has a similar flavor to the original DTW function. Whereas these functions are essentially based on a dynamic programming approach, our function differs in the following two ways. First, we obtain the final matching (i.e., cross-similarity) by computing the *maximum cumulative score*, instead of computing the minimum cumulative distance that the DTW function uses. The (i, j) cell of the matrix contains the value $s(i, j)$, which is the best score to match the prefix of length i from X with the prefix of length j from Y . Second, we introduce *zero-resetting* for the scoring function. A cumulative score of the matrix, $s(i, j)$, is replaced by zero if the score is a negative value. This approach has been proposed in the bioinformatics domain (e.g., the Smith-Waterman algorithm [17]). While the search techniques in this area are used for biological sequences, which is represented by symbols, our scoring function handles numerical sequences. We replace $s(i, j)$ with zero if it is a negative value, which means the conditions of

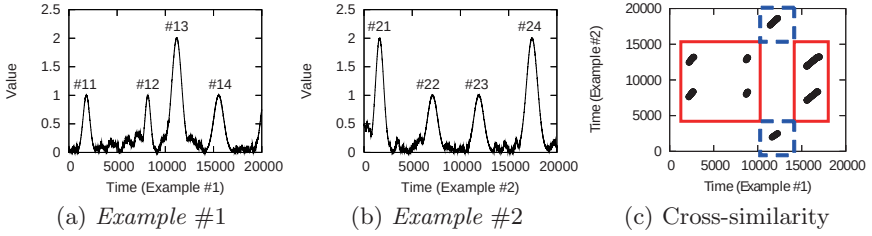


Fig. 4. Illustration of the proposed method. The left figure (a) and the center figure (b) show data sequences. The right figure (c) shows cross-similarity between Example #1 and Example #2. The result of detecting small spikes and large spikes is plotted clearly.

Definition 1 are no longer satisfied for the subsequences of X and Y ending at (i, j) . A new alignment is then created from (i, j) . An interval in which zero is continued indicates that the subsequence pairs in the interval are not qualifying at all.

Observation 1. *Once we introduce zero-resetting, we can detect qualifying subsequence pairs.*

Zero-resetting is a good first step – we can evaluate the cross-similarity between the partial sequence pairs, $X[i_s : i_e]$ and $Y[j_s : j_e]$. As the next task, we identify the position of these subsequences in the scoring matrix, especially their beginning position, i_s and j_s . In the static case, we can easily identify the beginning of the qualifying subsequence pairs by backtracking the warping path from the end of the pairs. To handle data streams efficiently, we propose to store the starting position $p(i, j)$ in the scoring matrix to keep track of the path in streaming fashion. More specifically, the (i, j) cell of the matrix contains the score (i.e., $s(i, j)$), which indicates the best score to match the prefix of length i from X and that of length j from Y (i.e., $i = 1, \dots, n; j = 1, \dots, m$). Our matrix also stores $p(i, j)$, which is the starting position corresponding to $s(i, j)$. The starting position $p(i, j)$ is described as a coordinate value. For example, consider a subsequence $X[i_s : i_e]$ and a subsequence $Y[j_s : j_e]$, the starting position $p(i_e, j_e)$ would be (i_s, j_s) . We will give an arithmetic example of a scoring matrix later (Fig. 6).

Observation 2. *The scoring matrix includes the starting position of each subsequence as well as its cumulative score. Thus, we can identify the pairs of similar subsequences in streaming fashion.*

The scoring function can tell us which subsequence pairs have a high score. Finally, we visualize results of the scoring function using a scatter plot. This scatter plot identifies the matching elements i_e and j_e of similar subsequence pairs $X[i_s : i_e]$ and $Y[j_s : j_e]$, and the cross-similarity between X and Y is reflected. The abscissa axis is an element of X and the ordinate axis is an element of Y in the figure. Specifically, when subsequence pairs $X[i_s : i_e]$ and $Y[j_s : j_e]$ are similar, position (i_e, j_e) of the figure is plotted.

We illustrate the scatter plot in Fig. 4. Data sequences are shown in Fig. 4 (a) and (b), and the scatter plot that shows the cross-similarity between these data sequences is shown in Fig. 4 (c). In the scatter plot, the square of solid lines includes detected small spikes and the square of dotted lines includes detected large spikes. Moreover, the relationship of subsequences emerges from this figure. For example, in Fig. 4 (a) and (b) the small spikes #11 and #22 look similar. For the scatter plot, the high level of similarity between #11 and #22 is shown at the lower-left corner in the solid square. We can also observe the periodicity of cross-similarity between two sequences. Six locations are regularly (horizontally and vertically) plotted in the solid square, which means small spikes, similar to #11 and #22, appear in cycles. Note that the intervals between these spikes are different, thus we can identify the time-varying periodicity.

Observation 3. *If we use the scatter plot, we can identify cross-similarity and the existence of periodicity.*

4.3 Algorithm

Given two evolving sequences $X = (x_1, \dots, x_i, \dots, x_n, \dots)$ and $Y = (y_1, \dots, y_j, \dots, y_m, \dots)$, the score $S(X[i_s : i_e], Y[j_s : j_e])$ of $X[i_s : i_e]$ and $Y[j_s : j_e]$ is computed as follows:

$$\begin{aligned}
 S(X[i_s : i_e], Y[j_s : j_e]) &= s(i_e, j_e) \\
 s(i, j) &= \max \begin{cases} 0 \\ 2\varepsilon - ||x_i - y_j|| + s_{best} \end{cases} \\
 s_{best} &= \max \begin{cases} s(i, j - 1) \\ s(i - 1, j) \\ s(i - 1, j - 1) \end{cases} \\
 s(i, 0) &= 0, \quad s(0, j) = 0
 \end{aligned} \tag{4}$$

We compute the average score s' to evaluate the similarity of $X[i_s : i_e]$ and $Y[j_s : j_e]$.

$$s' = \frac{s(i_e, j_e)}{L_w} \tag{5}$$

The starting position $p(i, j)$ of the cell (i, j) is computed as follows:

$$p(i, j) = \begin{cases} p(i, j - 1) & (s_{best} = s(i, j - 1)) \\ p(i - 1, j) & (s_{best} = s(i - 1, j)) \\ p(i - 1, j - 1) & (s_{best} = s(i - 1, j - 1)) \\ (i, j) & (s_{best} = 0) \end{cases} \tag{6}$$

We obtain the starting position of $S(X[i_s : i_e], Y[j_s : j_e])$ as:

$$(i_s, j_s) = p(i_e, j_e) \tag{7}$$

Fig. 5 shows an algorithm using the scoring function. For each incoming data point, our method incrementally updates the scores $s(i, j)$, and determines the starting positions $p(i, j)$ according to the computation of $s(i, j)$. We identify the subsequence pair $X[i_s : i_e]$ and $Y[j_s : j_e]$ by using Equations (4)(5)(6)(7). We report them as a qualifying subsequence pair when if they satisfy $s' \geq \varepsilon$, $L_x \geq \zeta$, and $L_y \geq \zeta$. In this figure, we focus on computing scores when we receive x_i at time-tick i . Note that the scores of incoming y_j at time-tick j are also computed similarly by this algorithm.

Lemma 1. *Given X and Y , then we have $s(i_e, j_e) \geq \varepsilon L_w$ if $X[i_s : i_e]$ and $Y[j_s : j_e]$ satisfy the first condition of Definition 1.*

Proof. From the first condition of Definition 1, we have

$$d_{i_s, j_s}(i_e - i_s + 1, j_e - j_s + 1) \leq \varepsilon L_w$$

then

$$2\varepsilon L_w - d_{i_s, j_s}(i_e - i_s + 1, j_e - j_s + 1) \geq \varepsilon L_w.$$

Since $X[i_s : i_e]$ and $Y[j_s : j_e]$ satisfy the condition, from Equation (4), we have

$$s(i_e, j_e) = 2\varepsilon L_w - d_{i_s, j_s}(i_e - i_s + 1, j_e - j_s + 1).$$

Thus, we obtain

$$s(i_e, j_e) \geq \varepsilon L_w$$

which completes the proof. $\square$

We show the details of our algorithm in Fig. 6. Assume two sequences $X = (5, 12, 6, 10, 6, 5, 1)$, $Y = (11, 6, 9, 4, 13, 8, 5)$, $\varepsilon = 3$, and $\zeta = 4$. The algorithm keeps $s(i, j)$ and $p(i, j)$ for the cell (i, j) , however, for simplicity, in this figure we show the average score of $s(i, j)$ (i.e., s'), instead of the original score, $s(i, j)$. The colored cells mean qualifying subsequences. For example, the cell $(6, 4)$ means that the $X[2 : 6]$ and $Y[1 : 4]$ are similar, where their average score (i.e., score per an element) is 4.6.

4.4 Complexity

Let X be an evolving sequence of length n and Y be an evolving sequence of length m .

Lemma 2. *The naive solution requires $O(nm^2 + n^2m)$ space and $O(nm^2 + n^2m)$ time.*

Proof. The naive solution has to maintain $O(nm)$ time warping matrices, and updates $O(nm^2)$ values if we receive x_i at time-tick i , and $O(n^2m)$ values if we receive y_j at time-tick j to identify qualifying subsequences. Therefore, it requires $O(nm^2 + n^2m)$ time. Since the naive solution maintains two arrays of m numbers and that of n numbers for each matrix, overall, it needs $O(nm^2 + n^2m)$ space. $\square$

Algorithm

Input: a new value x_i at time-tick i

Output: qualifying subsequence pairs and distances

1. **for** $j = 1$ to m **do**
2. Compute $s(n, j)$ by Equation (4);
3. Compute s' by Equation (5);
4. Compute $p(n, j)$ by Equation (6);
5. **if** $s' \geq \varepsilon \wedge L_x \geq \zeta \wedge L_y \geq \zeta$ **then**
6. $(i_s, j_s) := p(n, j); \quad i_e := n; \quad j_e := j;$
7. Report (i_s, i_e, j_s, j_e, s') ;
8. **end if**
9. **end for**

7	5	6 (1, 7)	0 (1, 7)	4.25 (1, 4)	0 (1, 4)	3.5 (1, 4)	3.86 (1, 4)	2.13 (1, 4)
6	8	1 (1, 6)	0 (1, 4)	4 (1, 4)	3.5 (1, 4)	3.2 (1, 4)	2.17 (1, 4)	0 (1, 4)
5	13	0 (1, 4)	5 (1, 4)	0 (1, 4)	0.6 (2, 1)	0 (2, 1)	0 (2, 1)	0 (2, 1)
4	4	5 (1, 2)	0 (1, 4)	1.5 (2, 1)	0 (2, 1)	4.5 (2, 1)	4.6 (2, 1)	3.33 (2, 1)
3	9	0 (1, 2)	1 (1, 2)	2.67 (2, 1)	4 (2, 1)	2.6 (2, 1)	1.5 (4, 1)	0 (4, 1)
2	6	5 (1, 2)	0 (1, 2)	5.5 (2, 1)	0.33 (2, 1)	5.5 (4, 1)	5.33 (4, 1)	0 (4, 1)
1	11	0 (1, 1)	5 (2, 1)	0 (2, 1)	5 (4, 1)	0 (4, 1)	0 (6, 1)	0 (7, 1)
j	i	5	12	6	10	6	5	1
	i	1	2	3	4	5	6	7

Fig. 5. Proposed algorithm for detecting cross-similarity

Fig. 6. Proposed scoring function. The upper number indicates the score per an element. The number in parentheses indicates the starting position.

Table 1. Detail of data sets and experimental parameter settings

Data sets	Seq#1 length	Seq#2 length	Threshold	ζ
<i>Sines</i>	25000	25000	5.5e-5	500
<i>Spikes</i>	28000	28000	1.0e-6	1100
<i>Temperature</i>	30411	26151	1.5e-2	1800
<i>Mocap</i>	8400	4560	5.8e-1	240

Lemma 3. *Our method requires $O(m + n)$ space and $O(m + n)$ time.*

Proof. Our method maintains a single matrix, and updates $O(m)$ numbers if we receive x_i at time-tick i , and $O(n)$ numbers if we receive y_j at time-tick j to identify qualifying subsequences. Thus, it requires $O(m + n)$ time and $O(m + n)$ space. $\square$

5 Experiments

To evaluate the effectiveness of our method, we performed experiments on real and synthetic datasets. We compared our method with the naive solution and SPRING. SPRING is proposed by Sakurai et al. [15] and is a method based on DTW for finding similar subsequences to a fixed-length query sequence from data streams. SPRING is not intended to be used for finding cross-similarity, but we can apply this method to solve the problem we mentioned in Section 1. Specifically, an algorithm with SPRING for cross-similarity needs $O(m^2 + nm)$ space and $O(m^2 + nm)$ time, since SPRING needs $O(m)$ matrices, and updates $O(m^2)$ values if we receive x_i at time-tick i , and $O(nm)$ values if we receive y_j at time-tick j to identify qualifying subsequence pairs.

Our experiments were conducted on an 3 GHz Intel Pentium4 with 2 GB of memory, running Linux. The experiments were designed to answer the following questions:

1. How successful is the proposed method in capturing cross-similarity?
2. How does it scale with the sequence length in terms of the computation time?

5.1 Detecting Cross-Similarity

We used two real data sets and two synthetic data sets for our experiment. The details of each data set and the settings of experiments are shown in Table 1.

Sines: *Sines* is a synthetic data set which consists of discontinuous sine waves with white noise (See Fig. 9 (a) and (b)). We varied the period of each sine wave in the sequence. The intervals between these sine waves are also different. Our method can perfectly identify all sine waves and the time-varying periodicity as shown in Fig. 9 (c). In this figure, the difference in the period of each sine wave appears as the difference in the slope.

Spikes: *Spikes*, which is a synthetic data set shown in Fig. 8 (a) and (b), consists of large and small spikes. The data of different-length intervals between spikes were generated by a random walk function. The period of each spike is also different. In Fig. 8 (c) and (d), we can confirm that our method can detect large spikes and small spikes. The difference in the period of each spike appears as the difference in plot length; the wide spikes indicate the long plot length and the narrow spikes indicate the short plot length.

Temperature: *Temperature* is measured by a temperature sensor about for 10 days and it's measurement value are Celsius (See Fig. 7 (a) and (b)). This sensor sends a measurement value at one minute intervals. This data set lacks measurement values at a lot of time. We set ζ to 1800, corresponds to about half a day.

Temperature #1 and *Temperature #2* have the consecutive change of temperature fluctuating from about 18 degrees centigrade to 32 degrees centigrade in order to break in the weather. In spite of lack of measurement values, our method is successful to detect these patterns (See Fig. 7 (c)).

Mocap: As an extension to multiple streams, we used the *Mocap* real data set. *Mocap* is a real data set created by recording motion information from a human actor while the actor is performing an action (e.g. walking, running, kicking). Special markers are placed on the joints of the actor (e.g., knees, elbows), and their x-, y- and z-velocities are recorded at about 120 Hz. The data were from the CMU motion capture database ¹. We selected the data of limbs from the original data and used them as 8-dimensional data. These data include walking

¹ <http://mocap.cs.cmu.edu/>

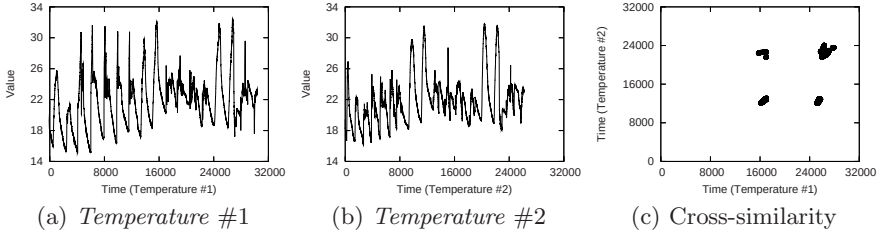


Fig. 7. Discovery of cross-similarity using *Temperature*

Table 2. Detail of *Mocap* data sets. Lengths of each data sequence are different.

(a) *Mocap* #1

Sec.	Time-tick	Motions
0 - 8	0 - 960	walking
8 - 16	960 - 1920	running
16 - 20	1920 - 2400	jumping
20 - 29	2400 - 3480	walking
29 - 38	3480 - 4560	kicking
38 - 44	4560 - 5280	left-leg jumping
45 - 50	5400 - 6000	right-leg jumping
52 - 59	6240 - 7080	stretching
59 - 70	7080 - 8400	walking

(b) *Mocap* #2

Sec.	Time-tick	Motions
0 - 3	0 - 360	walking
4 - 9	480 - 1080	jumping
10 - 14	1200 - 1680	walking
16 - 20	1920 - 2400	punches
20 - 25	2400 - 3000	walking
26 - 33	3120 - 3960	kicking
33 - 38	3960 - 4560	punches

motion and the length of each data is different. The detail of motion is shown in Table 2. We set ζ to 240, corresponds to about two seconds.

The result in Fig. 10 demonstrates our method can capture walking motion perfectly and the plotted intervals are matched to the interval between walking motions shown in Table 2.

5.2 Performance

We did experiments to evaluate the efficiency and to verify the complexity of our method, which is discussed in Section 4.4.

Our method, the naive implementation, and SPRING are compared in terms of computation time for varying the sequence length of X and Y . We used *Sines* for this experiment, which allowed us to control the sequence length. Time is the average processing time needed to update the scoring matrix for each time-tick and to detect the qualifying subsequence pairs.

As we expected, our method identifies the qualifying subsequence pairs much faster than naive and SPRING implementation as shown in Fig. 11. The trend shown in the figure agrees with our theoretical discussion in Section 4.4. Our method achieves a dramatic reduction in computation time.

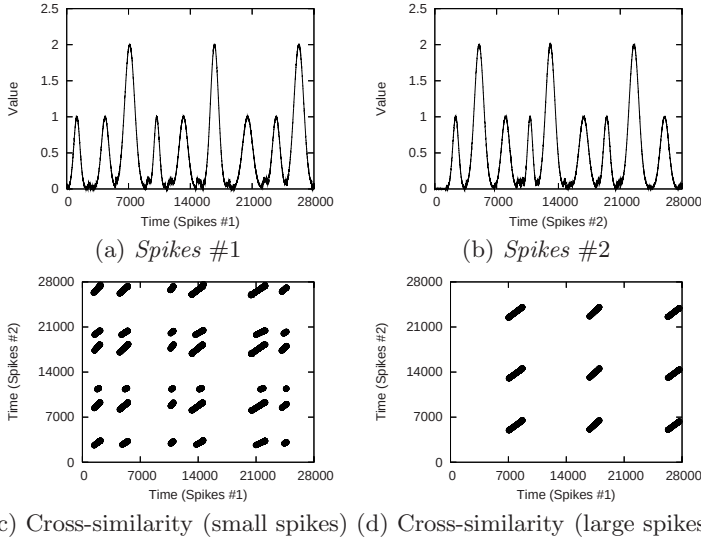


Fig. 8. Discovery of cross-similarity using *Spikes*. Upper figures (a) and (b) show data sequences. Lower figures (c) and (d) indicate results of cross-similarity.

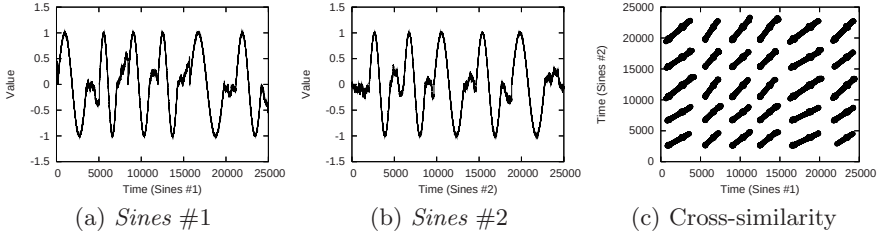


Fig. 9. Discovery of cross-similarity using *Sines*

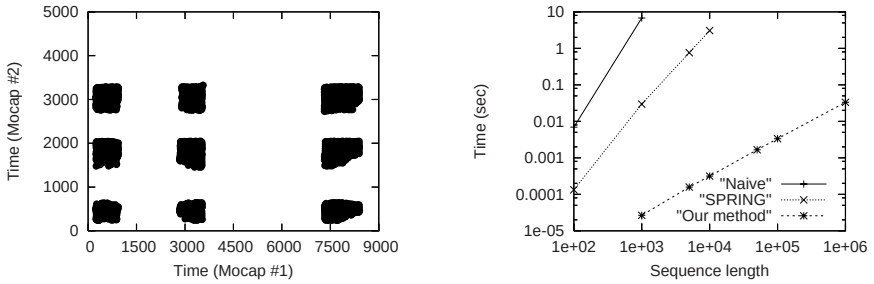


Fig. 10. Discovery of cross-similarity using **Fig. 11.** Wall clock time for cross-similarity as a function of sequence length

6 Conclusions

We introduced the problems of cross-similarity over data streams and proposed a new method to address this problem. Our method is based on two ideas. With the scoring function, similar subsequence pairs are detected effectively from data streams. With the streaming algorithm, cross-similarity are visualized and periodicity can be discovered.

Our experiments on real and synthetic data sets demonstrated that our method works as expected, detecting the qualifying subsequence pairs effectively. In conclusion, our method have the following characteristics: (1) In contrast to the naive solution, our method improves the performance greatly and can be processed at high speed. (2) Our method requires only a single matrix to detect cross-similarity and it consumes a small amount of resources.

References

1. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient Similarity Search in Sequence Database. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)
2. Beckmann, N., Keriegel, H.P., Schneider, R., Segger, B.: The r^* -tree: An efficient and robust access method for points and the rectangles. In: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 1990, pp. 322–331 (1990)
3. Berndt, D.J., Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series. In: AAAI 1994 Workshop on Knowledge Discovery in Databases (KDD Workshop 1994), Seattle, Washington, USA, July 1994, pp. 359–370. AAAI Press, Menlo Park (1994)
4. Chan, K., Fu, A.W.-C.: Efficient Time Series Matching by Wavelets. In: Proceedings of the 15th International Conference on Data Engineering (ICDE 1999), Sydney, Australia, May 1999, pp. 126–133 (1999)
5. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003), Washington, DC, USA, August 2003, pp. 493–498 (2003)
6. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast Subsequence Matching in Time-Series Database. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 1994, pp. 419–429 (1994)
7. Gao, L., Wang, X.S.: Continually Evaluating Similarity-Based Pattern Queries on a Streaming Time Series. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 2002, pp. 370–381 (2002)
8. Indyk, P., Koudas, N., Muthukrishnam, S.: Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. In: Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000), Cairo, Egypt, September 2000, pp. 363–372 (2000)
9. Jang, J.-S.R., Lee, H.-R.: Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input. In: Proceedings of the ninth ACM International Conference on Multimedia, Ottawa, Canada, September-October 2001, pp. 401–410 (2001)

10. Kawasaki, H., Yatabe, T., Ikeuchi, K., Sakauchi, M.: Automatic Modeling of a 3D City Map from Real-World Video. In: Proceedings of the seventh ACM International Conference on Multimedia (Part 1), Orlando, Florida, USA, October–November 1999, pp. 11–18 (1999)
11. Mount, D.W.: Bioinformatics: Sequence and Genome Analysis. Cold Spring Harbor, New York (2000)
12. Papadimitriou, S., Brockwell, A., Faloutsos, C.: Adaptive, Hands-Off Stream Mining. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) VLDB 2003. LNCS, vol. 2944, pp. 560–571. Springer, Heidelberg (2004)
13. Papadimitriou, S., Sun, J., Faloutsos, C.: Streaming Pattern Discovery in Multiple Time-Series. In: Proceedings of the 31th International Conference on Very Large Data Bases (VLDB 2005), Trondheim, Norway, August–September 2005, pp. 697–708 (2005)
14. Popivanov, I., Miller, R.J.: Similarity Search Over Time-Series Data Using Wavelets. In: Proceedings of the 18th International Conference on Data Engineering (ICDE 2002), San Jose, CA, USA, February–March, 2002, pp. 212–221 (2002)
15. Sakurai, Y., Faloutsos, C., Yamamuro, M.: Stream Monitoring under the Time Warping Distance. In: Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey, April 2007, pp. 1046–1055 (2007)
16. Sakurai, Y., Papadimitriou, S.: Braid: Stream Mining through Group Lag Correlations. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, June 2005, pp. 599–610 (2005)
17. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147, 195–197 (1981)
18. Tanaka, Y., Uehara, K.: Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS, vol. 2734, pp. 252–265. Springer, Heidelberg (2003)
19. Wang, T.: TWStream: Finding Correlated Data Streams under Time Warping. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 213–225. Springer, Heidelberg (2006)
20. Zhu, Y., Shasha, D.: StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In: Bressan, S., Chaudhri, A.B., Li Lee, M., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 358–369. Springer, Heidelberg (2003)

A Tree-Based Approach for Event Prediction Using Episode Rules over Event Streams

Chung-Wen Cho¹, Ying Zheng², Yi-Hung Wu³, and Arbee L.P. Chen⁴

¹ Department of Computer Science, National Tsing Hua University, Taiwan, R.O.C.

² Department of Computer Science, Duke University, U.S.A

³ Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan, R.O.C.

⁴ Department of Computer Science, National Chengchi University, Taiwan, R.O.C.
alpchen@cs.nccu.edu.tw

Abstract. Event prediction over event streams is an important problem with broad applications. For this problem, rules with predicate events and consequent events are given, and then current events are matched with the predicate events to predict future events. Over the event stream, some matches of predicate events may trigger duplicate predictions, and an effective scheme is proposed to avoid such redundancies. Based on the scheme, we propose a novel approach CBS-Tree to efficiently match the predicate events over event streams. The CBS-Tree approach maintains the recently arrived events as a tree structure, and an efficient algorithm is proposed for the matching of predicate events on the tree structure, which avoids exhaustive scans of the arrived events. By running a series of experiments, we show that our approach is more efficient than the previous work for most cases.

Keywords: Continuous query, episode rules, minimal occurrence, event stream, prediction.

1 Introduction

In many applications, events such as stock fluctuations in the stock market [11], alarms in telecommunication networks [9], and road conditions in traffic control [8] often arrive rapidly as a continuous stream. Particular events on this kind of streams are important to applications but may expire in a short period of time. Therefore, an effective mechanism that predicts the incoming events based on the past events is helpful for the quick responses to particular events. An example is the forecast of traffic jams according to the nearby road conditions. In daily rush hours, the avoidance of all traffic jams is essential to traffic control. In general, there are two main steps to enable effective prediction over event streams. The first step is to obtain the causal relationships among events, which can be either derived from the past events or given by users. The second step is to keep monitoring the incoming events and utilize these relationships for online prediction. In this paper, we consider the causal

relationships in the form of rules and aim at designing an efficient algorithm for the second step. In the following, we first illustrate the form of rules in our consideration by a traffic control example.

Fig. 1 shows a map with four checkpoints denoted as a, b, c, and d. Suppose each checkpoint is associated with a sensor that can periodically report its traffic condition, including the flow and occupancy [8]. The flow is the number of cars passing by a sensor per minute. The occupancy is a ratio of one-minute interval in which the detection zone is occupied by a car. If the sensor of a road reports low flow and high occupancy, the traffic of this road can be regarded as “congested.” An example rule, represented in the form of $\alpha \Rightarrow \beta$, is shown in Fig. 2, where α is called the predicate and β the consequent. The predicate stands for a set of events that constitute a directed acyclic graph. Each vertex in the graph represents an event, while each edge from vertex u to vertex v indicates that the event u must occur before the event v . For instance, the predicate in Fig. 2 makes it a condition that the two events sensing high flow and low occupancy on segments a and b respectively precede the event sensing high flow and high occupancy on segment c. Note that this predicate allows the events on a and b to occur in any order. The consequent is a single event and the rule means that the consequent will appear in the near future after the predicate shows up.

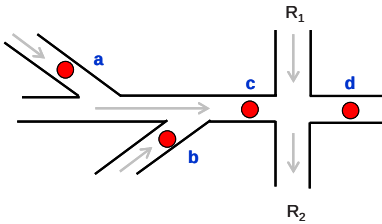


Fig. 1. A roadmap

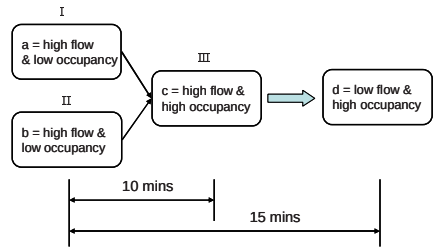


Fig. 2. An episode rule

Furthermore, two time bounds are assigned to the rule and its predicate, respectively. The time bound on the predicate, called *predicate window*, is a temporal constraint on the occurrences of all the events in the predicate, while the one on the entire rule, called *rule window*, corresponds to the temporal constraint on the occurrences of all the events in the entire rule. Consider the rule in Fig. 2 again. If all the events in the predicate occur within 10 minutes in accordance with the temporal order specified in the predicate, the event in the consequent will appear with certain probability in no more than 15 minutes. Specifically, segment d will become low flow and high occupancy (i.e., traffic jam occurs on d). Therefore, once we find a match for the predicate over the traffic event stream, the traffic control system can make a proper response accordingly, e.g., limiting the traffic flow from R_1 to d or guiding the traffic flow on c to R_2 , before the traffic jam on d occurs. We call this kind of rules the *episode rule* [10]. The problem we address in this paper is thus formulated below:

Given a set of episode rules, keep monitoring each of them over the stream of events to detect whether all the events in the predicate appear in the order as specified in the predicate of the rule and satisfy the time bound, and return as soon as a match is found the event in the consequent together with the time interval it will appear.

In this paper, we assume that the system receives only one event at a time. Moreover, we consider an event as a set of values from separate attributes and do not explicitly distinguish them by the attributes. Continue the traffic control example in Fig. 2. Let W, X, Y, and Z denote the traffic events “a = high flow & low occupancy”, “b = high flow & low occupancy”, “c = high flow & high occupancy”, and “d = low flow & high occupancy”, respectively. An example stream of traffic events from timestamp 8:00 to 8:07 are depicted in Fig. 3, where the events W, X, and Y within the time interval [8:02, 8:05] match the predicate. Thus, it would be reported that event Z will arrive within the time interval (8:02, 8:17). We call the occurrences of the events corresponding to the matches of the predicate as *predicate occurrences*. Let the pair (e, t) denote the event e that occurs at time t. In this example, we say that $\langle (W, 8:02), (X, 8:03), (Y, 8:05) \rangle$ is a predicate occurrence of the episode rule in Fig. 2.

For an episode rule, there can be more than one predicate occurrence in a time period. For example, given the episode rule in Fig. 2, in Fig. 3 there are three predicate occurrences, $O_1 = \langle (X, 8:00), (W, 8:02), (Y, 8:05) \rangle$, $O_2 = \langle (W, 8:02), (X, 8:03), (Y, 8:05) \rangle$, and $O_3 = \langle (W, 8:02), (X, 8:03), (Y, 8:07) \rangle$. Note that only the time interval of O_2 ([8:02, 8:05]) does not enclose the time interval of any other. We call this kind of occurrences the *minimal occurrence* [10]. The events of a minimal occurrence span a shorter period. From O_2 and the rule window, it can be predicted that event Z will occur within the time interval (8:05, 8:17). We call this time interval the *predicted interval* of O_2 . It can be seen that the predicted intervals of O_1 and O_3 are (8:05, 8:15) and (8:07, 8:17), respectively. Since both the two predicted intervals are enclosed in the predicted interval of O_2 , O_1 and O_3 provide less but redundant information for prediction than O_2 . Therefore, one of our goals in this paper is to discover all the minimal occurrences for the predicate of every episode rule but ignore the other predicate occurrences. Notice that, many real time applications such as intrusion detection systems concern when the consequent will likely occur, rather than when it will most likely occur. Our work aims at promptly reporting the time interval in which the consequent will occur for these applications.

The burst and endlessness characteristic of data streams make it impossible to retrieve every occurrence and then verify whether it should be reported. Thus, we need an efficient way to ensure that no redundant information is reported (i.e., only minimal occurrences are reported). Cho et al. [2] propose a method named *ToFel*

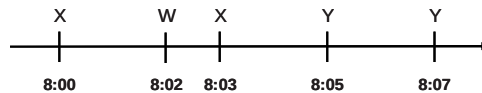


Fig. 3. An example stream of traffic events

(Topological Forward Retrieval) to match the minimal occurrences from the stream. ToFel takes advantage of the topological characteristic in the predicate to find the minimal occurrences by incrementally maintaining parts of the predicate occurrences, and thus avoids scanning the stream backward. It constructs one event filter with a queue for each predicate to be matched. The filter continuously monitors the newly arrived events and only keeps those that are likely to be parts of the minimal occurrences. However, ToFel may suffer from the plenty of *false alarms* over the stream, i.e., partial matches that do not finally lead to a minimal occurrence. Moreover, since ToFel needs to keep the arrived events for the occurrences of a predicate, the memory required by ToFel is proportional to the given episode rules. In the following, we briefly introduce the other related works and then discuss the difference between them and ours.

Over the past few decades, significant projects such as Ode [7] and SAMOS [6] have developed individual active database management systems. One of the core issues in these projects is to propose the algebra with a rich number of event operators to support various types of queries (*composite events*). For the content-based subscription systems [1, 4], Aguilera et al. [1] index subscriptions as a tree structure where a path from the root to a leaf corresponds to a set of subscriptions whose sets of attribute-value-operator triples are identical. An event will be evaluated by traversing the tree from the root to some leaves. If a leaf is finally reached, the event will fit the subscriptions corresponding to the retrieved path. Demers et al. [4] allow users to specify a subscription whose match is formed from a set of events. Their approach can deal with multiple descriptions simultaneously through the proposed indexing techniques on the automata. Wu et al. [13] proposed an event sequence query model (SASE) with attribute value comparison for RFID data streams. A query plan based on NFA with a stack structure is proposed to retrieve the query occurrences from the stream. In the RFID-based data stream management systems proposed in [5, 12], query models and system architectures were discussed but no detailed implementations were provided.

The difference between these works and ours mainly lies in our concept of the minimum occurrence and the corresponding matching process over event streams. The algorithms based on automata [4, 7, 13] or petri nets [6] cannot avoid redundant predictions. Some of the complex event processors [4, 13] cannot cope with graph-based queries. The query format considered in [1] is set-based while ours is graph-based, and their answer must be matched at a timestamp while ours is matched over a window of timestamps. As a result, all of them are not adaptable to handle episode-based queries.

In this paper, we propose a novel approach called *CBS-Tree* (Circular Binary Search Tree). CBS-Tree maintains the recently arrived events in a tree structure, and an efficient algorithm is proposed for the retrieval of the predicate occurrences upon the tree structure, which avoids scanning the stream repeatedly. The memory requirement in CBS-Tree is only affected by the maximum of all the predicate windows and thus is independent of the amount of episode rules. Moreover, since the occurrence retrieval on CBS-Tree is triggered by the sinks of predicates (the vertices with no edge out of them), the proposed approach does not need to maintain the partial match for every predicate occurrence. Therefore, CBS-Tree outperforms ToFel in processing time when the stream has plenty of false alarms.

The remainder of the paper is organized as follows. Section 2 presents the problem statements including the preliminary and our problem. Section 3 introduces the CBS-Tree method. Due to the limitation of space, all the formal proofs and pseudo codes are omitted and can be found in [3]. The experimental results are discussed in Section 4. Finally, we give a conclusion and the future directions in Section 5.

2 Problem Formulation

In this section, we first define several terms related to our problem, and then the basic concepts used in our approach are presented. An *episode* is a directed acyclic graph g , where each vertex stands for an event, and all vertices correspond to distinct events. The event corresponding to vertex v is denoted as $\mathcal{E}(v)$. Each directed edge (u, v) in g indicates that $\mathcal{E}(u)$ must precede $\mathcal{E}(v)$. We call this precedence the *temporal order* between vertex u and vertex v and denote it as $u \prec v$. Note that, there can be more than one sources (the vertices with no edge into them) and sinks in an episode.

An *event stream* can be represented as $\hat{S} = \langle (e_1, t_1), (e_2, t_2), \dots (e_n, t_n) \dots \rangle$, where $t_1 < t_2 < \dots < t_n \dots$. Given an *event sequence* $S' = \langle (e'_1, t'_1), (e'_2, t'_2) \dots, (e'_m, t'_m) \rangle$, where $t'_1 < t'_2 < \dots < t'_m$, we define the *time interval* or *interval* of S' as $[t'_1, t'_m]$, and the *start time* and *end time* of S' as t'_1 and t'_m , respectively. Moreover, we say S' occurs at timestamp t'_m . S' is a *subsequence* of $\hat{S}$ if there exist m integers $i_1, i_2, \dots, i_m$, such that $1 \leq i_1 < i_2 < \dots < i_m$ and $\forall 1 \leq j \leq m, e'_j = e_{i_j}$ and $t'_j = t_{i_j}$. Given an episode α with a time bound ω_α , the *episode occurrence* or *occurrence* of α over $\hat{S}$ is an event sequence S with time interval $[t_s, t_e]$ satisfying that: (1) S is a subsequence of $\hat{S}$; (2) the events corresponding to the vertices in α have an one-to-one mapping to the events in S ; (3) the order in which the events occur in S is consistent with the temporal order $\prec$ specified in α ; and (4) $t_e - t_s \leq \omega_\alpha$. Under the given one-to-one mapping, each vertex v in α corresponds to a unique instance of $\mathcal{E}(v)$ in S , which is called the *mapping instance* of v in S . Given any two occurrences of α , O_1 and O_2 , with intervals $[t_{s1}, t_{e1}]$ and $[t_{s2}, t_{e2}]$, respectively, if $t_{s1} \leq t_{s2} < t_{e2} \leq t_{e1}$ and $t_{e2} - t_{s2} < t_{e1} - t_{s1}$, we say that $[t_{s2}, t_{e2}]$ is *enclosed* by $[t_{s1}, t_{e1}]$. For ease of presentation, we also say that O_2 is *enclosed* by O_1 . Note that, the second condition excludes two occurrences in the same time interval from consideration. A *minimal occurrence* of α is an (episode) occurrence that is not enclosed by any other (episode) occurrence of α . Note that, with this definition, there can be multiple minimal occurrences in the same time interval.

An *episode rule* ρ is a 5-tuple $(\alpha, \beta, \omega_\alpha, \omega_\rho, \chi)$. Here, α is the episode representing the predicate of ρ and β is the event representing the *consequent* of ρ . ω_α and ω_ρ stand for the predicate window and rule window, respectively. The interpretation of ρ is that if α has an occurrence O with interval $[t_s, t_e]$ ($t_e - t_s \leq \omega_\alpha$), β will occur during interval $(t_e, t_s + \omega_\rho)$ with probability χ . We call the interval $(t_e, t_s + \omega_\rho)$ the *predicted interval* of the occurrence O .

The properties of minimal occurrences were discussed in [2, 3]. Due to the lack of space, in the following, we just give an outline for minimal occurrences.

Property 1. If the occurrence of α O_1 with interval $[t_{s1}, t_{e1}]$ is not a minimal occurrence, there must be a minimal occurrence of α O_2 with interval $[t_{s2}, t_{e2}]$ such that the predicted interval of O_1 is enclosed by that of O_2 .

By Property 1, any non-minimal occurrence can be ignored. Therefore, we conclude that only the minimal occurrences of α should be matched for the prediction of β . We summarize it in Lemma 1.

Lemma 1. Given the episode rule ρ , the union of predicted intervals for β derived from all the occurrences of α is exactly the same as the union of predicted intervals derived from only the minimal occurrences of α .

Intuitively, we may infer that all the minimal occurrences should be matched if each of them has a unique end time. However, it is possible to have multiple minimal occurrences in the same time interval. Therefore, we need a unique way to select one from a set of minimal occurrences whose time intervals are all identical. We introduce the concept of *latest occurrence*, as defined below, which represents a unique form for one kind of minimal occurrences. Based on the concept, only the minimal occurrences in this form will be matched.

Definition 1. *Latest instance of an event.* Given a timestamp t and an event instance (X, t_1) , where $t_1 \leq t$, if there does not exist another event instance (X, t_2) such that $t_1 < t_2 \leq t$, (X, t_1) is called the *latest instance* of X to t .

Definition 2. *Latest occurrence of an episode.* Given a timestamp t , the occurrence $\langle (\varepsilon(v_1), t_1), (\varepsilon(v_2), t_2), \dots, (\varepsilon(v_n), t_n) \rangle$ of α , where $t_n \leq t$, is called the *latest occurrence* of α to t if both of the following conditions hold:

(a) For every sink v_j of α , $(\varepsilon(v_j), t_j)$ is the latest instance of $\varepsilon(v_j)$ to t ; (b) For every non-sink vertex v_k of α , if $v_{k1}, v_{k2}, \dots, v_{km}$ are the direct successors of v_k , where $1 \leq k < n$ and $t_k < t_{k1} < t_{k2} < \dots < t_{km} \leq t_n$, $(\varepsilon(v_k), t_k)$ is the latest instance of $\varepsilon(v_k)$ to t_{k1} .

Let Σ_t denote the set of all occurrences of α that occur at time t . By Definition 2, the latest occurrence of α to t , denoted as LO_t , is included in Σ_t . If Σ_t has one or more minimal occurrences, LO_t must be one of them. However, Σ_t is not always in this case and LO_t is not necessarily a minimal occurrence. We call the latest occurrence that is also a minimal occurrence the *min-latest occurrence*. The latest occurrences that are not minimal can be identified by checking whether they contain any of special event instances in the min-latest occurrence previously obtained, which are defined as follows.

Definition 3. *Rejected event instance set.* Given the episode α and the min-latest occurrence $\langle (\varepsilon(v_1), t_1), (\varepsilon(v_2), t_2), \dots, (\varepsilon(v_n), t_n) \rangle$ previously obtained, the *rejected event instance set* is defined recursively as follows:

(a) $(\varepsilon(v_1), t_1)$ is a rejected event instance; (b) Given that $(\varepsilon(v_i), t_i)$ is a rejected event instance, for every direct successors v_j of v_i , where $1 \leq i < j \leq n$, $(\varepsilon(v_j), t_j)$ is also a rejected event instance if there is no occurrence of $\varepsilon(v_i)$ in the time interval (t_i, t_j) .

Given the rejected event instance set, the latest occurrence that contains a rejected event instance must have the same start time as the min-latest occurrence previously obtained. On the other hand, a latest occurrence that does not contain a

reject event instance must be a minimal occurrence. In the following, we formulate these findings as Lemma 2 and Lemma 3, respectively.

Lemma 2. Given a min-latest occurrence $O = \langle (\epsilon(v_1), t_1), (\epsilon(v_2), t_2), \dots, (\epsilon(v_n), t_n) \rangle$, the latest occurrence $O' = \langle (\epsilon(v'_1), t'_1), (\epsilon(v'_2), t'_2), \dots, (\epsilon(v'_n), t'_n) \rangle$, where $t_n < t'_n$, is not a minimal occurrence if it contains a rejected event instance deduced from O .

Lemma 3. A latest occurrence of α is a minimal occurrence if it does not contain any rejected event instance.

To conclude this section, only the latest occurrences that do not contain any rejected event instance are the min-latest occurrences we are looking for.

3 The CBS-Tree (Circular Binary Search Tree) Approach

In this section, we describe our tree-based method. Subsection 3.1 introduces the proposed tree structure, CBS-Tree, and the tree maintenance. In subsection 3.2, we present how to use CBS-Tree to accelerate the min-latest occurrence retrieval for a given predicate episode.

3.1 CBS-Tree

The CBS-Tree, which is an index of recently arrived events over the stream, can achieve a fast retrieval of the min-latest occurrence for a given episode. Let L be the number of event instances recorded in the CBS-tree. We design CBS-Tree as a complete binary tree with L leaves, where each leaf corresponds to an event instance and each node is numbered from left to right in bottom-up fashion. As a result, the most recent L event instances are kept in the leaves, which are numbered from 1 to L . Moreover, the event instance with timestamp t is assigned to the leaf node numbered $((t - 1) \bmod L) + 1$. In addition, each internal node is associated with two sets, the *event set* R_E and the *timestamp set* R_T , to respectively keep the events and timestamps of all the event instances assigned to its descendants.

Let the event stream $\hat{S}$ be $\langle (A, 1), (B, 2), (A, 3), (C, 4), (D, 5), (C, 6), (B, 7), (D, 8), \dots \rangle$. For example, Fig. 4 shows the CBS-Tree with 7 leaves, corresponding to the event instances in time interval $[1, 7]$ over $\hat{S}$. Each node in the tree is numbered as mentioned and the sets R_E and R_T of each internal node are depicted above the node. In the leaf layer, node 1 stands for the event instance $(A, 1)$, node 2 represents the event instance $(B, 2)$, and so forth. For the internal node 8, since it is the parent of nodes 1 and 2, its event set is $\{A, B\}$ and timestamp set is $\{1, 2\}$.

Tree Maintenance. Initially, since no event has arrived, the event set and timestamp set of each node are empty. Once an event arrives, the maintenance operation begins at the leaf node that corresponds to the timestamp of the arrived event, and then iteratively updates its ancestors in bottom-up fashion. For example, in Fig. 5, we show the final state of CBS-Tree after the event instance $(D, 8)$ arrives. Since we only need to record the events whose arrival times range from timestamps 2 to 8, node 1 corresponding to the event instance $(A, 1)$ now corresponds to the event

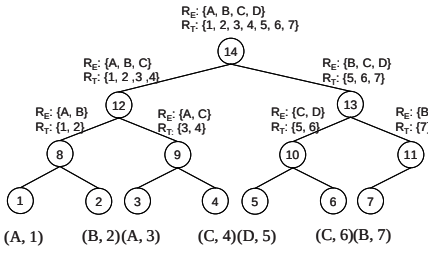


Fig. 4. The CBS-Tree

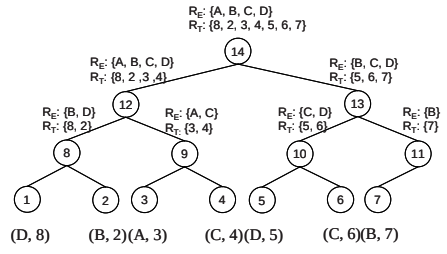


Fig. 5. The CBS-Tree after updating

instance (D, 8) instead. Therefore, we update the event set and timestamp set of node 8 to $\{B, D\}$ and $\{8, 2\}$, respectively. We then update node 12 and node 14 respectively. In this way, the tree maintenance at each timestamp requires traversing only one path from a leaf to the root. Thus, the time complexity of tree maintenance is $O(\log L)$.

3.2 CBS-Tree-Based Retrieval

We now present how to utilize the CBS-Tree for fast retrieval of the min-latest occurrences. According to Definition 2, we retrieve all the event instances of a latest occurrence in the inverse topological order of their corresponding vertices in the episode. Take Fig. 2 as an example. We first find the mapping instances of vertex (III), then vertices (II) and (I). During the retrieval, if one of the mapping instances is a rejected event instance, we stop the retrieval and can be sure that there is no min-latest occurrence.

In Algorithm 1, we present the algorithm named *CBS-Tree Retrieval* (shortly *CBR*) for finding the min-latest occurrence of the episode α occurred within time interval $[lt, rt]$, where rt is the current time and $lt = rt - \omega_\alpha + 1$. In the algorithm, we first determine whether the newly arrived event can be the mapping instance of a sink in α . If this condition holds, we then retrieve the latest occurrence of α . Otherwise, there is no new min-latest occurrence of α with end time rt . To retrieve the min-latest occurrence, as depicted in lines 2-4 of Algorithm 1, we iteratively retrieve the latest instance mapped to every vertex in α by CBS-Tree, each of whose direct successors has been associated with a mapping instance in $[lt, rt]$. In this way, if there does not exist a mapping instance for a vertex, we say that no min-latest occurrence of α with end time rt exists (line 5). The procedure *FindLatestEventOccurrence* in line 4 returns the occurring time of the latest instance for a given event and a time interval. The left-endpoint of the specified time interval is always set to lt , and the right-endpoint is computed from the mapping instances of the direct successors of v by Definition 2 (line 3). *FindLatestEventOccurrence* will return -1 to indicate that the desired latest instance is not found or is a rejected event instance. Detailed descriptions of *FindLatestEventOccurrence* will be presented later. In line 6, once a new min-latest occurrence is found, the corresponding rejected event instance set will be derived.

Algorithm 1 *CBS-Tree Retrieval* (r, α, lt, rt, R)

Inputs: The root of the CBS-Tree r , the predicate episode α , the specified time interval $[lt, rt]$, and the rejected event instance set R deduced from the min-latest occurrence of α last retrieved.

Output: The min-latest occurrence of α with end time rt .

1. **If** the event instance with arrival time rt does not correspond to a sink of α , return and report there does not exist the desired episode occurrence ;
Else let the event instance be the mapping instance of the sink corresponding to it ;
2. **While** there exists a vertex v of α , which does not map to an event instance and is a sink of α or all the direct successors of v have the mapping instances over the event stream do
3. { **If** v is a sink node, $t' = rt - 1$;
Else let $\{v_1, v_2, \dots, v_i\}$ be the set of direct successors of v , $\{(\epsilon(v_1), t_1), (\epsilon(v_2), t_2), \dots, (\epsilon(v_i), t_i)\}$ be the set of the mapping instances of $\{v_1, v_2, \dots, v_i\}$, and $t_1 < t_2 < \dots < t_i$, set $t' = t_1 - 1$;
4. $t = \text{FindLastestEventOccurrence}(v.\text{event}, lt, t', r)$; // $v.\text{event}$: the event corresponding to vertex v ; r : the root of the CBS-Tree
5. **If** ($t = -1$) Return and report that there does not exist the min-latest occurrence with end time rt ; }
6. Derive the rejected event instances from the newly retrieved min-latest occurrence by CBS-Tree, and report the retrieved min-latest occurrence ;

We now describe how the procedure *FindLastestEventOccurrence* proceeds by using CBS-Tree. In general, the retrieval for a latest instance begins at the root, and iteratively checks some children of the current node until a leaf node is reached. We will show that the total number of paths we have to check from the root to leaves is always at most 2. Hence the time complexity for retrieving an event instance is $O(\log L)$ with a multiplier less than or equal to 2.

In the following, we will focus on the general case of CBS-Tree in which every leaf corresponds to an event instance. At the end of this section, we then discuss the special case of CBS-Tree where some leaves do not correspond to event instances. For each node $\acute{n}$, we use $\acute{n}.\text{id}$, $\acute{n}.\text{lch}$, $\acute{n}.\text{rch}$, $\acute{n}.\text{LR}_E$ and $\acute{n}.\text{RR}_E$ to denote the node ID, the left child, the right child, and the event sets of the left child and the right child, respectively. Let $\acute{n}.\text{lmt}$ and $\acute{n}.\text{rmt}$ be the timestamps corresponding to the event instances represented by the leaf nodes under $\acute{n}$ whose ID number is the smallest (the left-most leaf node) and the largest (the right-most leaf node), respectively. Moreover, we use $\acute{n}.\text{lat}$ and $\acute{n}.\text{smt}$ to denote the largest and smallest timestamps in the timestamp set of node $\acute{n}$, respectively. The leaf nodes corresponding to them are denoted as n_{latest} and n_{earliest} , respectively. According to the location of n_{latest} under $\acute{n}$, we have three types of situations to consider. As shown in Fig. 6, Type 1 means that n_{latest} is the rightmost leaf node, while the other two types indicate that n_{latest} is under either $\acute{n}.\text{lch}$ or $\acute{n}.\text{rch}$. The timestamps are also depicted in the figures.

Two useful properties for event instance retrieval are explored from the relation of the timestamps represented by the leaf nodes under a given node. We respectively present them in Property 2 and Property 3.

Property 2. Given a CBS-Tree and a node $\acute{n}$ in it, if and only if $\acute{n}.lch.lat < \acute{n}.rch.smt$, among the leaf nodes under $\acute{n}$, the latest updated one (n_{latest}) must be the right-most leaf node.

Based on the above property, the following corollary can be derived.

Corollary 1. Given a CBS-Tree and a node $\acute{n}$ in it, if $\acute{n}.lch.lat < \acute{n}.rch.smt$, (1) for any two leaf nodes under $\acute{n}$ l_1 and l_2 , if and only if $l_1.id < l_2.id$, the timestamp of $l_1 <$ that of l_2 , and (2) $\acute{n}.lch.smt = \acute{n}.lch.lmt$, $\acute{n}.lch.lat = \acute{n}.lch.rmt$, $\acute{n}.rch.smt = \acute{n}.rch.lmt$, and $\acute{n}.rch.lat = \acute{n}.rch.rmt$.

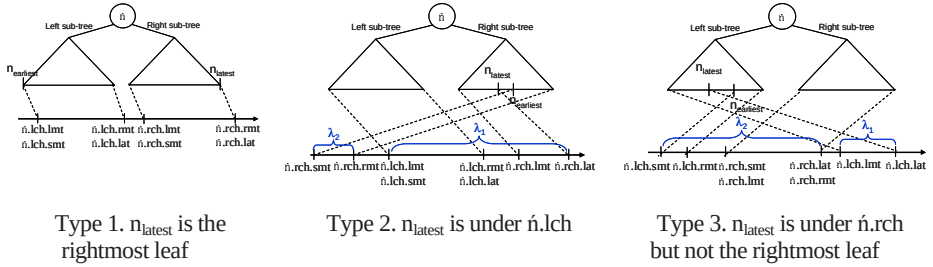


Fig. 6. Three types of situations

The latest event instance retrieval can be categorized into four cases. The first three are treated based on Property 2, while the last one is treated based on Property 3, which will be presented later. The first three cases occur when $\acute{n}.lch.lat < \acute{n}.rch.smt$, and by Property 2, only Type 1 in Fig. 6 should be considered. Given a query event $\acute{e}$, the specified time interval $[lt, rt]$, and the currently retrieved node $\acute{n}$, the largest value of rt and the smallest value of lt are limited by $\acute{n}.rch.rmt$ and $\acute{n}.lch.lmt$, respectively. We therefore divide the specifications of lt and rt into three cases: (1) $rt = \acute{n}.rch.rmt$, (2) $lt = \acute{n}.lch.lmt$ and $rt < \acute{n}.rch.rmt$, and (3) $lt > \acute{n}.lch.lmt$ and $rt < \acute{n}.rch.rmt$. Fig. 7 shows these cases and their sub-cases, which are detailed below. In these cases, if $[lt, rt]$ has an overlap with the time interval of $\acute{n}.rch$, we should first consider $\acute{n}.rch$ and then $\acute{n}.lch$.

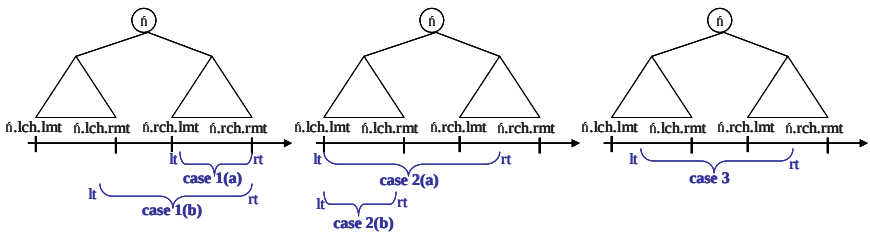


Fig. 7. Illustration of the first three cases

Case 1. $rt = \acute{n}.rch.rmt$. There are two sub-cases. (a) The specified time interval $[lt, rt]$ is entirely covered by the time interval $[\acute{n}.rch.lmt, \acute{n}.rch.rmt]$, i.e., $lt \geq \acute{n}.rch.lmt$. By the definition of latest event instance, if there is an answer, it must be found in the leaf nodes under $\acute{n}.rch$. Therefore, if $\acute{e}$ is contained in the event set of $\acute{n}.rch$, we

then search the right sub-tree. Note that, we pass the larger of lt and $\acute{n}.rch.lmt$ as the left bound of the specified time interval for searching the right sub-tree. (b) The specified time interval $[lt, rt]$ has an overlap with the time interval $[\acute{n}.lch.lmt, \acute{n}.lch.rmt]$, i.e., $lt \leq \acute{n}.lch.rmt$. We first check whether there is an instance of $\acute{e}$ in the right sub-tree, i.e., check whether $\acute{e}$ is contained in the event set of $\acute{n}.rch$. If so, we can obtain the answer by searching the right sub-tree. Otherwise, we check if there exists any instance of $\acute{e}$ in the left sub-tree by passing the time interval $[lt, \acute{n}.lch.rmt]$. To sum up for case 1, only one of the two sub-trees should be searched. Therefore, we can make a unique choice on the two children to follow during the retrieval. Moreover, the same procedure of case 1 will be repeated in the next tree level. As a result, at most 1 path of CBS-tree should be traversed in case 1.

Case 2. $rt < \acute{n}.rch.rmt$ and $lt = \acute{n}.lch.lmt$. There are also two sub-cases. (a) The specified time interval $[lt, rt]$ has an overlap with the time interval $[\acute{n}.rch.lmt, \acute{n}.rch.rmt]$, i.e., $rt \geq \acute{n}.rch.lmt$. Two steps are needed for this case. At step (i), we check whether there is an instance of $\acute{e}$ within $[\acute{n}.rch.lmt, rt]$. If not, we then proceed to step (ii), searching the left sub-tree. (b) The specified time interval $[lt, rt]$ is entirely covered by the time interval $[\acute{n}.lch.lmt, \acute{n}.lch.rmt]$, i.e., $rt \leq \acute{n}.lch.rmt$. Obviously, if there is an answer, it must be found in the left sub-tree. Note that, we pass the smaller of rt and $\acute{n}.lch.rmt$ as the right bound of the specified time interval for searching the left sub-tree in both sub-cases. Moreover, it will fall into case 2 again in the next tree level, when step (i) of sub-case (a) and sub-case (b) are processed, and fall into case 1, when step (ii) of sub-case (a) is processed. To sum up, at most 2 paths will be traversed if the answer exists. One is to repeatedly apply case 2 in the deeper tree levels, and then finally $\acute{e}$ is not found in the leaves, while the other is for the execution of step (ii) of sub-case (a) (case 2 turns to case 1), and the answer is found.

Case 3. $rt < \acute{n}.rch.rmt$ and $lt > \acute{n}.lch.lmt$. This case can be reduced to case 1 or case 2, or recursively performed by itself. For this case, we first check the right sub-tree, and then the left sub-tree if no answer exists in the right sub-tree. Note that, the parameters lt and rt used for searching a sub-tree should be set as close as possible. Moreover, if $\acute{n}.rch.lmt$ is assigned as lt to search the right sub-tree, case 3 will be reduced to case 2 in the next tree level. Otherwise, case 3 will be reconsidered again. Similarly, if $\acute{n}.lch.rmt$ is assigned as rt to search the left sub-tree, case 3 will be reduced to case 1 in the next tree level. Otherwise, case 3 will also be reconsidered again. In this way, if the answer exists in the right sub-tree, we can obtain it by traversing at most 2 paths according to case 2. Otherwise, it costs at most 1 path to see that there is no answer in the right sub-tree, and at most 1 path to retrieve the left sub-tree according to case 1. In total, case 3 needs at most 2 paths to obtain the answer.

We now present Property 3 followed by Case 4.

Property 3. Given a CBS-Tree and a node $\acute{n}$ in it, let n_{latest} be the latest updated one among the leaves under $\acute{n}$. For any two leaf nodes under $\acute{n}$ n_x and n_y , the timestamp of n_x is larger than that of n_y if $n_x.id \leq n_{latest}.id$ and $n_y.id > n_{latest}.id$.

According to Property 2, the following corollary can be derived.

Corollary 2. Given a CBS-Tree and a node $\acute{n}$ in it, for two leaf nodes under $\acute{n}$ n_x and n_y , where $n_x.id \leq n_{latest}.id$ and $n_y.id \leq n_{latest}.id$, the timestamp of n_x is smaller than

that of n_y if and only if $n_x.id < n_y.id$. The same statement also applies to the case where $n_x.id > n_{latest}.id$ and $n_y.id > n_{latest}.id$.

The last case occurs when $\acute{n}.lch.lat > \acute{n}.rch.smt$, and by Property 3, either Type 2 or Type 3 in Fig. 6 should be considered. Divide the leaves under $\acute{n}$ into two non-empty sets λ_1 and λ_2 such that $n_x.id \leq n_{latest}.id \ \forall n_x \in \lambda_1$ and $n_y.id > n_{latest}.id \ \forall n_y \in \lambda_2$. In Type 2, n_{latest} is under $\acute{n}.lch$ and the time interval of the left sub-tree can be thus divided into two regions $[\acute{n}.lch.lmt, \acute{n}.lch.lat]$ and $[\acute{n}.lch.smt, \acute{n}.lch.rmt]$. Moreover, the two sets of leaves λ_1 and λ_2 are associated with timestamps in the intervals $[\acute{n}.lch.lmt, \acute{n}.lch.lat]$ and $[\acute{n}.lch.smt, \acute{n}.rch.rmt]$, respectively. To find the latest instance of an event, we should first consider λ_1 and then λ_2 . In Type 3, n_{latest} is a leaf node under the right sub-tree but not the right-most one. Similarly, the time interval of the right sub-tree is divided into two regions $[\acute{n}.rch.lmt, \acute{n}.rch.lat]$ and $[\acute{n}.rch.smt, \acute{n}.rch.rmt]$. Moreover, in this situation, λ_1 and λ_2 correspond to the interval $[\acute{n}.lch.lmt, \acute{n}.rch.lat]$ and $[\acute{n}.rch.smt, \acute{n}.rch.rmt]$, respectively.

Case 4. There are two sub-cases to consider in this case. (a) $rt \geq \acute{n}.lch.lmt$. It means that rt is located at the interval of λ_1 for either Type 2 or Type 3, as shown in Fig. 8, and thus the interval $[lt, rt]$ may span the intervals of both sets λ_1 and λ_2 . Therefore, we first search λ_1 in the interval $[\acute{n}.lch.lmt, rt]$ and then λ_2 in $[lt, \acute{n}.rch.rmt]$. For the retrieval in $[\acute{n}.lch.lmt, rt]$, if $lt > \acute{n}.lch.lmt$, $[lt, rt]$ is entirely covered by $[\acute{n}.lch.lmt, rt]$, which can be regarded as case 3. Otherwise, the retrieval in $[\acute{n}.lch.lmt, rt]$ belongs to case 2 since the left bound is fixed to $\acute{n}.lch.lmt$. Therefore, it requires at most two paths to find the answer or one path to recognize no answer in $[\acute{n}.lch.lmt, rt]$. For the later, the interval $[lt, \acute{n}.rch.rmt]$ (λ_2) is then considered. Similarly, the retrieval in $[lt, \acute{n}.rch.rmt]$ belongs to case 1 since the right bound is fixed to $\acute{n}.rch.rmt$. Therefore, it only requires one path for the retrieval. (b) $rt < \acute{n}.lch.lmt$. Since the leaf node corresponding to $\acute{n}.lch.lmt$ is the leftmost one in the set λ_1 , any leaf node in the interval $[lt, rt]$ must be in the set λ_2 . Depending on whether rt is equal to $\acute{n}.lch.lmt$ or not, this sub-case is equivalent to either case 1 ($rt = \acute{n}.lch.lmt$) or case 3 ($rt < \acute{n}.lch.lmt$).

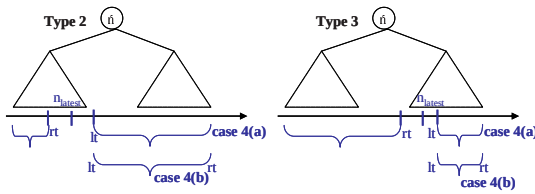


Fig. 8. The sub-cases of Case 4

Given the event $\acute{e}$, the specified time interval $[lt, rt]$, and a node of the CBS-Tree $\acute{n}$, the procedure `FindLatestEventOccurrence` iteratively considers one of the four cases presented above. During the latest event instance retrieval, whenever an instance in a leaf is retrieved, we check if it is a rejected event instance. If so, we terminate the retrieval; otherwise, the occurring time of the instance is returned.

Let N be the number of vertices in an episode. The time complexity of CBS-Tree will be $O(N \log L)$. For the special case that the number of received event instances

is smaller than L , we adopt the *DirectMatch* [2] which simply retrieves the min-latest occurrence of a predicate episode from raw data.

4 Experiment Results

In this section, we evaluate the performances of CBS-Tree and ToFel by a series of experiments on real datasets. All the experiments are performed on a Pentium IV 2.4GHz PC with 2GB RAM and under the Microsoft Windows XP environment. We consider two real datasets Intel lab data and Traffic data for performance evaluation, which are respectively described in [3].

All the notations used for generating predicate episodes and their definitions are stated as follows. *AveV*: the average number of vertices in a predicate episode, *AveE*: the average number of out-edges for a vertex in the predicate episodes, and *AveW*: the average size of the predicate window in a predicate episode. Let $I(\delta)$ be the function of normal distribution with mean δ . To generate a predicate episode α from the real dataset, we first generate the corresponding predicate window ω_α by $\Gamma(\text{AveWindow})$. Then, we randomly pick $\Gamma(\text{AveVertex})$ events as vertices from the records with record numbers $t, t+1, t+2, \dots, t+\text{AveWindow}-1$ in the dataset. Let the vertices in α be $v_1, v_2, \dots, v_n$. For each vertex $v_k (1 \leq k \leq n)$, we compute the number of its out-edges n_{edge} from $\Gamma(\text{AveE})$. Then, we randomly select n_{edge} different vertices from the set $\{v_i | k < i\}$ as its direct successors. To avoid a vertex $v_j (1 < j \leq n)$ being over-connected, we also add *AveE* dummy vertices denoted as $v_{n+1}, v_{n+2}, \dots, v_{n+\text{AveE}}$ during the edge construction. If v_k selects one of these vertices to be its direct successor, no operation is carried out for this connection. If there are many isolated sub-graphs in a seed, we add the minimum number of edges to combine them into a connected graph. Note that, we only show the performance of our approach on the matching of the min-latest occurrences, since it is the main cost of the event prediction problem. Therefore, we only need to generate the set of predicate episodes with predicate windows instead of the episode rules.

We set various parameters to evaluate our efficiency with respect to the sizes of predicate windows and the structure of episodes. If not specified in the experiments, the default settings for the number of episodes, *AveV*, and *AveE* are 10,000, 5, and 2, respectively. In the following, each figure corresponds to the results from a set of experiments. Moreover, each point of a curve in the figure stands for the execution time of one approach.

At first, we are interested in how the data distribution and the size of predicate window influence the performances of CBS-Tree and ToFel. Given the predicate episode α , if the event instances corresponding to the vertices of α come often, ToFel needs to frequently maintain the queues of α for the arrived event instances even if they finally do not form the min-latest occurrences. On the other hand, since the matching of min-latest occurrence in CBS-Tree is triggered only by the event instances corresponding to the sinks of α , the above situation which causes ToFel to frequently perform queue updates is not significant to CBS-Tree. Therefore, ToFel is more sensitive than CBS-Tree when the stream data is relatively dense. Table 1 shows the total frequencies of the k events with highest frequencies in the two real

Table 1. Event frequencies in the real datasets

	k=5	k=10	k=15	k=20	k=25	k=30	k=35	k=40
Intel lab dataset	1.3%	2.6%	3.8%	4.8%	5.8%	6.7%	7.6%	8.4%
Traffic dataset	26.4%	37.3%	44.2%	49.5%	53.4%	56.2%	58.6%	60.6%

datasets, respectively. Intuitively, a dataset with a smaller k and a larger total frequency indicates a denser dataset. As we can see, the density of Traffic dataset is significantly higher than Intel lab dataset.

For a sparse dataset, as the size of predicate window becomes larger, the event instances are more likely to become a part of the minimal occurrence and the number of answers increases. Therefore, the execution times of the two approaches will grow with the increase of AveW since there are more episode occurrences in a large window (this also implies more minimal occurrences). For a dense dataset, events can form episode occurrences in a small window. However, in a large window, since some occurrences turn to redundant ones, the number of minimal occurrences is thus reduced. As a result, the execution times of the two approaches will decrease with the growth of AveW. Fig. 9 and Fig. 10 depict the performance of the two approaches with respect to the AveW values varying from 50 to 500 on Intel lab data and the traffic data, respectively. CBS-Tree outperforms ToFel for all the cases in both figures. In Fig. 9, the execution time of CBS-Tree slightly increases with the growth of window size, but is still better than that of ToFel in the cases of larger window sizes. In Fig. 10, since the traffic dataset is denser than Intel lab dataset, ToFel spends significantly more time to process the traffic data against the Intel lab data. In the following, we evaluate the performances of CBS-Tree and ToFel by three sets of experiments with different settings of AveV, AveE, and the number of predicate episodes, respectively. Due to the lack of space, we will only show the cases that set AveW to 500.

The execution times of the two approaches with various AveV values are shown in Fig. 11 and Fig. 12. As the number of vertices in an episode increases, the two approaches need more time to match a min-latest occurrence, and the number of answers decreases as the AveV value increases. CBS-Tree outperforms ToFel in either Intel lab data or the traffic data. In the experiments, most of the irrelevant event instances can be

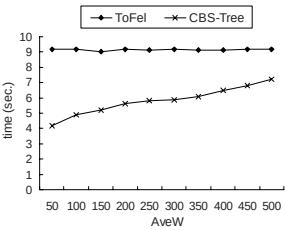


Fig. 9. Execution time with different AveW on Intel lab data

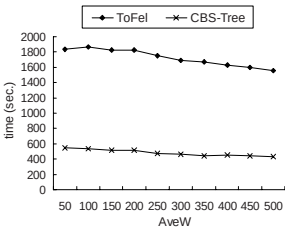


Fig. 10. Execution time with different AveW on Traffic data

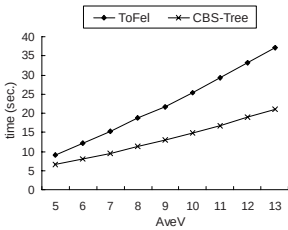


Fig. 11. Execution time with different AveV on Intel lab data

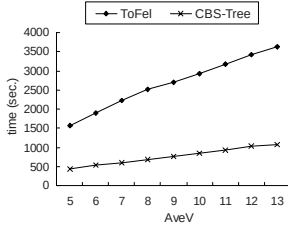


Fig. 12. Execution time with different AveV on Traffic data

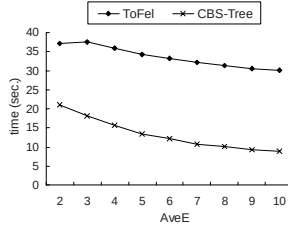


Fig. 13. Execution time with different AveE on Intel lab data

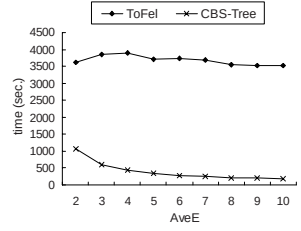


Fig. 14. Execution time with different AveE on Traffic data

efficiently skipped by CBS-Tree when the value of AveV is large. Moreover, CBSTree exhibits an excellent performance when the dataset is relatively dense.

In Fig. 13 and Fig. 14, we evaluate the performances of the two approaches with respect to different settings of AveE. In this experiment, we set AveV to 14 for all the cases. In general, the larger the number of edges in an episode is, the stronger the temporal constraints among vertices in the episode will be. This results in a smaller number of min-latest occurrences in the answer set. Therefore, the execution time of the approaches decreases as the AveE value increases. In the figures, CBS-Tree exhibits better performance than ToFel no matter how large the value of AveE is. For ToFel, when the number of edges in an episode is high, ToFel will take more time to maintain the queue (In ToFel, an event instance for a vertex can be kept if there are some event instances kept for all its direct predecessors. Intuitively, there are more direct predecessors for a vertex when the AveE value is larger). This cancels out the time saved when fewer answers are found under a large value of AveE.

5 Conclusion and Future Work

In this paper, we propose a novel approach to match the predicate episode of an episode rule over event streams for the prediction of the consequent event. Our approach only finds the minimal occurrence such that the duplicate prediction can be avoided. The approach is based on the CBS-Tree which maintains the recent event instances such that the minimal occurrence can be efficiently retrieved. We formulate four cases for retrieving the event instances of the minimal occurrence via CBS-Tree. Each of the four cases only takes $O(\log L)$ time complexity to retrieve the desired event instance, where L is the number of event instances indexed in the tree. We evaluate the performance of our approach by varying the size of predicate window and the structural scale of the episode such as the number of vertices and the number of edges between two vertices. The experiment results show that our approach outperforms the previous work in most cases. The results also reveal that CBS-Tree has the outstanding performance when the dataset is relatively dense. In the future, we will extend our approach to handle more complex predicates such as the episodes with negation operators or the events with multiple attributes.

References

1. Abadi, D.J., et al.: Aurora: A Data Stream Management System. In: Proceedings of the ACM SIGMOD Conference, p. 666 (2003)
2. Cho, C.W., Zheng, Y., Chen, A.L.P.: Continuously Matching Episode Rules for Predicting Future Events over Event Streams. In: Proceedings of joint conference of Asia-Pacific Web Conference and International Conference on Web-Age Information Management, pp. 884–891 (2007)
3. Cho, C.W., Zheng, Y., Chen, A.L.P.: CBS-Tree: Event Prediction Using Episode Rules over Event Streams, Tech. Report CS-1207-31, Department of Computer Science, National Tsing Hua University (December 2007)
4. Demers, A.J., Gehrke, J., Hong, M.S., Riedewald, M., White, W.M.: Towards Expressive Publish/Subscribe Systems. In: Proceedings of International Conference on Extending Database Technology, pp. 627–644 (2006)
5. Franklin, M.J., Jeffery, S.R., Krishnamurthy, S., Reiss, F., Rizvi, S., Wu, E., Cooper, O., Edakkunni, A., Hong, W.: Design Considerations for High Fan-In Systems: The HiFi Approach. In: Proceedings of Biennial Conference on Innovative Data Systems Research, pp. 290–304 (2005)
6. Gatzia, S., Dittrich, K.R.: SAMOS: an Active Object-Oriented Database System. IEEE Database Engineering Bulletin 15(1-4), 23–26 (1992)
7. Gehani, N.H., Jagadish, H.V., Shmueli, O.: Composite Event Specification in Active Databases: Model & Implementation. In: Proceedings of International Conference on Very Large Data Bases, pp. 327–338 (1992)
8. Hall, F.L.: Traffic stream characteristics, Traffic Flow Theory. U.S. Federal Highway Administration (1996)
9. Hätönen, K., Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H.: Knowledge Discovery from Telecommunication Network Alarm Databases. In: Proceedings of International Conference on Data Engineering, pp. 112–115 (1996)
10. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery 1(3), 259 (1997)
11. Ng, A., Fu, A.W.C.: Mining Frequent Episodes for Relating Financial Events and Stock Trends. In: Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 27–39 (2003)
12. Wang, F., Liu, P.: Temporal Management of RFID Data. In: Proceedings of International Conference on Very Large Data Bases, pp. 1128–1139 (2006)
13. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the ACM SIGMOD Conference, pp. 407–418 (2006)

Effective Skyline Cardinality Estimation on Data Streams^{*}

Yang Lu, Jiakui Zhao, Lijun Chen, Bin Cui, and Dongqing Yang

Key Laboratory of High Confidence Software Technologies (Peking University),
Ministry of Education, China
School of Electronics Engineering and Computer Science, Peking University, China
{yanglu,jkzhao,ljchen,bin.cui,dqyang}@pku.edu.cn

Abstract. In order to incorporate the skyline operator into the data stream engine, we need to address the problem of skyline cardinality estimation, which is very important for extending the query optimizer's cost model to accommodate skyline queries. In this paper, we propose robust approaches for estimating the skyline cardinality over sliding windows in the stream environment. We first design an approach to estimate the skyline cardinality over uniformly distributed data, and then extend the approach to support arbitrarily distributed data. Our approaches allow arbitrary data distribution, hence can be applied to extend the optimizer's cost model. To estimate the skyline cardinality in online manner, the live elements in the sliding window are sketched using Spectral Bloom Filters which can efficiently and effectively capture the information which is essential for estimating the skyline cardinality over sliding windows. Extensive experimental study demonstrates that our approaches significantly outperform previous approaches.

1 Introduction

Skyline queries are very important for many applications, such as data mining and multi-criteria decision making, and have attracted much attention [4,9,12]. Given two multi-dimensional elements ξ_1 and ξ_2 , if ξ_1 is better than or equal to ξ_2 over all dimensions and strictly better than ξ_2 over at least one dimension, we say that ξ_1 dominates ξ_2 , and is marked as $\xi_1 \succ \xi_2$. If an element is not dominated by any other element, it is a skyline element, and the skyline query returns all skyline elements. Continuously monitoring skylines over sliding windows [11,14] in the stream environment also received much attention; the skyline changes over time as the window slides and the skyline changes are reported to the user continuously in real-time manner. In order to incorporate the skyline operator into the data stream engine, we need to solve the problem of skyline cardinality estimation, which is very important for extending the optimizer's cost model.

There are some previous works [2,5] which considered the problem of skyline cardinality estimation over static datasets. However, the approaches are based

^{*} This work is supported by project 2007AA01Z153 under the National High-tech Research and Development of China and the National Natural Science Foundation of China under Grant No.60603045.

on very strong assumptions on the data distribution, e.g., no duplicate values over each dimension. The approach in [6] allows duplicate values, but only two possible values, e.g. 0 and 1, are allowed over the dimension which contains duplicate values, hence the restriction is still very strong. Since duplicate values are very common, above approaches do not scale well to real-life applications. In this paper, we propose robust approaches for estimating the skyline cardinality, and our approaches can support the skyline computation over arbitrarily distributed data. In addition, since the elements in the sliding window change over time as the window slides, we use Spectral Bloom Filters [7] to continuously capture the information which is essential for estimating the skyline cardinality. Our contributions in this paper can be summarized as follows:

1. We propose an approach which only uses the value cardinality of each dimension to estimate the skyline cardinality, under the assumption that the data over each dimension is uniformly distributed.
2. We design a robust approach which considers the data distribution over each dimension. This enhanced approach can estimate the skyline cardinality effectively and efficiently over arbitrarily distributed data.
3. We propose to use Spectral Bloom Filters to capture the information, such as value cardinality and value frequency over each dimension, which is essential for estimating the skyline cardinality over sliding windows.
4. We conduct extensive experimental study to demonstrate that our approaches yield better performance than existing approaches.

The rest of this paper is organized as follows: Section 2 surveys related works; Section 3 proposes our skyline cardinality estimation approaches; experimental results are shown in Section 4, followed by our conclusion in Section 5.

2 Related Works

The skyline problem was originally studied as the maximal vector problem; Kung et al. [10] proposed the first algorithm for finding the maximal vectors from a set of memory resident vectors. Börzsönyi et al. [4] introduced the skyline operator into relational database systems. Recently, continuously monitoring skylines over sliding windows [11,14] also received much attention.

There are some previous works which considered the problem of skyline cardinality estimation over static datasets. Under assumptions of statistical independence across dimensions, no duplicate values over each dimension, and dimension domains are all totally ordered, Bentley et al. [2] and Godfrey [8] proposed methods for estimating the skyline cardinality using carefully designed recurrence. Under the same assumptions, Buchta [5] and Chaudhuri et al. [6] proposed to estimate the skyline cardinality using integrals, and Buchta [5] further derived that the skyline cardinality equals $\Theta((\ln n)^{k-1}/(k-1)!)$, where n is the number of elements in the space and k is the number of dimensions. Above approaches cannot be applied to most real-life applications as the “no duplicate values” assumption is impractical. Chaudhuri et al. [6] relaxed the “no duplicate values”

assumption, but the dimensions which contain duplicate values can only have two possible values, which is still a suffering constraint. In addition, the approach uses integrals to estimate the skyline cardinality, and the integrals have no a close form, hence can only be approximated by some scientific computing methods. In this paper, under assumptions of statistical independence across dimensions which is commonly used by query optimizers, we consider the problem of sliding-window skyline cardinality estimation over arbitrarily distributed data in the stream environment.

3 Skyline Cardinality Estimation

In this section, we introduce how to estimate the skyline cardinality over sliding windows in the stream environment.

3.1 Estimation under Strong Assumptions

The approaches for skyline cardinality estimation proposed in [2,5,6] can be extended to the stream context without modifications and are summarized by Theorem 1, Theorem 2 and Theorem 3 respectively. Theorem 1 and Theorem 2 are theoretically equivalent. The three approaches suffer from the strong assumptions as described in Section 2, and hence do not apply to real-life applications.

Theorem 1. *Suppose that there are n k -dimensional live elements in a sliding window; under assumptions of statistical independence across dimensions, no duplicate values over each dimension, and dimension domains are all totally ordered, the expected number of the skyline elements $\Psi(n, k)$ can be recursively characterized as*

$$\Psi(n, k) = \Psi(n - 1, k) + \frac{1}{n}\Psi(n, k - 1)$$

with initial conditions

$$\begin{aligned}\Psi(1, k) &= 1 & k \geq 1 \\ \Psi(n, 1) &= 1 & n \geq 1.\end{aligned}$$

Theorem 2. *Suppose that there are n k -dimensional live elements in a sliding window; under the same assumptions as those in Theorem 1, the expected number of the skyline elements $\Psi(n, k)$ can be characterized as*

$$\Psi(n, k) = n \int_0^1 \cdots \int_0^1 (1 - x_1 \cdots x_k)^{n-1} dx_1 \cdots dx_k.$$

Theorem 3. *Suppose that there are n k -dimensional live elements in a sliding window; there are only two possible values over each of the first k° dimensions, and there are no duplicate values over the other dimensions. For simplicity and without loss of generality, suppose that the two possible values over the first k° dimensions are 1 and 0, and 1 dominates 0; p_i denotes the probability that the*

value of the i th dimension equals 1, where $1 \leq i \leq k^\circ$. Under assumptions of statistical independence across dimensions and dimension domains are all totally ordered, the expected number of the skyline elements $\Psi(n, k)$ equals

$$n \sum_{v \in \{0,1\}^{k^\circ}} \int_0^1 \cdots \int_0^1 \Phi_v(x_1, \dots, x_{k-k^\circ}) dx_1 \cdots dx_{k-k^\circ}$$

where $\Phi_v(x_1, \dots, x_{k-k^\circ})$ can be characterized as

$$\begin{aligned} \Phi_v(x_1, \dots, x_{k-k^\circ}) &= \mathbb{P}_1(v) (1 - \mathbb{P}_2(v) x_1 \cdots x_{k-k^\circ})^{n-1} \\ \mathbb{P}_1(v) &= \prod_{i=1}^{k^\circ} p_i^{v_i} (1 - p_i)^{1-v_i} \\ \mathbb{P}_2(v) &= \prod_{i=1}^{k^\circ} p_i^{1-v_i}. \end{aligned}$$

3.2 Estimation over Uniformly Distributed Data

As we introduced previously, existing approaches assume that there are no duplicate values over each dimension or the dimension only has two possible values. To achieve better applicability, we relax the restriction on the data distribution. Under assumptions of statistical independence across dimensions and the data over each dimension is uniformly distributed, we use the value cardinality, i.e. the number of the distinct elements, over each dimension to characterize the expected number of the skyline elements. The theoretical analysis is based on the *Inclusion-Exclusion Principle* [13]. Lemma 1 gives the probability that an element is dominated by all other n elements. Lemma 2 gives the probability that an element is dominated by at least one of other n elements. Lemma 3 gives the probability that none of other n elements can dominate an element. Theorem 4 gives the expected number of the skyline elements in a sliding window which contains n k -dimensional live elements.

Lemma 1. Suppose that $\xi_0 \xi_1 \cdots \xi_n$ are $n+1$ k -dimensional elements, where $\xi_i = \langle x_{i1}, x_{i2}, \dots, x_{ik} \rangle$. The data over each dimension is uniformly distributed, and the value cardinality of the j th dimension, i.e. the number of the distinct values over the j th dimension, is c_j ; $v_{j1}, v_{j2}, \dots, v_{jc_j}$ denote the distinct values over the j th dimension, where $v_{j1} < v_{j2} < \cdots < v_{jc_j}$. Under assumptions of statistical independence across dimensions, the probability that $\forall_{i(1 \leq i \leq n)} (\xi_i \succ \xi_0)$, i.e. $\mathbb{P}_\circ(n)$, can be characterized as,

$$\mathbb{P}_\circ(n) = \prod_{j=1}^k \sum_{t=1}^{c_j} \frac{t^n}{c_j^{n+1}} \quad n \geq 1, k \geq 1.$$

Proof. $\mathbb{P}_\circ(n)$ can be characterized as,

$$\mathbb{P}_\circ(n) = \mathbb{P}\{(\forall_{i(1 \leq i \leq n)} (\xi_i \succ \xi_0))\}$$

$$\begin{aligned}
&= \prod_{j=1}^k \sum_{t=1}^{c_j} \mathbb{P}\{x_{0j} = v_{jt}\} \mathbb{P}\{\forall_{i(1 \leq i \leq n)} (x_{0j} \geq x_{ij}) \mid x_{0j} = v_{jt}\} \\
&= \prod_{j=1}^k \sum_{t=1}^{c_j} \left(\frac{1}{c_j} \cdot \left(\sum_{\theta=1}^t \frac{1}{c_j} \right)^n \right) = \prod_{j=1}^k \sum_{t=1}^{c_j} \frac{t^n}{c_j^{n+1}}.
\end{aligned}$$

Lemma 2. *Under the same conditions as those in Lemma 1, the probability that $\exists_{i(1 \leq i \leq n)} (\xi_i \succ \xi_0)$, i.e. $\mathbb{P}_{\bullet}(n)$, can be characterized as,*

$$\mathbb{P}_{\bullet}(n) = \sum_{i=1}^n \left((-1)^{i-1} \binom{n}{i} \prod_{j=1}^k \sum_{t=1}^{c_j} \frac{t^i}{c_j^{i+1}} \right) \quad n \geq 1, \quad k \geq 1.$$

Proof. Using the *Inclusion-Exclusion Principle* [13] and Lemma 1, we have

$$\begin{aligned}
\mathbb{P}_{\bullet}(n) &= \mathbb{P}\{(\xi_1 \succ \xi_0) \vee \cdots \vee (\xi_n \succ \xi_0)\} \\
&= \sum_{i=1}^n \mathbb{P}\{\xi_i \succ \xi_0\} - \sum_{1 \leq i_1 < i_2 \leq n} \mathbb{P}\{(\xi_{i_1} \succ \xi_0) \wedge (\xi_{i_2} \succ \xi_0)\} \\
&\quad + \sum_{1 \leq i_1 < i_2 < i_3 \leq n} \mathbb{P}\{(\xi_{i_1} \succ \xi_0) \wedge (\xi_{i_2} \succ \xi_0) \wedge (\xi_{i_3} \succ \xi_0)\} \\
&\quad - \cdots + (-1)^{n-1} \mathbb{P}\{(\xi_1 \succ \xi_0) \wedge \cdots \wedge (\xi_n \succ \xi_0)\} \\
&= \binom{n}{1} \mathbb{P}_{\circ}(1) - \binom{n}{2} \mathbb{P}_{\circ}(2) + \binom{n}{3} \mathbb{P}_{\circ}(3) - \cdots + (-1)^{n-1} \binom{n}{n} \mathbb{P}_{\circ}(n) \\
&= \sum_{i=1}^n (-1)^{i-1} \binom{n}{i} \mathbb{P}_{\circ}(i) = \sum_{i=1}^n \left((-1)^{i-1} \binom{n}{i} \prod_{j=1}^k \sum_{t=1}^{c_j} \frac{t^i}{c_j^{i+1}} \right).
\end{aligned}$$

Lemma 3. *Under the same conditions as those in Lemma 1, the probability that $\nexists_{i(1 \leq i \leq n)} (\xi_i \succ \xi_0)$, i.e. $\mathbb{P}_{\star}(n)$, can be characterized as,*

$$\mathbb{P}_{\star}(n) = \sum_{t_1=1}^{c_1} \cdots \sum_{t_k=1}^{c_k} \left(\prod_{i=1}^k \frac{1}{c_i} \right) \left(1 - \prod_{j=1}^k \frac{t_j}{c_j} \right)^n \quad n \geq 1, \quad k \geq 1.$$

Proof. By Lemma 2, $\mathbb{P}_{\star}(n)$ can be characterized as,

$$\begin{aligned}
\mathbb{P}_{\star}(n) &= \mathbb{P}\{\nexists_{i(1 \leq i \leq n)} (\xi_i \succ \xi_0)\} = 1 - \mathbb{P}_{\bullet}(n) \\
&= \sum_{i=0}^n \left((-1)^i \binom{n}{i} \prod_{j=1}^k \sum_{t=1}^{c_j} \frac{t^i}{c_j^{i+1}} \right) = \sum_{t_1=1}^{c_1} \cdots \sum_{t_k=1}^{c_k} \left(\prod_{l=1}^k \frac{1}{c_l} \right) \sum_{i=0}^n (-1)^i \binom{n}{i} \prod_{j=1}^k \frac{t_j^i}{c_j^{i+1}} \\
&= \sum_{t_1=1}^{c_1} \cdots \sum_{t_k=1}^{c_k} \left(\prod_{l=1}^k \frac{1}{c_l} \right) \sum_{i=0}^n \binom{n}{i} \left(- \prod_{j=1}^k \frac{t_j}{c_j} \right)^i = \sum_{t_1=1}^{c_1} \cdots \sum_{t_k=1}^{c_k} \left(\prod_{i=1}^k \frac{1}{c_i} \right) \left(1 - \prod_{j=1}^k \frac{t_j}{c_j} \right)^n.
\end{aligned}$$

Theorem 4. *Suppose that there are n k -dimensional live elements in the sliding window; the data over each dimension is uniformly distributed, and the value cardinality of the j th dimension is c_j ; under the assumption of statistical independence across dimensions, the expected number of the skyline elements $\Psi(n, k)$ can be characterized as,*

$$\Psi(n, k) = n \cdot \sum_{t_1=1}^{c_1} \cdots \sum_{t_k=1}^{c_k} \left(\prod_{i=1}^k \frac{1}{c_i} \right) \left(1 - \prod_{j=1}^k \frac{t_j}{c_j} \right)^{n-1}$$

where $n \geq 1$ and $k \geq 1$.

Proof. By Lemma 3 and $\Psi(n, k) = n\mathbb{P}_*(n-1)$, the theorem can be easily proved.

3.3 Estimation over Arbitrarily Distributed Data

Theorem 4 assumes that the data over each dimension is uniformly distributed; however, skewed data is very common in real-life datasets, and Theorem 4 may not work well for such datasets. Corollary 1 gives an approach for estimating the skyline cardinality over arbitrarily distributed data, in which probability functions of all dimensions are considered.

Corollary 1. *Suppose that $\xi_1 \xi_2 \cdots \xi_n$ are n k -dimensional live elements in a sliding window, where $\xi_i = \langle x_{i1}, x_{i2}, \dots, x_{ik} \rangle$. The probability function of the data over the j th dimension is f_j ; $\mathbb{P}\{x_{ij} = v_{jt}\} = f_j(t)$, $v_{j1} < v_{j2} < \cdots < v_{jc_j}$, where c_j is the value cardinality of the j th dimension. Under assumptions of statistical independence across dimensions, the expected number of the skyline elements $\Psi(n, k)$ equals*

$$n \cdot \sum_{t_1=1}^{c_1} \cdots \sum_{t_k=1}^{c_k} f_1(t_1) \cdots f_k(t_k) \left(1 - \prod_{j=1}^k \sum_{\theta=1}^{t_j} f_j(\theta) \right)^{n-1}.$$

Proof. The proof borrows the same ideas from the proof of Theorem 4; for space limitations, we omit the details.

Estimating the skyline cardinality using Corollary 1 has a computational complexity of $O(\prod_{j=1}^k c_j)$, where c_j is the value cardinality of the j th dimension; if the number of dimensions and the value cardinalities of some dimensions are large, the computational cost overhead is unacceptable. Definition 1 gives the definition of high and low value cardinality. If a dimension was defined with high value cardinality, we may consider that there are no duplicate values over the dimension, hence the probability function of the dimension needs not to be considered and we can reduce the computational cost thereafter. With well tuned threshold value ϵ , we can get good approximation of the skyline cardinality.

Definition 1 (High and Low Value Cardinality). *Suppose that the probability function over a dimension is f ; the value of a randomly selected element*

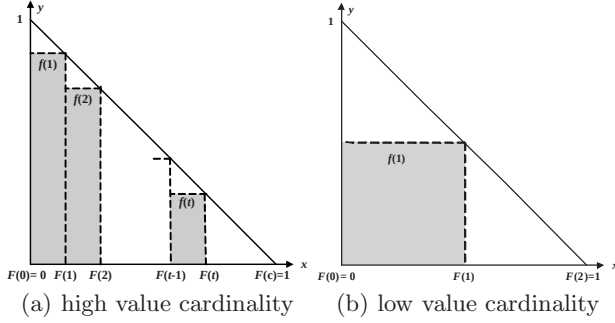


Fig. 1. High and low value cardinality

has a $f(t)$ probability to be v_t , $v_1 < v_2 < \dots < v_c$, where c is the value cardinality of the dimension; if the following inequation

$$\left| \frac{1}{2} - \sum_{t=1}^c f(t) (1 - F(t)) \right| < \epsilon, \quad F(t) = \sum_{\theta=1}^t f(\theta)$$

holds, where ϵ is the threshold value, we say that the dimension has high value cardinality; otherwise, the dimension has low value cardinality.

Figure 1 illustrates the high and low value cardinality defined in Definition 1, where the area of the triangle is 0.5; if the area of the greyed regions approximates 0.5, we are sure that any two values over the dimension have a very small probability to be equal, hence we may consider that there are no duplicate values over the dimension. Theorem 5 gives an efficient approach for estimating the skyline cardinality over arbitrarily distributed data; since the probability functions of dimensions with high value cardinality are no longer considered, the computational complexity is significantly reduced. The cardinality threshold ϵ in Definition 1 has great influence on the performance of this approach. If ϵ has a very small value, fewer dimensions can have high value cardinality; in this case, the computational complexity will be higher, but the result will be more accurate. Otherwise, more dimensions can have high value cardinality; in this case, the computational complexity will be lower, but the result may be less accurate.

Theorem 5. Suppose that $\xi_1 \xi_2 \dots \xi_n$ are n k -dimensional live elements in a sliding window, where $\xi_i = \langle x_{i1}, x_{i2}, \dots, x_{ik} \rangle$. The data over k° dimensions has low value cardinality, and the data over each other dimension has high value cardinality; without loss of generality, suppose that the data over each of the first k° dimensions has low value cardinality, and the probability function of the data over the j th dimension is f_j ; $\mathbb{P}\{x_{ij} = v_{jt}\} = f_j(t)$, $v_{j1} < v_{j2} < \dots < v_{jc_j}$, where c_j is the value cardinality of the j th dimension. If $k = k^\circ$, the expected number of the skyline elements in the sliding window $\Psi(n, k)$ equals

$$n \cdot \sum_{t_1=1}^{c_1} \dots \sum_{t_k=1}^{c_k} f_1(t_1) \dots f_k(t_k) \left(1 - \prod_{j=1}^k \sum_{\theta=1}^{t_j} f_j(\theta) \right)^{n-1}.$$

If $k > k^\circ$, $\Psi(n, k)$ can be approximated by

$$\sum_{t_1=1}^{c_1} \cdots \sum_{t_{k^\circ}=1}^{c_{k^\circ}} f_1(t_1) \cdots f_{k^\circ}(t_{k^\circ}) \Psi_{t_1, \dots, t_{k^\circ}}(n, k)$$

where $\Psi_{t_1, \dots, t_{k^\circ}}(n, k)$ can be recursively characterized as

$$\Psi_{t_1, \dots, t_{k^\circ}}(n, k) = \Psi_{t_1, \dots, t_{k^\circ}}(n-1, k) + \frac{\Psi_{t_1, \dots, t_{k^\circ}}(n, k-1)}{n}$$

with initial conditions

$$\begin{aligned} \Psi_{t_1, \dots, t_{k^\circ}}(1, k) &= 1 \quad (k \geq k^\circ + 1) \\ \Psi_{t_1, \dots, t_{k^\circ}}(n, k^\circ + 1) &= \frac{1 - \left(1 - \prod_{j=1}^{k^\circ} \sum_{\theta=1}^{t_j} f_j(\theta)\right)^n}{\prod_{j=1}^{k^\circ} \sum_{\theta=1}^{t_j} f_j(\theta)} \quad (n \geq 1). \end{aligned}$$

Proof. The proof borrows the same ideas from the proof of Theorem 4; for space limitations, we omit the details.

3.4 Computing Skyline Cardinality

In order to utilize the theorems presented above to estimate the skyline cardinality, we have to get the information about the distribution of the data, such as the value cardinality and the value frequency. In this subsection, we discuss how we can compute the skyline cardinality online in a data stream environment.

The Spectral Bloom Filter (SBF) [7] is an extension of the standard bloom filter [3] for supporting the estimation of the value frequency and the value cardinality. The bit vector in the standard bloom filter is replaced by a counter vector in SBF. Initially, all counters are set to 0; κ hash functions $h_1, h_2, \dots, h_\kappa$ are used to hash elements into the counters. Three strategies, i.e. Minimum Selection (MS), Minimal Increase (MI), and Recurring Minimum (RM), are used to maintain SBF. For sliding windows, in order to estimate the skyline cardinality using Theorem 5, the RM strategy is the best choice, since it supports deletions and has relatively lower error rate. In dynamic environments, the naive method for estimating the skyline cardinality is to recompute the expected skyline cardinality using Theorems 4 or 5 whenever the distribution is changed; however, the method is both space and time inefficient and is not necessary. In our work, in the case of estimating the skyline cardinality using Theorem 4, we use a threshold value γ_c to demonstrate that when the change of the cardinality of a dimension exceeds γ_c , the expected skyline cardinality should be recomputed. In the case of estimating the skyline cardinality using Theorem 5, an additional threshold value γ_f is used to demonstrate that when the change of the frequency of a value exceeds γ_f , the expected skyline cardinality should be recomputed.

Algorithm 1: Estimating_Skyline_Cardinality(k, F, γ_c, γ_f)

Input : k : the number of dimensions
 F : the data stream
 γ_c : threshold of cardinality change over a dimension
 γ_f : threshold of frequency change of a value

```

1 begin
2    $n \leftarrow 0$ ;
3   while the data stream  $F$  is not terminated do
4     wait until an element  $\xi$  arrives or expires;
5     if  $\xi$  is an arriving element then
6        $n \leftarrow n + 1$ ;
7       for  $i \leftarrow 1$  to  $k$  do
8          $find \leftarrow sbf[i].lookfor(\xi.x[i])$ ;
9         if  $find = false$  then
10           $v[i].insert(\xi.x[i])$ ;  $\lambda_c[i] \leftarrow \lambda_c[i] + 1$ ;
11          end
12           $sbf[i].insert(\xi.x[i])$ ;  $\lambda_f[i].insert(\xi.x[i])$ ;
13        end
14      else
15         $n \leftarrow n - 1$ ;
16        for  $i \leftarrow 1$  to  $k$  do
17           $num \leftarrow sbf[i].getnum(\xi.x[i])$ ;
18          if  $num = 1$  then
19             $v[i].delete(\xi.x[i])$ ;  $\lambda_c[i] \leftarrow \lambda_c[i] - 1$ ;
20            end
21             $sbf[i].delete(\xi.x[i])$ ;  $\lambda_f[i].delete(\xi.x[i])$ ;
22          end
23        end
24         $recompute \leftarrow false$ ;
25        for  $i \leftarrow 1$  to  $k$  do
26          if  $|\lambda_c[i]| > \gamma_c$  then
27             $recompute \leftarrow true$ ; break;
28          end
29          if  $\lambda_f[i].check(\gamma_f) = true$  then
30             $recompute \leftarrow true$ ; break;
31          end
32        end
33        if  $recompute = true$  then
34           $card \leftarrow computeT5(n, k, sbf, v)$ ; report( $card$ );
35          for  $i \leftarrow 1$  to  $k$  do
36             $\lambda_c[i] \leftarrow 0$ ;  $\lambda_f[i].clear()$ ;
37          end
38        end
39      end
40 end

```

Algorithm 1 shows how to estimate the skyline cardinality over sliding windows using Theorem 5; an array of SBFs maintained by the RM strategy are used to summarize the data over each dimension. Initially, the number of the live elements in the sliding window n is set to 0 (line 2); while the stream F is not terminated, the algorithm waits until a new element arrives or a live element expires (line 4). If a new element ξ arrives, the number of the live elements in the sliding window is increased by 1 (line 6). Then, for each dimension of the element, determine whether the dimension value is contained by the corresponding SBF (line 8); if not contained, the dimension value is inserted into the vector $v[i]$ which consists of the distinct values over the dimension and the value cardinality change over the dimension $\lambda_c[i]$ is increased by 1 (line 10). Finally, each dimension value of the new element is inserted into the corresponding

SBF and update the vector $\lambda_f[i]$ which records the frequency changes of each distinct value over the corresponding dimension. The process of processing an expired element (lines 14-23) is just the reverse process of processing an arriving element. After processing an element, the threshold values γ_c and γ_f are used to determine whether the skyline cardinality needs to be recomputed (lines 24-32); if needs to be recomputed, recompute the expected skyline cardinality using Theorem 5 and reset λ_c and λ_f (lines 33-38). `computeT5( $n, k, sbf, v$ )` computes the expected skyline cardinality, where n is the number of the live elements in the sliding window, k is the number of the dimensions, v stores the distinct values over each dimension, and sbf is the array of SBFs which can be used to estimate the number of the times that a value occurs over a dimension. Given the parameters, computing the expected skyline cardinality using Theorem 5 is rather straightforward; for space limitations, we omit the details.

The algorithm for estimating the skyline cardinality using Theorem 4 is quite similar but simpler, as we only need to record the value cardinality for each dimension and consider the only threshold γ_c . But the method may suffer from non-uniform distributions, since Theorem 4 assumes that the data over each dimension is uniformly distributed, and skewed data distribution may affect the accuracy of Theorem 4.

4 An Experimental Study

In this section, we experimentally evaluate the performance of our approaches, i.e. Theorems 4 and 5, for estimating the skyline cardinality over sliding windows in the stream environment. Since Theorems 1 and 2 are theoretically equivalent and Theorem 2 can only be approximated by some scientific computing methods, we consider Theorem 1 as a competitor of our approaches. We also compare our approaches with Theorem 3 over datasets in which the value cardinality of a dimension is limited to 2.

We use 3 hash functions and 3,000 counters for the SBF of a dimension. The algorithms are implemented by the C++ programming language and run on a 2.0GHz Intel CPU with 1GB of memory. To better show the performance under different data distributions, we conduct the experiments on synthetic datasets. We test the performance over 3-dimensional and 6-dimensional datasets which contain 30,000 elements, and the data over each dimension is generated by the GNU Scientific Library [1]. The details of the datasets used in our experiments are shown in Table 1, where c is the value cardinality of a dimension. The skewed data over a dimension is generated by two distributions alternately. One half of the skewed data submits to the uniform distribution and the other half submits to the binomial distribution with parameters p and c° . The random number generator of binomial distribution returns the number of successes in c° independent trials with probability p . For “continuous distribution” in Table 1, it represents that no duplicate values appear over a certain dimension.

For each dataset, the actual skyline cardinality is evaluated by the average number of skyline elements of the sliding window; we use the average computed

Table 1. Figures and corresponding datasets

Figure	Dataset
Figure 2(a)	2 dimensions: continuous distribution; 1 dimension: uniform distribution and $c = 50$.
Figure 2(b)	2 dimensions: continuous distribution; 1 dimension: skewed distribution, $c^\circ = 50$ and $p = 0.5$.
Figure 2(c)	1 dimension: continuous distribution; 1 dimension: uniform distribution and $c = 50$; 1 dimension: the first half data satisfies $c = 100$, the other half satisfies $c = 10$, and the data over this dimension is randomly generated.
Figure 3(a)	2 dimensions: continuous distribution; 1 dimension: uniform distribution and $c = 50$.
Figure 3(b)	1 dimension: continuous distribution; 2 dimensions: uniform distribution and $c = 50$.
Figure 3(c)	2 dimensions: continuous distribution; 1 dimension: skewed distribution, $c^\circ = 50$ and $p = 0.5$.
Figure 3(d)	1 dimension: continuous distribution; 1 dimension: skewed distribution, $c^\circ = 50$ and $p = 0.3$; 1 dimension: skewed distribution, $c^\circ = 50$, and $p = 0.7$.
Figure 3(e)	5 dimensions: continuous distribution; 1 dimension: uniform distribution and $c = 2$.
Figure 3(f)	3 dimensions: continuous distribution; 2 dimensions: uniform distribution and $c = 50$; 1 dimension: uniform distribution and $c = 2$.

skyline cardinality as the result of Theorems 4 and 5. In the experimental figures, T1, T3, T4 and T5 represent the result of Theorem 1, Theorem 3, Theorem 4, and Theorem 5 respectively.

4.1 Effect of the Thresholds

In the first set of experiments, we test the effect of the threshold ϵ , i.e. the watershed of high and low value cardinality, to the performance of Theorem 5. Figure 2(a) and Figure 2(b) illustrate the skyline cardinality with respect to ϵ over two different datasets, when the sliding window size is set to 500. In Figure 2(a), when ϵ is smaller than 0.01, the result of Theorem 5 is close to the actual skyline cardinality; but when ϵ is greater than 0.015, the results of Theorem 5 are degraded and superpose the results of Theorem 1. This happens because the dimension with uniformly distributed data is judged as a dimension with low value cardinality when $\epsilon \leq 0.01$, and hence the result is a good estimation of the actual skyline cardinality. The dimension is misjudged as a dimension with high value cardinality when $\epsilon \geq 0.015$. Thus the three dimensions are all with high value cardinality. Theorem 5 is equivalent to Theorem 1 at this moment and the results prove this point. The watershed of high and low value cardinality in Figure 2(b) is greater than that in Figure 2(a), because the value of $|1/2 - \sum_{t=1}^c f(t)(1 - F(t))|$ (see Definition 1) of a dimension in Figure 2(b) is greater than that in Figure 2(a) due to the skewed data distribution. From above results, we can see that the selection of ϵ is important for better performance of Theorem 5, and we fix the cardinality threshold ϵ at 0.005 in the following experiments.

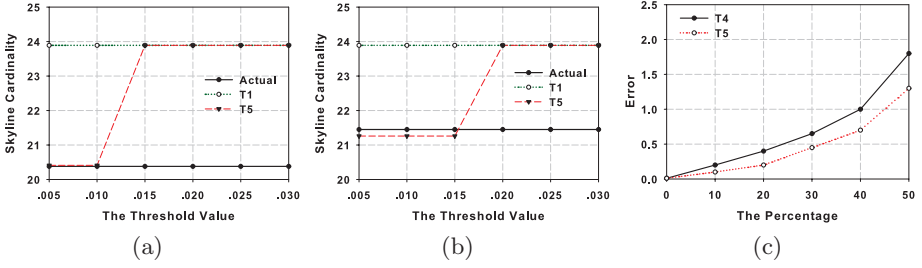


Fig. 2. The effect of thresholds

Next, we evaluate the effect of the recomputation threshold on the accuracy of skyline cardinality estimation. We fix the window size at 500. γ_c which is the threshold of cardinality change over a dimension and γ_f which is the threshold of frequency change of a value are given the same value for ease of the presentation. We only need to examine γ_c for Theorem 4, while both γ_c and γ_f for Theorem 5. In this experiment, we vary the threshold from 0% to 50% and calculate the average error and times of recomputation. For example, 10% means that we do not recompute the skyline until the change of value cardinality (value frequency) on any dimension is larger than 10%. The average error stands for the difference between the results of Theorems 4(5) and results of respective Theorems with the certain percentage of changes. As shown in Figure 2(c), both the errors of Theorem 4 and Theorem 5 increase when $\gamma_c(\gamma_f)$ increases. However, the error is not significant compared with skyline cardinality which is around 20, as we recompute the skyline once the change on any dimension exceeds the threshold. We also find that the error of Theorem 5 is smaller than that of Theorem 4. The reason is that Theorem 4 computes the skyline cardinality only considering the change of value cardinality, while Theorem 5 may recompute when the data distribution changes.

4.2 Performance over Different Datasets

Figure 3(a) to Figure 3(f) present the experimental results of different approaches under various data distributions. The actual skyline cardinality and the estimated skyline cardinality of different methods increase when the window size increases, as more objects need to be evaluated.

For the given window size and the number of dimensions, the results of Theorem 1 remain the same regardless of the change of data distributions. Therefore, Theorem 1 shows a poorer performance for the datasets, in which not all the dimensions are of continuously distributed data. In Figure 3(a) and Figure 3(b), both Theorem 4 and Theorem 5 provide a good estimation for the actual skyline cardinality and this fully supports the effectiveness of our methods. One dimension with continuously distributed data is replaced by one dimension with uniformly distributed data in Figure 3(b), and the actual skyline cardinality decreases accordingly for the intuitionistic reason that when the value cardinality

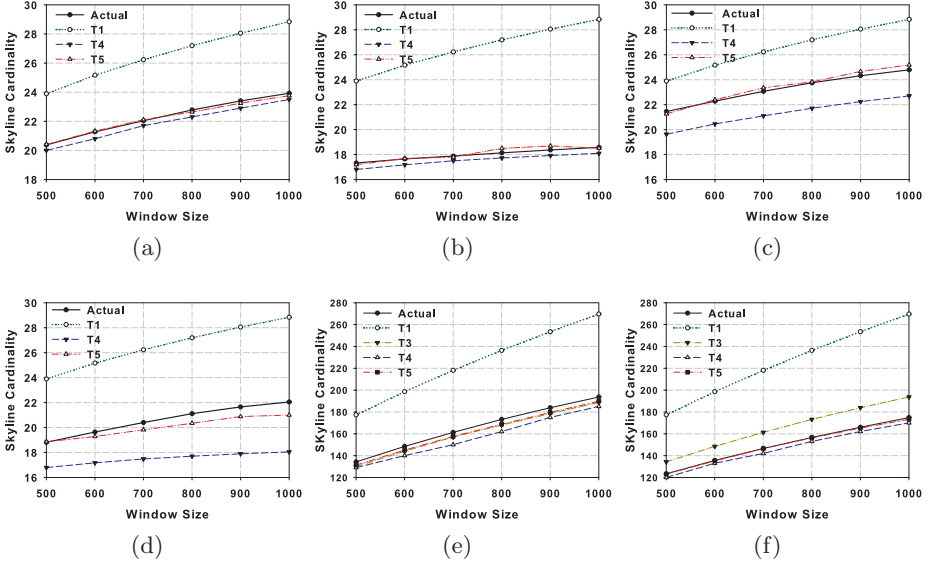


Fig. 3. Performance under different datasets

of a dimension is reduced, more elements are probably dominated. The skewed data distribution is introduced in Figure 3(c) and Figure 3(d), and Theorem 4 performs worse than Theorem 5 as we expect. The actual skyline cardinality in Figure 3(d) is smaller than that in Figure 3(c) because of the reduced value cardinality of one dimension. Comparing Figure 3(a) and Figure 3(c), we can find that the live elements in the window have higher probabilities to be skyline elements under skewed data distribution.

Next, we consider Theorem 3 as a competitor. The dataset used in Figure 3(e) satisfies the assumptions of Theorem 3. We can see that the result of Theorem 3 is almost as good as that of Theorem 5. While in Figure 3(f), we cannot compute the results of Theorem 3 directly as two dimensions are not continuously distributed. We treat them as continuously distributed to approximate the result for Theorem 3. Both Theorem 4 and Theorem 5 outperform Theorem 3, because the preconditions of Theorem 3 does not consider the data distribution whose value cardinality is greater than 2.

5 Conclusion

In this paper, we relaxed the strong assumptions of previous work and proposed Theorem 4, which only uses the value cardinality of each dimension to estimate the skyline cardinality under the assumption that the data over each dimension is uniformly distributed. We also gave a robust approach, i.e. Theorem 5, to effectively estimate skyline cardinality over arbitrarily distributed data. To apply

Theorem 4 and Theorem 5 in the data stream environment, we used SBF to capture the value cardinality and the probability that a value occurs over a dimension. Finally, we compared our approaches with all previous approaches by extensive experimental study, and our approaches, especially Theorem 5, significantly outperform previous approaches.

References

1. <http://www.gnu.org/software/gsl/>
2. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the average number of maxima in a set of vectors and applications. *J. ACM* 25(4), 536–543 (1978)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13(7), 422–426 (1970)
4. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: *Proceedings of ICDE 2001*, pp. 421–430 (2001)
5. Buchta, C.: On the average number of maxima in a set of vectors. *Inf. Process. Lett.* 33(2), 63–65 (1989)
6. Chaudhuri, S., Dalvi, N.N., Kaushik, R.: Robust cardinality and cost estimation for skyline operator. In: *Proceedings of ICDE 2006*, p. 64 (2006)
7. Cohen, S., Matias, Y.: Spectral bloom filters. In: *Proceedings of SIGMOD 2003*, pp. 241–252 (2003)
8. Godfrey, P.: Skyline cardinality for relational processing. In: Seipel, D., Turull-Torres, J.M.a. (eds.) *FoIKS 2004. LNCS*, vol. 2942, pp. 78–97. Springer, Heidelberg (2004)
9. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: *Proceedings of VLDB 2002*, pp. 275–286 (2002)
10. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. *J. ACM* 22(4), 469–476 (1975)
11. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: *Proceedings of ICDE 2005*, pp. 502–513 (2005)
12. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: *Proceedings of SIGMOD 2003*, pp. 467–478 (2003)
13. Rosen, K.H.: *Discrete Mathematics and Its Applications*, 4th edn. WCB/McGraw-Hill, Boston (1999)
14. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. *IEEE Trans. Knowl. Data Eng.* 18(2), 377–391 (2006)

Detecting Current Outliers: Continuous Outlier Detection over Time-Series Data Streams

Kozue Ishida and Hiroyuki Kitagawa

¹ Graduate School of Systems and Information Engineering

² Center for Computational Sciences

University of Tsukuba

1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

kozue-i@kde.cs.tsukuba.ac.jp, kitagawa@cs.tsukuba.ac.jp

Abstract. The development of sensor devices and ubiquitous computing have increased time-series data streams. With data streams, current data arrives continuously and must be monitored. This paper presents outlier detection over data streams by continuous monitoring. Outlier detection is an important data mining issue and discovers outliers, which have features that differ profoundly from other objects or values. Most existing outlier detection techniques, however, deal with static data, which is computationally expensive. Specifically, for outlier detection over data streams, real-time response is very important. Existing techniques for static data, however, are fraught with many meaningless processes over data streams, and the calculation cost is too high. This paper introduces a technique that provides effective outlier detection over data streams using differential processing, and confirms effectiveness.

Keywords: outlier detection, DB-Outlier, data stream, time-series data.

1 Introduction

We face an explosive increase of data, so data mining, which identifies important information and knowledge, has become increasingly important. Outlier detection is a data mining issue; it discovers outliers with features that differ greatly from other objects or values. Applications include fraud detection, network intrusion detection, and financial analysis. Various outlier detection techniques over static data have been proposed, including the statistical-based approach [1,2] and distance-based approach [3,4].

The diffusion of sensor devices and improved ubiquitous computing have brought with them an increase in time-series data streams. Because data arrives continuously, the volume of data streams is very large. Data mining techniques for data streams are therefore important; the same applies to outlier detection.

Examples of outlier detection over data streams are: to monitor observation values from sensors continuously and detect outlier sensors that show values very

	Object State set	O_1	O_2	$\cdots$	O_N
t_1	S^1	$(a_{11}^1, \dots, a_{1k}^1)$	$(a_{21}^1, \dots, a_{2k}^1)$	$\cdots$	$(a_{N1}^1, \dots, a_{Nk}^1)$
t_2	S^2	$(a_{11}^2, \dots, a_{1k}^2)$	$(a_{21}^2, \dots, a_{2k}^2)$	$\cdots$	$(a_{N1}^2, \dots, a_{Nk}^2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_{M-1}	S^{M-1}	$(a_{11}^{M-1}, \dots, a_{1k}^{M-1})$	$(a_{21}^{M-1}, \dots, a_{2k}^{M-1})$	$\cdots$	$(a_{N1}^{M-1}, \dots, a_{Nk}^{M-1})$
$\uparrow$					
t_M	S^M	$(a_{11}^M, \dots, a_{1k}^M)$	$(a_{21}^M, \dots, a_{2k}^M)$	$\cdots$	$(a_{N1}^M, \dots, a_{Nk}^M)$

Fig. 1. Arrival of a State Set S^M

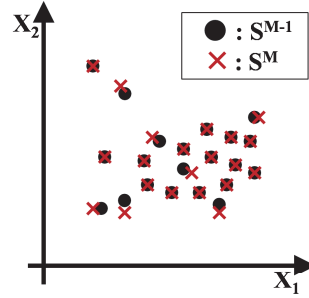


Fig. 2. Change of Data Distribution

different from other sensors, to monitor stock price movements of individual companies and detect outlier companies whose stock prices change very differently from those of other companies, or to monitor the location of moving objects and detect outlier objects that are distantly-positioned from other objects.

When we have a set of objects that change state (such as observation values, internal states, or locations) over time, and if object O_i 's state differs greatly from other objects' states at a given time, we can regard O_i as an outlier at the time. In continuously monitoring of a set of objects in which the state changes over time, we need to detect such outlier objects continuously.

Formally, when we have N objects whose state changes over time, let the set of objects' state at time t_j , $S^j = \{s_1^j, \dots, s_N^j\}$, where each state tuple $s_i^j = (a_{i1}^j, \dots, a_{ik}^j)$ describes the state of object O_i at t_j . If the state of O_i , that is s_i^j , differs greatly from that of other objects, then we regard O_i as an outlier at t_j .

We look at the problem of detecting outlier objects at current time t_M , where S^M arrives continuously as data streams $(S^1, \dots, S^M)$, as shown in Fig. 1.

A straightforward solution to the problem is to apply the existing approach for static data, at every arrival of S^M . However, the yield of such an approach is too voluminous. Therefore, repeating the process for every time stamp is inefficient.

In general, a state set S^M is the change of the last state set S^{M-1} . In many cases, therefore, the differences between them are not great, as shown in Fig. 2. In cases like this, we can detect outliers effectively in S^M based on differential calculation using the result of outlier detection for S^{M-1} .

This paper proposes continuous outlier detection over data streams based on distance-based outlier detection (DB-Outlier), which is the basic approach to outlier detection. Experiments confirm effectiveness of the proposed approach. The rest of the paper is organized as follows: Section 2 mentions related work on outlier detection. Section 3 defines DB-Outliers and CDB-Outliers. Section 4 describes existing DB-Outlier detection methods as a preliminary to our proposal, and Section 5 explains continuous CDB-Outlier detection, our proposed method. Section 6 evaluates the effectiveness and efficiency of the algorithm compared with existing algorithms. Section 7 presents conclusions and future work.

2 Related Work

Various outlier detection methods have been proposed. This section introduces existing work on outlier detection over static data and stream data.

2.1 Outlier Detection over Static Data

The major methods of outlier detection over static data are as follows:

Statistical-based approaches [1,2] assume that the dataset follows a statistical model. With these method we detect objects that deviate from the model as outliers. However, statistical approaches make a lot of assumptions (e.g., distribution model), and have difficulty dealing with high dimensional datasets.

Clustering approaches (e.g., CLARANS [5], DBSCAN [6], BIRCH [7], Wave Cluster [8], CLIQUE [9]) detect outliers as by-products. In most cases, the main objective is to find clusters in the dataset. For that reason, this method does not focus on outlier detection.

The density-based approach [10] adopts a Local Outlier Factor (LOF) which represents the local density of each data point's neighborhood and declares the degree of outlierness. The method detects objects having a high LOF as outliers.

The distance-based approach [3,4] is a simpler and more common approach. It merely calculates the distance between two data points; distance is a common notion of many knowledge and technical areas. Effectiveness and importance are shown in [3,4].

2.2 Outlier Detection over Stream Data

Because of increased interest, varied research is underway on data streams. The same is true for outlier detection. L. Su et al. propose outlier detection on distributed data streams [11] based on their original outlier model. K. Yamanishi et al. propose on-line outlier detection using statistical models [12,13].

Compared with these approaches, our proposal is simpler and more versatile because we use the distance-based approach. Moreover, our proposed approach features efficiency based on differential processing.

3 Definition of Outliers

3.1 DB-Outlier

A DB-Outlier is defined by E. M. Knorr et al. as follows [4]:

Definition 1. An object O_i in a dataset S is a $DB(p, D)$ -outlier if at least fraction p of the objects in S lie greater than distance D from O_i .

For an object O_i , the D -neighborhood of O_i contains the set of objects $O_q \in S$, where $d(O_i, O_q) \leq D$. Note that $d(O_i, O_q)$ denotes the distance between O_i and O_q . To simplify the discussion, we use another parameter $M[M = N(1 - p)$, N : data size of S], which is the maximum number of objects within the D -neighborhood of an outlier. k represents the dimension of S .

To clarify the definition, we show an example in Fig. 3. We have 30 objects ($N=30$) and parameter $p = 0.9$. Thus, if an object has at most $M[= 30(1 - 0.9) = 3]$ objects in its D -neighborhood, which is described as circles, then the object is a DB-Outlier. The left object has $2(\leq M)$ objects in its D -neighborhood, so it is a DB-Outlier. On the other hand, the right object is a non-outlier, because it has $12(> M)$ objects in its D -neighborhood.

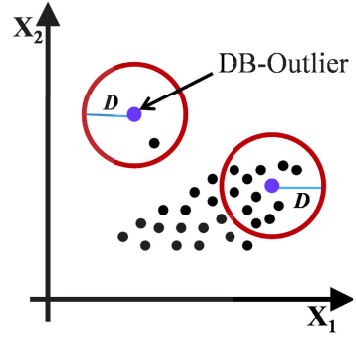


Fig. 3. DB-Outlier, $k = 2$, $p = 0.9$, $N = 30$

3.2 CDB-Outlier

In this paper, we call the current DB-Outlier in a data stream “CDB-Outlier.” A data stream can be described as a state set $S^j (1 \leq j \leq M)$, which is illustrated in Fig. 1. We then provide the definition of CDB-Outlier in Definition 2.

Definition 2. An object O_i is a DB-Outlier at time t_j if tuple s_i^j of O_i is a DB-Outlier in state set S^j . We call the DB-Outlier at current time t_M **CDB-Outlier**.

4 Algorithms for DB-Outlier Detection

This section presents DB-Outlier detection algorithms for static data [4]. We first explain the Simple algorithm. We then show the Cell-Based algorithm for the quick processing which is used in our proposal.

4.1 Simple Algorithm

This algorithm merely follows the definition of DB-Outlier. It applies the following processes for all objects in dataset S .

For each object O_i , we calculate the distance between O_i and other objects. Once there are more than M objects in the D -neighborhood, we stop the search and declare O_i as a non-outlier. Otherwise, we report O_i as an outlier.

The main drawback is time complexity $O(kN^2)$. This occurs because distance is calculated every 2 points until the object has been judged.

4.2 Cell-Based Algorithm

To skip distance calculations, the Cell-Based algorithm [4] employs a cell structure on a data space.

Cell Structure. O_i ’s tuple $s_i^j = (a_{i1}^j, \dots, a_{ik}^j)$ can be coded as a point in k -dimensional space with axes of $X_1, \dots, X_k$. We divide this k -dimensional space into cells whose diagonal is $\frac{D}{2}$ length (length of the side $l = \frac{D}{2\sqrt{k}}$). Let $C_{x_1, \dots, x_k}$ describe the cell with x_1 -th index of X_1 axis, ..., x_k -th index of X_k axis.

We then define L_1 neighbors and L_2 neighbors of $C_{x_1, \dots, x_k}$.

Definition 3. The L_1 neighbors of $C_{x_1, \dots, x_k}$, $L_1(C_{x_1, \dots, x_k})$ are the immediately neighboring cells of $C_{x_1, \dots, x_k}$, defined as follows,

$$L_1(C_{x_1, \dots, x_k}) = \{C_{u_1, \dots, u_k} \mid |u_i - x_i| \leq 1 \ (1 \leq i \leq k) \wedge C_{u_1, \dots, u_k} \neq C_{x_1, \dots, x_k}\}.$$

Definition 4. The L_2 neighbors of $C_{x_1, \dots, x_k}$, $L_2(C_{x_1, \dots, x_k})$, are cells that satisfy the following formula,

$$L_2(C_{x_1, \dots, x_k}) = \{C_{u_1, \dots, u_k} \mid |u_i - x_i| \leq \lceil 2\sqrt{k} \rceil \ (1 \leq i \leq k) \wedge C_{u_1, \dots, u_k} \notin L_1(C_{x_1, \dots, x_k}) \wedge C_{u_1, \dots, u_k} \neq C_{x_1, \dots, x_k}\}.$$

Properties A

- (A1) If $O_i \in C_{x_1, \dots, x_k}$, $O_p \in C_{x_1, \dots, x_k}$, then $d(O_i, O_p) \leq \frac{D}{2}$.
- (A2) If $O_i \in C_{x_1, \dots, x_k}$, $C_{u_1, \dots, u_k} \in L_1(C_{x_1, \dots, x_k})$, and $O_q \in C_{u_1, \dots, u_k}$, then $d(O_i, O_q) \leq D$.
- (A3) If $O_i \in C_{x_1, \dots, x_k}$, $C_{u_1, \dots, u_k} \notin L_1(C_{x_1, \dots, x_k})$, $C_{u_1, \dots, u_k} \notin L_2(C_{x_1, \dots, x_k})$, $C_{u_1, \dots, u_k} \neq C_{x_1, \dots, x_k}$, and $O_r \in C_{u_1, \dots, u_k}$, then $d(O_i, O_r) > D$.

As shown in Fig. 4, it is obvious that property (A1) is met. Fig. 5 illustrates property (A2). $d(O_i, O_q)$ is max when the two objects are located as shown in this figure. Fig. 6 illustrates property (A3). $d(O_i, O_r)$ is minimum when the two objects are located as shown in this figure.

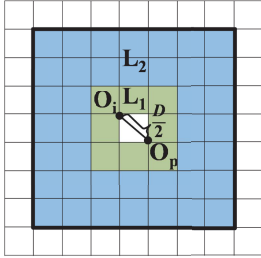


Fig. 4. Property (A1)

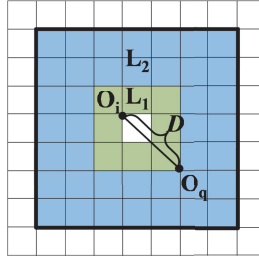


Fig. 5. Property (A2)

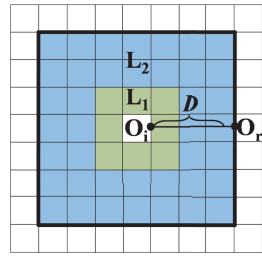


Fig. 6. Property (A3)

Let n , n_1 , and n_2 be the numbers of objects in $C_{x_1, \dots, x_k}$, $L_1(C_{x_1, \dots, x_k})$, and $L_2(C_{x_1, \dots, x_k})$, respectively. Then we derive properties B from properties A as follows:

Properties B

- (B1) If $n > M$, $C_{x_1, \dots, x_k}$ contains no DB-outliers.
- (B2) If $n + n_1 > M$, $C_{x_1, \dots, x_k}$ contains no DB-outliers.
- (B3) If $n + n_1 + n_2 \leq M$, all objects in $C_{x_1, \dots, x_k}$ are DB-outliers.

We color cells that satisfy (B1) as *red*, (B2) as *pink*, and (B3) as *yellow*. We can detect outliers in those cells without any distance calculation. Then we identify nonempty and uncolored cells as *white*. The decision for cell colors is called **CCD** (Cell-Color Decision). And the algorithm with CCD is as follows:

1. **For each** cell $C_q, n_q \leftarrow 0$
2. **For each** object O_i , do:
Map O_i to an appropriate cell C_q , store O_i , and increment n_q .
3. **For each** cell C_q , if $n_q > M$, label C_q *red*.
4. **For each red** cell C_r , do:
Label each of the L_1 neighbors of C_r *pink*, provided the neighbor has not already been labeled *red*.
5. **For each** non-empty uncolored cell C_w , do:
 - a. $n_{1w} \leftarrow \sum_{i \in L_1(C_w)} n_i$
 - b. **If** $n_w + n_{1w} > M$, label C_w *pink*.
 - c. **Else**
 - i. $n_{2w} \leftarrow \sum_{i \in L_2(C_w)} n_i$
 - ii. **If** $n_w + n_{1w} + n_{2w} \leq M$, label C_w *yellow*, mark all objects in C_w as outliers.
 - iii. **Else**
 1. Label C_w as *white*.
 2. **For each** object $O_i \in C_w$, do:
 - a. $Count_i \leftarrow n_w + n_{1w}$
 - b. **For each** object $O_q \in C_{L_2}, C_{L_2} \in L_2(C_w)$, **if** ($dist(O_i, O_q) \leq D$):
Increment $Count_i$ by 1. **If** $Count_i > M$, O_i cannot be an outlier, so goto(5cii2)
 - c. Mark O_i as an outlier

Fig. 7. Cell-Based Algorithm

Algorithm. Fig. 7 illustrates the Cell-Based algorithm. Step 2 quantizes each object to its appropriate cell. Step 3 labels all cells containing more than M objects, *red* (Property (B1)). Step 4 labels uncolored cells that have *red* cells in L_1 neighbors, *pink* (Property (B2)). Other cells satisfying Property (B2) are labeled *pink* in step 5b. Step 5cii labels cells satisfying Property (B3) as *yellow*, and reports all objects in the cells as DB-Outliers. Finally, uncolored cells (not satisfying Properties (B1), (B2), (B3)) are labeled as *white* in step 5ciii. CCD is a set of processes shown in step 3-5ciii1. We then operate only objects in *white* cells (C_w) using an object-by-object process as follows: For each object $O_i \in C_w$, we calculate distance between O_i and each object O_q in cells $\in L_2(C_w)$, and count the number of objects in its D -neighborhood. We count $n + n_1$ in $Count_i$ in advance because all L_1 neighbors are always within the D -neighborhood. Once the number of objects in the D -neighborhood exceeds M , we declare O_i a non-outlier. If the count remains less than or equal to M after all calculation, we report O_i as an outlier. We call this process for each *white* cell (5ciii2) **WCP** (White-Cell Process), and the process for each object in a *white* cell (5ciii2a-c) **WOP** (White-Object Process).

Cell-by-cell basis decisions using Properties B help to determine whether or not an object in the cell is an outlier, and this leads to skipping a lot of distance calculations and reducing the process time compared with the Simple algorithm.

5 Proposed Method

This section describes continuous CDB-Outlier detection over data streams. That means we consider the detection of DB-Outliers when a state set S^M arrives, as shown in Fig. 1.

The straightforward approach to detecting CDB-Outliers is to process DB-Outlier detection for every state set S^M . However, in most cases, the data distribution of S^M is similar to that of S^{M-1} , so processing for all objects in S^M again has many meaningless calculations. Therefore, our proposed method does differential processing based on the change between S^{M-1} and S^M using the idea of the Cell-Based algorithm.

5.1 Assumptions

We do differential processing with objects whose states changed from t_{M-1} to t_M (say **SC-Objects**(State-Change Objects)) and their current state sets $\Delta^M (= \{(O_i, s_i^M) \mid 1 \leq i \leq N \wedge s_i^{M-1} \neq s_i^M\})$. Our proposed method targets all state sets except the initial state set (S^1) of data streams. We use the original Cell-Based algorithm for S^1 .

5.2 Differential Processing on an SC-Object

To simplify the problem, we consider the case with one SC-Object, O_P . There are two ways to change the state in the state space divided into cells. Let " $O_i \in C_{x_1, \dots, x_k}^j$ " represent that cell $C_{x_1, \dots, x_k}$ contains O_i at t_j .

[1] Move to a different cell:

$$O_P \in C_{x_1, \dots, x_k}^{M-1}, O_P \in C_{x_1, \dots, x_k}^M, \\ C_{x_1, \dots, x_k}^{M-1} \neq C_{x_1, \dots, x_k}^M.$$

O_P influences cells at t_{M-1} and t_M and their L_1 and L_2 neighbors, which is described as colored area in Fig. 8. Since n of $C_{x_1, \dots, x_k}^{M-1}$ and $C_{x_1, \dots, x_k}^M$ change and influence those cells.

[2] Move in the same cell:

$$O_P \in C_{x_1, \dots, x_k}^{M-1}, O_P \in C_{x_1, \dots, x_k}^M, C_{x_1, \dots, x_k}^{M-1} = C_{x_1, \dots, x_k}^M.$$

Because n of $C_{x_1, \dots, x_k}^M$ does not change, *red*, *pink*, and *yellow* cells within L_2 neighbor are not influenced. It influences only *white* cells within the L_2 neighbor. There are two cases:

[2a] There are *white* cells in $L_2(C_{x_1, \dots, x_k}^M)$.

[2b] $C_{x_1, \dots, x_k}^M$ itself is a *white* cell.

Fig. 9 illustrates the case [2a], and C_w represents a *white* cell in $L_2(C_{x_1, \dots, x_k}^M)$. $O_q \in C_w$ can be influenced because O_P can enter or exit O_q 's D -neighborhood. In case [2b], as shown in Fig. 10, the area of O_P 's D -neighborhood will change, and O_P influences outlier decision of O_P .

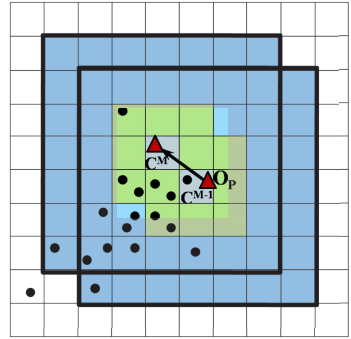


Fig. 8. Case[1]

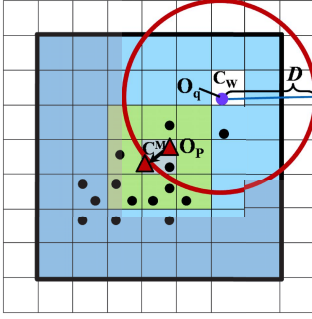


Fig. 9. Case[2a]

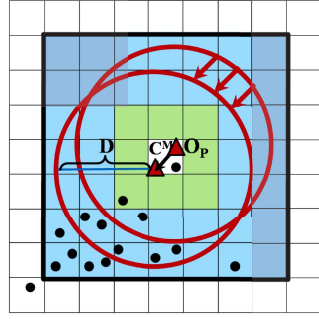


Fig. 10. Case[2b]

5.3 Target Cells for Re-outlier Detection

Actually, there are more than one SC-Objects from t_{M-1} to t_M . Therefore, we expand the above idea to more than one SC-Objects. We identify the target cells to reprocess, and classify them taking into account overlaps of each SC-Object's process. For example, the area of influence for case [1] contains that of case [2]. Targeted cells are the following 4 types.

Type A: Cells containing SC-Objects which have moved to or from another cell, at t_M (C^{M-1} and C^M in Fig. 8), namely

$$TypeA = \{C_{x_1, \dots, x_k} \mid (1 \leq i \leq N) \wedge ((O_i \in C_{x_1, \dots, x_k}^{M-1} \wedge O_i \notin C_{x_1, \dots, x_k}^M) \vee (O_i \notin C_{x_1, \dots, x_k}^{M-1} \wedge O_i \in C_{x_1, \dots, x_k}^M))\}.$$

Type B: Cells of L_1 and L_2 neighbors of Type A (Colored cells, in Fig. 8), except for those classified as Type A cells, namely

$$TypeB = \{C_{x_1, \dots, x_k} \mid (C_{u_1, \dots, u_k} \in L_1(C_{x_1, \dots, x_k}) \wedge C_{u_1, \dots, u_k} \in TypeA) \vee (C_{u_1, \dots, u_k} \in L_2(C_{x_1, \dots, x_k}) \wedge C_{u_1, \dots, u_k} \in TypeA) \wedge C_{x_1, \dots, x_k} \notin TypeA)\}.$$

Type C: *White* cells that are in the L_2 neighbor of the cells that contain SC-Objects having moved within the same cell (C_w in Fig. 9), except for those classified as Type A and B cells, namely

$$TypeC = \{C_{x_1, \dots, x_k} \mid C_{u_1, \dots, u_k} \in L_2(C_{x_1, \dots, x_k}) \wedge (1 \leq i \leq N) \wedge (O_i \in C_{u_1, \dots, u_k}^{M-1} \wedge O_i \in C_{u_1, \dots, u_k}^M \wedge C_{u_1, \dots, u_k}^{M-1} = C_{u_1, \dots, u_k}^M \wedge s_i^{M-1} \neq s_i^M) \wedge color(C_{x_1, \dots, x_k}) = white \wedge C_{x_1, \dots, x_k} \notin TypeA \cup TypeB\}.$$

Type D: *White* cells that contain SC-Objects that have moved within the same cell (C^M in Fig. 10), except for those classified as Type A, B and C cells, namely

$$TypeD = \{C_{x_1, \dots, x_k} \mid (1 \leq i \leq N) \wedge (O_i \in C_{x_1, \dots, x_k}^{M-1} \wedge O_i \in C_{x_1, \dots, x_k}^M \wedge C_{x_1, \dots, x_k}^{M-1} = C_{x_1, \dots, x_k}^M \wedge s_i^{M-1} \neq s_i^M) \wedge color(C_{x_1, \dots, x_k}) = white \wedge C_{x_1, \dots, x_k} \notin TypeA \cup TypeB \cup TypeC\}.$$

5.4 Algorithm

We only process the target cells with the proposed algorithm, shown in Fig. 11.

The input is a set of SC-Objects and its state at t_M , $\Delta^M (= \{(O_i, s_i^M) \mid (1 \leq i \leq N) \wedge s_i^{M-1} \neq s_i^M\})$. The output is CDB-Outliers.

Step 1 updates cells that contain SC-Objects at time t_M . If the cell C^M of O_P at t_M differs from C^{M-1} at t_{M-1} , then step 1a labels both cells, A(Case [1]). If there are *white* cells $C_\omega \in L_2(C^M)$, then step 1bi labels C_ω , C(Case [2a]). If the color of C^M is *white*, then step 1bii labels C^M , D(Case [2b]). Step 2 labels cells in L_1 and L_2 neighbor of Type A cells, B, and updates their n_1 and n_2 . We have now targeted the cells to be reprocessed. We then classify the targeted cells based on labels A, B, C and D. Cells labeled A are in Type A, cells labeled B except for Type A cells are in Type B, cells labeled C except for Types A and B cells are in Type C, cells labeled D except for Types A, B, and C cells are in Type D. Step 3 does Re-CCD as shown in Fig. 12. Re-CCD differs from CCD because it does not count objects in cells of L_1 and L_2 neighbors, and uses only existing or updated n , n_1 and n_2 . It also leads to reduced processing. Step 4 processes WCP, mentioned in the original Cell-Based algorithm (4.2), over Type C cells. Step 5 processes WOP over SC-Objects in Type D cells. The objects in *yellow* cells and outlier objects in *white* cells are output as CDB-Outliers.

6 Experiments and Results

Experiments with 2-dimensional (2-D) synthetic data and 3-dimensional (3-D) real data confirm improved processing time for the proposed method.

All of our tests were run on a Microsoft Windows Vista machine with an AMD Athlon(tm) 64 2 Dual Core Processor 3800+ 2GHz CPU and 1982MB of main memory. We implemented the software with Java 1.6.0_02.

This section explains the comparative approaches, describes the datasets, and provides the results and discussions.

6.1 Comparative Approaches

We compared our proposed method with two DB-Outlier methods.

CM(Cell-Based Method): A method that executes the Cell-Based algorithm for every time stamp.

SM(Simple Method): A method that executes the minimal process of the Simple algorithm with Δ^M . It processes only SC-Objects and the neighbor objects that can be influenced by SC-Objects.

Applying the original Simple algorithm for every time stamp takes huge computation time and is beyond comparison. Hence, we do not show the comparison.

Input: Δ^M
Output: CDB-Outliers

1. **For each** $(O_P, s_P^M) \in \Delta^M$, do: identify cell C^M at t_M
 - a. **If** $(C^M \neq C^{M-1})$:
 - i. Store O_P in C^M , and update n of C^M .
 - ii. Delete O_P from C^{M-1} , and update n of C^{M-1} .
 - iii. Label C^M and C^{M-1} , A.
 - b. **Else**:
 - i. **If** there are *white* cells (C_ω) in $L_2(C^M)$, label C_ω C.
 - ii. **If** $\text{color}(C^M) = \text{white}$, label C^M D.
2. **For each** cell C_A of Type A, do:
 - a. **For each** cell $C_{L1} \in L_1(C^M)$, do:
 - i. Update n_1 .
 - ii. Label C_{L1} B.
 - b. **For each** cell $C_{L2} \in L_2(C^M)$, do:
 - i. Update n_2 .
 - ii. Label C_{L2} B.
3. **For each** cell C_{AB} of Type A or B, do: Re-CCD.
4. **For each** cell C_C of Type C, do: WCP.
5. **For each** cell C_D of Type D, do:
 - a. **For each** SC-Object $O_P \in C_D$, do: WOP.

Fig. 11. Proposed Algorithm ($t_{M-1} \rightarrow t_M$)

1. **If** $n > M$, color *red*.
2. **Else if** $n + n_1 > M$, color *pink*.
3. **Else if** $n + n_1 + n_2 \leq M$, color *yellow*.
4. **Else**, do: WCP, color *white*.

Fig. 12. Re-CCD

6.2 Datasets

MO Data (2-D). We use moving objects' data streams (MO data) generated by the Mobi-REAL¹ simulator. Fig. 13 describes the distribution of moving objects at a given time, and each point represents each moving object. We set passes for moving objects: dense at the skirt area and sparse at the central area. Therefore, objects passing the central area are detected as CDB-Outliers. We use the x and y coordinates for each object per time unit (1sec) as 2-D datasets. The area of this simulation is $700 \times 700[m^2]$. Parameters are: the number of objects is 10000, the (SC-Objects)/(all objects) per time unit is 50[%], the average of moved distance of the SC-Objects per time unit is 1.5[m], $D = 15[m]$, and $p = 0.9995$.

Stock Data (3-D). We use daily stock dataset (Dec., 2007, Japan) [14] as a 3-D real dataset. We detect brands that behave very differently from others. This dataset consists of the band, closing stock price, and completed amount

¹ <http://www.mobireal.net/index.html>

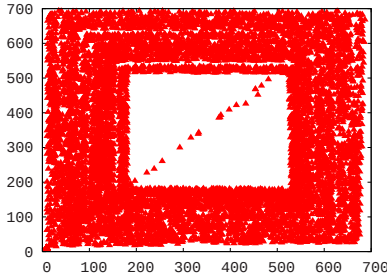
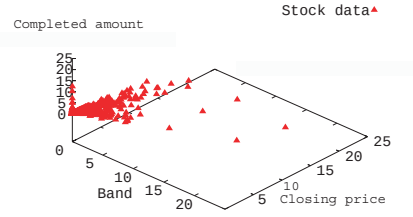


Fig. 13. MO Data


 Fig. 14. Stock Data ($\times 10^5$)

and changes per day. To reduce the non-uniformity of each attribute, we assign weights as $\text{band} \times 10^3$, $(\text{closing stock price}) \times 10^2$. Fig. 14 shows data distribution at a given time. The data has 4039 objects. The average of $(\text{SC-Objects})/(\text{all objects})$ per time unit is 83% .

6.3 Results and Discussions

We compared the processing time for the proposed method (PM) and comparative method (CM, SM), and evaluated the relationship with each parameter.

MO data (2-D). Figs. 15-20 shows results of MO data. In 2-D cases, SM takes too long (around 30000(ms)) to process a dataset per 1000(ms), so we can say that SM is not appropriate at all. Thus, we evaluate only CM and PM in 2-D experiments. Fig. 15 shows time versus the ratio of SC-Objects in all objects per second. Even if the ratio of SC-Objects reaches 100%, PM takes less time than CM, about 70% of CM. This occurs because, even if all objects change in state, not all objects move to different cells. Further, CM counts objects in L_1 and L_2 neighbors every time. Fig. 16 shows the relationship between time and the number of all objects. Both methods take more time as the number increases. PM exceeds CM in all cases. Fig. 17 shows time versus the average of moved distance of SC-Objects. With this result, we cannot see clear relation to each other. Fig. 18 illustrates the correlation between time and p . CM and PM shorten time as p grows. When p is large, the difference between CM and PM becomes large. In Fig. 19, we change D . At around $D = 30$, PM time approximately equals to CM. This occurs because, as shown in Fig. 20, the sum of reprocessed cells in PM is almost equal to the sum of all cells around $D = 30$. Actually, there are no CDB-Outliers for $D > 25$.

From the results, in many cases, PM is effective for continuous processing in 2-D datasets. However, if reprocessed cells in PM increase, PM loses its advantage.

Stock Data (3-D). Fig. 21 shows the relationship between time and D , where $p = 0.997$. It is obvious that CM is profoundly affected by D , and if we have

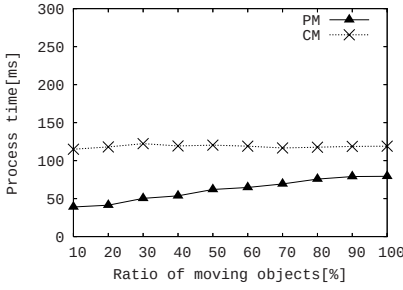


Fig. 15. MO Data: Time versus SC-Objects/All Objects

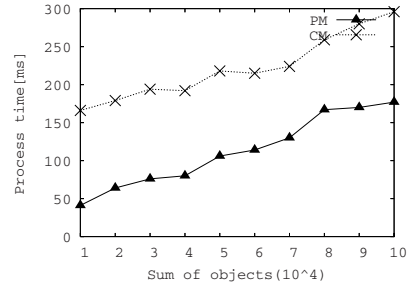


Fig. 16. MO Data: Time versus the Number of All Objects

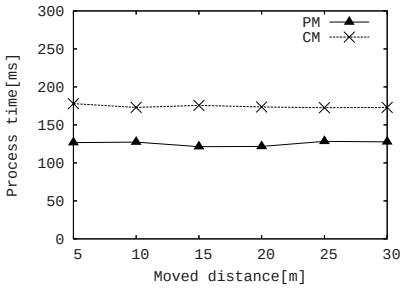


Fig. 17. MO Data: Time versus the Average Moved Distance

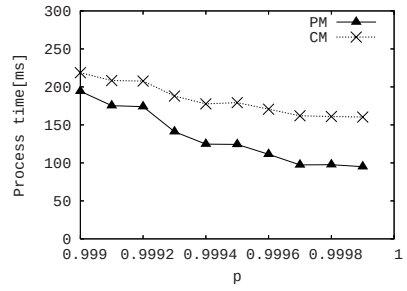


Fig. 18. MO Data: Time versus p

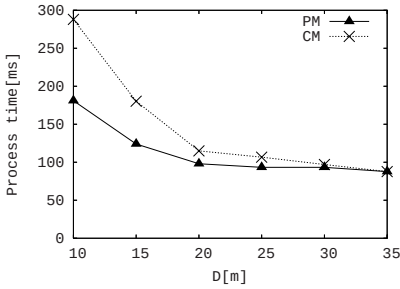


Fig. 19. MO Data: Time versus D

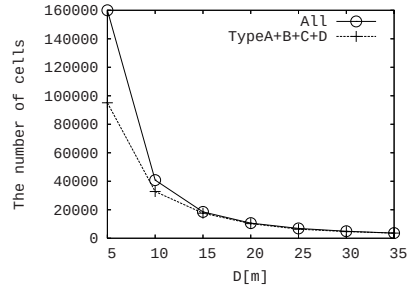
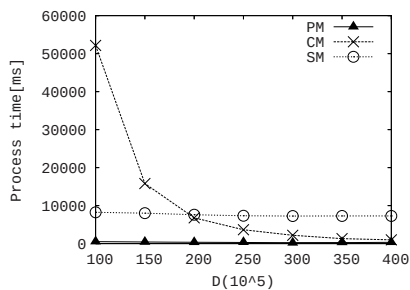
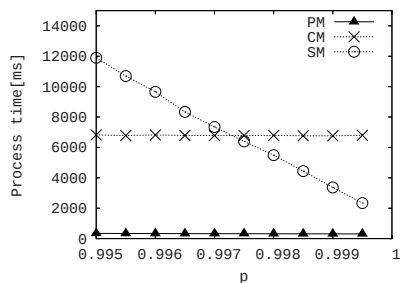


Fig. 20. MO Data: Number of Re-processed Cells versus D

small D , the process time is very long. With 3-D data, the number of all cells is bigger than that of 2-D data. Moreover, the number of cells in L_1 and L_2 neighbors is also bigger. Therefore the calculation cost is higher. On the other hand, PM is little influenced by D , since we use differential calculation. Fig. 22 is the relationship between process time and p , where $D = 200 \times 10^5$. SM is profoundly affected by p , because the smaller the p , the bigger the M . Therefore, distance


 Fig. 21. Stock data: Time versus D

 Fig. 22. Stock Data: Time versus p

calculations until “the object is not an outlier” increase. On the other hand, PM and CM are not significantly influenced because of cell-by-cell decisions.

PM can maintain the advantage of the cell-by-cell process with 3-D datasets. Therefore, PM takes shorter than CM and SM. With those results, we can say PM is also effective for 3-D datasets.

7 Conclusions and Future Work

In this paper, we have proposed continuous outlier detection over data streams. We employ DB-Outlier, and based on the Cell-Based algorithm for quick processing, we provide an effective algorithm using differential processing over data streams. We evaluated our proposed method with synthetic and real datasets, and showed its advantage over naive methods. Extensions to cope with high dimensional data and dynamic data streams are interesting future research issues.

Acknowledgment. This research has been supported in part by the Grant-in-Aid for Scientific Research from MEXT (# 19024006).

References

1. Barret, V., Lewis, T.: Outliers in Statistical Data. Wiley, Chichester (2001)
2. Eskin, E.: Anomaly Detection over Noisy Data using Learned Probability Distributions. In: ICML, pp. 255–262 (2000)
3. Knorr, E.M., Ng, R.T.: Finding Intensional Knowledge of Distance-Based Outliers. In: VLDB, pp. 211–222 (1999)
4. Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-Based Outliers: Algorithms and Applications. VLDB J. 8(3-4), 237–253 (2000)
5. Ng, R.T., Han, J.: Efficient and Effective Clustering Methods for Spatial Data Mining. In: VLDB, pp. 144–155 (1994)
6. Ester, M., Kriegel, H.P., Xu, X.: A Database Interface for Clustering in Large Spatial Databases. In: KDD, pp. 94–99 (1995)
7. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: SIGMOD, pp. 103–114 (1996)

8. Sheikholslami, G., Chatterjee, S., Zhang, A.: WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In: VLDB, pp. 428–439 (1998)
9. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: SIGMOD, pp. 94–105 (1998)
10. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying Density-Based Local Outliers. In: SIGMOD, pp. 93–104 (2000)
11. Su, L., Han, W., Yang, S., Zou, P., Jia, Y.: Continuous Adaptive Outlier Detection on Distributed Data Streams. In: HPCC, pp. 74–85 (2007)
12. Yamanishi, K., Takeuchi, J., Williams, G., Milne, P.: On-line Unsupervised Outlier Detection using Finite Mixtures with Discounting Learning Algorithms. In: KDD, pp. 320–324 (2000)
13. Yamanishi, K., Takeuchi, J.: Discovering Outlier Filtering Rules from Unlabeled Data: Combining a Supervised Learner with an Unsupervised Learner. In: KDD, pp. 389–394 (2001)
14. ITicker, <http://homepage1.nifty.com/hdatelier/>

Component Selection to Optimize Distance Function Learning in Complex Scientific Data Sets

Aparna Varde¹, Stephen Bique², Elke Rundensteiner³, David Brown^{3,4}, Jianyu Liang⁴, Richard Sisson^{4,5}, Ehsan Sheybani⁶, and Brian Sayre⁷

¹ Department of Math and Computer Science, Virginia State University, Petersburg, VA

² Naval Research Laboratory, Washington, DC

³ Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA

⁴ Department of Mechanical Engineering, Worcester Polytechnic Institute, Worcester, MA

⁵ Center for Heat Treating Excellence, Metal Processing Institute, Worcester, MA

⁶ Department of Engineering and Technology, Virginia State University, Petersburg, VA

⁷ Department of Biology, Virginia State University, Petersburg, VA
{avarde@vsu.edu, stephen.bique@nrl.navy.mil,
rundenst@cs.wpi.edu, dcb@cs.wpi.edu, jianyul@wpi.edu,
sisson@wpi.edu, esheyban@vsu.edu, bsayre@vsu.edu}

Abstract. Analyzing complex scientific data, e.g., graphs and images, often requires comparison of features: regions on graphs, visual aspects of images and related metadata, some features being relatively more important. The notion of similarity for comparison is typically distance between data objects which could be expressed as distance between features. We refer to distance based on each feature as a component. Weights of components representing relative importance of features could be learned using distance function learning algorithms. However, it is seldom known which components optimize learning, given criteria such as accuracy, efficiency and simplicity. This is the problem we address. We propose and theoretically compare four component selection approaches: Maximal Path Traversal, Minimal Path Traversal, Maximal Path Traversal with Pruning and Minimal Path Traversal with Pruning. Experimental evaluation is conducted using real data from Materials Science, Nanotechnology and Bioinformatics. A trademarked software tool is developed as a highlight of this work.

Keywords: Feature Selection, Data Mining, Multimedia, Scientific Analysis.

1 Introduction

Background. Performing analysis over complex data such as graphs and images involves numerous challenges [1, 4, 5, 8, 12, 15, 18, 23]. These pertain to factors such as nature of the data, semantics of the domain and requirements of specific problems. In our work, the focus is on graphical and image data from scientific domains [16, 17, 6]. We deal with 2-dimensional graphs plotting results of scientific experiments as a dependent versus an independent variable [16]. Our images are cross-sections of samples taken under a microscope at various stages of scientific processes [17, 6]. Analyzing such graphs and images using techniques such as clustering, helps draw conclusions about the concerned scientific processes. For example, if two graphs are in the same cluster, the processes that led to them can be considered similar.

Motivation. In order to capture subtleties of data, comparison of features is often needed. Some features on graphs could be regions depicting critical phenomena. There could be statistical observations on graphs showing behavior of process parameters. Absolute position of points on graphs could be important to distinguish between processes [16]. Images have visual features such as particle size, image depth, color and greyscale, and metadata features such as image source, location and type of microscope [17, 6]. If a sample was taken with a transmission electron microscope that penetrates through inner locations as opposed to a scanning electron microscope that observes surfaces, the corresponding observations have very different connotations [17, 6]. Thus a metadata feature such as type of microscope could be more important than a visual feature such as color. It is therefore important to make feature-based comparisons. Experts at best have subjective criteria for comparison but do not have a distance function for computational analysis. Various distance functions exist in the literature [10, 22, 12, 4]. However, it is seldom known apriori which of these best fits a given data set. Thus, it is important to learn a notion of distance that incorporates the features of the data and their relative importance.

Problem Definition. Features of complex data can be depicted by individual distance functions, a weighted sum of which forms the distance function for the data. In our work, an individual distance function depicting a feature is called a component. Weights of components giving their relative importance can be learned using distance function learning algorithms [19, 20]. However, it is challenging to select appropriate components for learning. It is not feasible for experts to determine components to use among all applicable ones. Also, it is desirable to optimize learning given factors such as accuracy, efficiency and simplicity. Accuracy is the effectiveness of the distance in preserving semantics and hence its usefulness in data analysis. Efficiency is a time criterion. Faster learning is better with respect to resource consumption and reusability. Simplicity relates to number of components in the distance. It is desirable to have simpler distance functions for storage, indexing and retrieval. Thus the problem we address is the optimization of distance function learning.

Proposed Solution. We aim to select combinations of components in distance function learning, incorporating the optimization criteria of accuracy, efficiency and simplicity. Four component selection approaches are proposed.

- *Maximal Path Traversal (MaxPT):* Given that each path is a combination of components, this considers the maximum number of possible paths.
- *Minimal Path Traversal (MinPT):* This covers a minimum number of paths in component selection using a suitable stopping criterion.
- *Maximal Path Traversal with Pruning (MaxPTP):* This prunes some components using a heuristic and traverses all the remaining paths.
- *Minimal Path Traversal with Pruning (MinPTP):* This uses a heuristic for pruning components and traverses a minimum number of remaining paths

Comparative Assessment. We compare the selection approaches theoretically by deriving their computational complexities. Experimental evaluation is conducted to assess the approaches with respect to accuracy, efficiency and simplicity. Real data from Materials Science, Bioinformatics and Nanotechnology is used for evaluation. Results are analyzed based on usefulness of selection approaches in applications.

Notable Application. A decision support system called AutoDomainMine [21] has been developed in the Heat Treating of Materials. This system uses MaxPTP to optimize feature selection in distance function learning over graphical data since it is found the most suitable for the given application based on our evaluation. The AutoDomainMine system is a trademarked software tool.

Contributions. The following contributions are made in this paper.

1. Proposing selection approaches to optimize distance function learning.
2. Deriving and comparing the computational complexity of the approaches.
3. Conducting evaluation with real data from three scientific domains.
4. Analyzing evaluation results in terms of usefulness in targeted applications.
5. Applying the MaxPTP approach to develop trademarked software.

2 Component Selection Approaches

In order to give the details of the approaches we first explain a few relevant concepts.

Distance Function: Let W_i be the weight depicting the relative importance of component C_i and let m be the total number of components. The distance function D for the given data is defined as $D = \sum_{i=1}^m W_i C_i$

Learning Algorithm: This refers to a supervised distance function learning algorithm L . Given a training set S of graphs / images depicting notion of correctness in the domain, and an initial distance function D , algorithm L learns the weights of the components, and outputs the learned distance function D . (Details are in Section 4.2).

Path: A path P is an arbitrary ordering of distinct component(s).

Accuracy: The accuracy α of a distance function is any valid measurement that characterizes its effectiveness in preserving semantics. The measure is specific for a particular domain.

Convergence: Learning algorithm L is said to converge if error in learning weights of components drops below a threshold ϵ , calculation of error being domain-specific.

For example, accuracy and error can be defined as success rate and failure rate in clustering respectively [22] using true positives, true negatives, false positives and false negatives given correct clusters.

We now describe the selection approaches. Let L be a learning algorithm, S be a training set of graphs / images, and m be all applicable components for the data set.

2.1 Maximal Path Traversal

The Maximal Path Traversal (MaxPT) approach explores the maximum possible paths in selecting components for distance function learning and returns the one corresponding to the highest accuracy.

MaxPTP selects components any order (top down, bottom up or a random order). The main requirement is that all paths must be considered. It uses each path, i.e., each combination of components as the initial distance function D in the learning algorithm L . The accuracy of the learned distance function is measured for each path. The one giving highest accuracy is finally output as the learned distance function D . Our MaxPT algorithm is presented next.

MaxPT: Maximal Path Traversal
Input: Training Set S , Learning Algorithm L Output: Learned distance function D

1. Identify all m applicable components for S
 2. $c = 1$
 3. While $c \leq m$
 - (a) Execute L with all possible combinations of components
in $D = \sum_{i=1}^c W_i C_i$ and record accuracy α
 - (b) $c = c + 1$
 4. Output D with maximum accuracy as learned distance function
-

Justification. MaxPT follows the Epicurean philosophy of experiencing everything to discover the best [13]. Hence, each path is traversed with the goal of finding an optimal solution in terms of accuracy.

2.2 Minimal Path Traversal

In Minimal Path Traversal (MinPT) the aim is to learn a distance function efficiently. Yet it aims to meet the basic needs of accuracy in the domain. MinPT proceeds top down using a *MinPT Heuristic* defined as follows.

MinPT Heuristic: If component C_i used alone as the distance function D gives higher accuracy than component C_j alone then C_i is preferred over C_j in selecting a combination of components in D .

MinPT uses each component C_i as the distance D in algorithm L and calculates accuracy over training set S . If for any C_i , L converges, this is a stopping criterion. It outputs the final D as the learned distance. Else, it considers the 2 best components (giving highest accuracies) as the initial distance and executes L . If convergence still does not occur, it continues with the best 3 components and so forth until L converges or all best combinations are considered. Our MinPT algorithm is presented next.

MinPT: Minimal Path Traversal
Input: Training Set S , Learning Algorithm L Output: Learned distance function D

1. Identify all m applicable components for S
 2. $i = 1$
 3. While $i \leq m$ and A has not yet converged
 - (a) Execute L with $D=C_i$, and record accuracy α
 - (b) $i = i + 1$
 4. If L has not yet converged
 - (a) $c = 2$
 - (b) While $c \leq m$ and L has not yet converged
 - (c) Execute L with $D = \sum_{i=1}^c W_i C_i$ where $C_1 \dots C_{cc}$ are
the c best components based on accuracy
 - (d) $c = c + 1$
 5. If L converges or maximum number of iterations is reached,
output D as learned distance function.
-

Justification. MinPT avoids paths that do not seem promising. If a path with the best and 2nd best components does not provide convergence, it is less likely that a path with the best and 3rd best components will. Hence, as a next step it considers the best, 2nd best and 3rd best components since this offers faster expectation of convergence.

2.3 Maximal Path Traversal with Pruning

This approach explores maximum possible paths in distance function learning after pruning some components based on a *Tolerance Limit Heuristic*.

Tolerance Limit Heuristic: The tolerance limit denoted as τ is defined as the lowest acceptable limit of accuracy. The tolerance limit heuristic states that if the accuracy of a component C_i is less than τ , then the component is not useful.

Tolerance limit is different from error threshold. If learning algorithm L has error threshold ϵ then if error drops below ϵ , L is said to converge. Threshold ϵ thus defines good performance. However, tolerance limit τ is a cut-off for the opposite extreme.

MaxPTP works as follows. It first considers each component C_i as distance D in learning algorithm L and records accuracy α over training set S . Using the *Tolerance Limit Heuristic*, it prunes components with accuracy α less than tolerance limit τ . For the remaining r components, it traverses all paths, with each combination as the initial distance in L . Among all the corresponding learned distance functions, the one giving the highest accuracy is returned as the output. Our MaxPTP algorithm is given below.

MaxPTP: Maximal Path Traversal with Pruning

Input: Training Set S , Learning Algorithm L , Tolerance Limit τ

Output: Learned distance function D

1. Identify all m applicable components for S
 2. $i = 1$
 3. While $i \leq m$ and L has not yet converged
 - (a) Execute L with $D=C_i$ and record accuracy α
 4. Prune out components with $\alpha < \tau$

Let r = number of remaining components
 5. $c = 2$
 6. While $c \leq r$
 - (a) Execute L with all possible combinations of components

in $D = \sum_{i=1}^c W_i C_i$ and record accuracy α
 - (b) $c = c + 1$
 7. Output D with maximum accuracy as learned distance function
-

Justification. MaxPTP is analogous to MaxPT in traversing other paths even after convergence occurs. This is done with a similar aim of attaining maximal accuracy with respect to any given learning algorithm L . However, MaxPTP tends to learn faster and favors fewer components by not exploring unwanted paths.

2.4 Minimal Path Traversal with Pruning

The component selection approach of Minimal Path Traversal with Pruning (MinPTP) presents an interesting twist to MinPT. While MinPT traverses the minimum number of paths, MinPTP tends to learn a distance function favoring the fewest components.

MinPTP first considers each component C_i as distance D in learning algorithm L and records its accuracy over training set S . If L converges for any component, this is a stopping criterion and that component alone is the learned distance function. If not, unwanted components are pruned using the tolerance limit heuristic. All components with accuracy α less than tolerance limit τ get pruned. MinPTP then proceeds using the remaining r components with accuracy greater than or equal to tolerance limit. It uses all paths with 2 components from among the r components in descending order of accuracies and continues with paths of 3, 4 components and so on till convergence occurs or all such paths are considered. Our MinPTP algorithm is given below.

MinPTP: Minimal Path Traversal with Pruning

Input: Training set S , Learning algorithm L , Tolerance limit τ

Output: Learned distance function D

1. Identify all m applicable components for S
 2. $i = 1$
 3. While $i \leq m$ and L has not yet converged
 - (b) Execute A with $D=C_i$ and record accuracy α
 - (c) $i = i + 1$
 4. If L has not yet converged, prune out components with $\alpha < \tau$

Let $r =$ number of remaining components

 - (a) $c = 2$
 - (b) While $c \leq r$ and L has not yet converged
 - (i) Execute A with combinations of c components in

$$D = \sum_{i=1}^c W_i C_i \text{ ordered from highest to lowest accuracy}$$
 - (ii) $c = c + 1$
 5. If L converges or maximum number of iterations is reached, output D as learned distance function
-

Justification. The main argument in MinPTP is that other paths need to be explored rather than directly choosing the path that seemingly gives the maximum accuracy. It favors combinations of fewer components thus differing from MinPT.

3 Computational Complexity

3.1 Derivation of Complexity

Complexity of MaxPT. We consider top down traversal. For the first m executions of the learning algorithm, each component is considered individually. Then all paths with 2 components are considered. From combinatorics, number of combinations of 2 components that can be made from m components is $\frac{m!}{2!(m-2)!}$ denoted as ${}^mC_2$.

Thus, this approach has ${}^mC_2$ combinations with 2 components. Next, it considers paths with 3 components, i.e., ${}^mC_3$ combinations. Each combination is selected as the initial distance function in the learning algorithm and hence one combination corresponds to one execution of the algorithm. Thus, the total number of executions in MaxPT is $O({}^mC_1 + {}^mC_2 + \dots + {}^mC_{(m-1)} + {}^mC_m)$ which denotes its computational complexity. This can be simplified and expressed as $O(2^m - 1)$.

Complexity of MinPT. The MinPT approach first executes the learning algorithm m times to find the individual accuracies given by the respective components. In every subsequent execution it considers only the best combinations of components.

Hence, in addition to the m executions with single components, it involves at most $m-1$ executions of the learning algorithm. Thus, the total number of executions can be at most $m + m-1 = 2m-1$ which is of the order of m . MinPT therefore in the average case gives a computational complexity of $O(m)$.

Complexity of MaxPTP. In MaxPTP, for the first m executions of the learning algorithm, each component is considered individually to determine its accuracy. Some components are pruned using the tolerance limit. Then, all paths with 2 components are considered from the r remaining components where $r \leq m$. Number of combinations of 2 components from r components is ${}^rC_2$.

Extending this logic, total number of combinations for MaxPTP is: m combinations with all components, followed by ${}^rC_2 + \dots + {}^rC_r$ combinations with r components. Thus, the complexity of MaxPTP is $O(m + {}^rC_2 + \dots + {}^rC_r)$ which effectively implies $O(2^r-1)$.

Complexity of MinPTP. In this approach, after pruning the unwanted components there are r components remaining with accuracies greater than or equal to the tolerance limit. Hence, MinPTP executes the learning algorithm at most r times.

Since components are chosen in descending order of accuracies, it is reasonable to expect that a fraction of the r executions will be sufficient to find a solution. To execute the learning algorithm with $r \leq m$ components, it is expected that execution time will be reduced by a factor of r/m of the time needed for all m features. It is thus found that the complexity of MinPTP is $O((m)(m-1)/2)$ in the average case.

3.2 Comparative Discussion

The computational complexity of MaxPT is found to be $O(2^m-1)$ which gives an exponential search space. MinPT, on the other hand, has a computational complexity of $O(m)$, providing a search space that is linear with respect to the total number of components m . It is thus clear that the complexity of MinPT is significantly lower than that of MaxPT. In MaxPTP, the complexity is $O(m + {}^rC_2 + \dots + {}^rC_r)$ where $r \leq m$. Thus, the search space is exponential but with potentially fewer combinations than in MaxPT, assuming that some components will be unwanted and get pruned out. Hence, the complexity of MaxPTP is likely to be lower than that of MaxPT, though both are exponential. Finally, consider MinPTP with complexity $O((m)(m-1)/2)$ which is quadratic. It is of a lower order than MaxPTP, though a higher order than MinPT.

Thus, important points on computational complexity are summarized as follows:

- MaxPT has an exponential complexity, the highest among the approaches.
- MinPT has a linear complexity, the lowest among the approaches.
- MaxPTP has an exponential complexity, but lower than that of MaxPT.
- MinPTP has a quadratic complexity, thus lower than that of MaxPTP.

4 Experimental Evaluation

4.1 Description of Real Scientific Data

Graphs modeling Heat Treating in Materials Science. One type of data we use consists of scientific graphs from the domain of Materials Science. They model the Heat Treating of Materials. Fig. 1 shows such a graph called a heat transfer curve. It plots the heat transfer coefficient versus temperature of a material where the heat transfer coefficient measures heat extraction capacity in a rapid cooling process called quenching [16]. Some features of the graph correspond to physical phenomena in the domain. For example, the Leidenfrost Point *LF* denotes the breaking of the vapor blanket around the part [16]. Other features relate to statistical observations and absolute position of points on the graph.

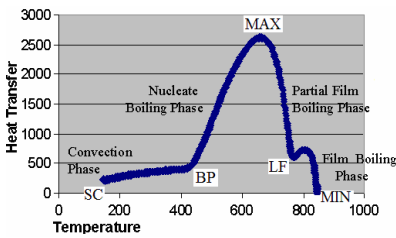


Fig. 1. Heat Transfer Curve

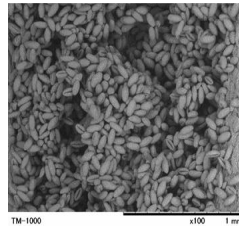


Fig. 2. Pollen Grain

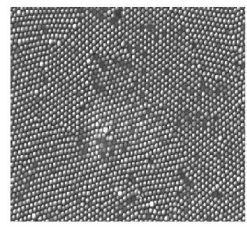


Fig. 3. Silicon Nanopore

Images with Cross-sections of Samples in Bioinformatics and Nanotechnology.

We deal with images taken under a microscope at various stages of scientific process from the domains of Bioinformatics and Nanotechnology. These are cross-sections of samples such as carbon nanofiber, silicon nanopore, pollen grain and herb leaf.

Fig. 2 shows an image from Bioinformatics. It is an inner cross-sectional view of pollen grain. In comparing such images, visual features such as pixel size and grey-scale (or color) are applicable. Also, domain experts consider other features pertaining to metadata such as the level of zooming and the nature of the cross-section (e.g., side view, inner view) [17].

Fig. 3 shows an image from Nanotechnology. It depicts the upper surface of a silicon nanopore array observed at the end of a chemical process called etching [6]. While comparing such images manually, experts observe visual features of the images such as nanoparticle size, interparticle distance and nanoparticle height. Also, meta-data features such as the stage of a process executed on the sample and the location of the sample are applicable.

4.2 Learning Algorithms

LearnMet for Graphs. The input to LearnMet is a training set with correct clusters of graphs provided by domain experts. LearnMet guesses an initial distance function D as a weighted sum of components applicable to the domain. It uses D as the notion of distance in clustering with a partition-based clustering algorithm to get predicted

clusters. It then evaluates clustering accuracy by comparing predicted and correct clusters to obtain the error between them in terms of true positives, true negatives, false positives and false negatives. It adjusts D based on the error using a Weight Adjustment Heuristic [20], and re-executes the clustering and evaluation until error is minimal or below a threshold. It outputs the D giving the lowest error as the learned distance function.

Accuracy α of the learned distance function D is measured as follows. Consider a distinct test set of correct clusters of graphs. Using the learned distance function, predicted clusters are obtained over the test set. Comparing the predicted and correct clusters, true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are determined. Accuracy α is then measured as success rate [22, 20] given as $\alpha = (TP+TN) / (TP+TN+FP+FN)$.

FeaturesRank for Images. In FeaturesRank, the inputs are pairs of images with similarity levels (instead of clusters) thus considering another method of grouping objects in scientific domains. The pairs of images are provided by experts in the form of a training set. For each pair, experts identify whether they consider its images to be different or similar and indicate the extent of similarity based on levels. This gives the notion of correctness as per the domain. FeaturesRank defines a preliminary distance function for the images as a weighted sum of distances between its features. It then uses an iterative approach for learning. In each iteration the given images are hierarchically placed into clusters such that the number of levels of similarity in the clusters is equal to that in the training set. Adjustments are made to the weights of the features in the distance function based on the difference between the extent of similarity in the clusters and the training samples using a Feature Weight Heuristic [19]. The distance function corresponding to minimal error is returned.

The accuracy of the learned distance function is measured using a distinct test set consisting of pairs of images with the correct extent of similarity for each pair given by experts. Using the learned distance function, the images in the test set are clustered in levels. For a given pair of images $P = (I_a, I_b)$, if the extent of similarity in the clusters is equal to that in the test set, then it is considered to be a correct pair. Accuracy is then defined as the ratio of the number of correct pairs over the all the pairs used [19]. If AP refers to all the pairs and CP refers to the number of correct pairs, accuracy α is measured as $\alpha = CP/AP$.

4.3 Experimental Details

A summary of our rigorous experimental evaluation is presented here. In the experiments shown below, error threshold is 0.01, maximum number of iterations is 1000 and tolerance limit is 0.25. The Materials Science training set has 50 graphs in 7 clusters while the distinct test set has 40 graphs in 6 clusters. The Bioinformatics training set has 200 images and the test set has 150 images, both with 3 levels of similarity. The Nanotechnology training set has 150 images and test set has 100 images, both with 4 levels of similarity. The total number of components in Materials Science, Bioinformatics and Nanotechnology are 20, 15 and 17 respectively. The results of our experiments are summarized in the charts below where, each bar shows observations averaged for 10 experiments altering clustering seeds for randomization.

Figs. 4, 5 and 6 show the average accuracy over test sets (distinct from training sets) in Materials Science, Bioinformatics and Nanotechnology respectively. The formulae given in Section 4.2 are used to measure the accuracy in each experiment from which the average accuracy over ten experiments is calculated. It is seen that MaxPT and MaxPTP give higher accuracies than MinPT and MinPTP. We also find that the accuracies of MaxPT and MaxPTP are fairly equal to each other.

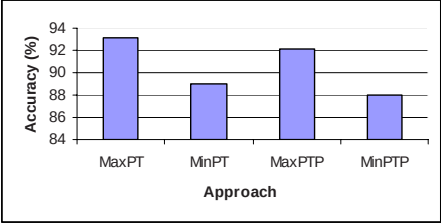


Fig. 4. Accuracy – Materials Science

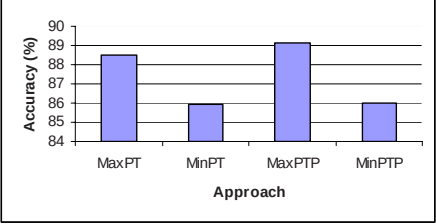


Fig. 5. Accuracy - Bioinformatics

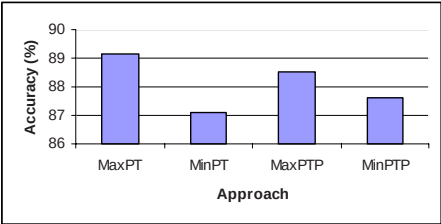


Fig. 6. Accuracy – Nanotechnology

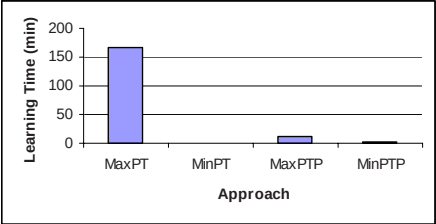


Fig. 7. Efficiency – Materials Science

Figs. 7, 8 and 9 respectively depict the average efficiency in terms of learning time over the training sets in Materials Science, Bioinformatics and Nanotechnology. The learning time is observed for each experiment from which the average over ten experiments is calculated. The average learning time is found to be the highest in MaxPT and the lowest in MinPT. MaxPTP needs far less learning time than MaxPT.

Figs. 10, 11 and 12 show the average simplicity in terms of the number components in the learned distance in Materials Science, Bioinformatics and Nanotechnology respectively. For each experiment the number of components is recorded from which average for 10 experiments is calculated. We find that MinPTP yields distance functions with the fewest number of components throughout, followed by MinPT. MaxPTP gives distance functions with fewer components than MaxPT.

From all the evaluation conducted whose summary is presented above, we find that:

- MinPT is optimal with respect to the efficiency criterion.
- MaxPT is optimal with respect to the accuracy criterion.
- MinPTP is optimal with respect to the simplicity criterion.
- MaxPTP balances between the 3 criteria with accuracy almost equal to MaxPT.

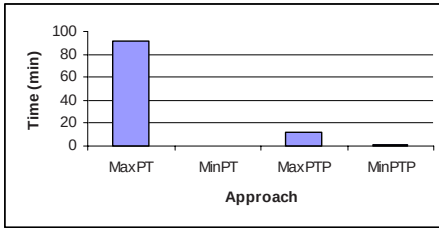


Fig. 8. Efficiency – Bioinformatics

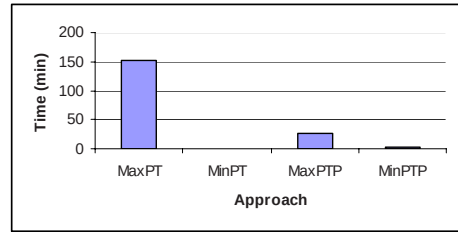


Fig. 9. Efficiency – Nanotechnology

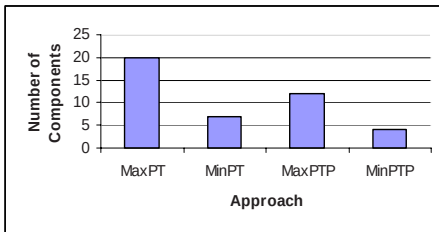


Fig. 10. Simplicity – Materials Science

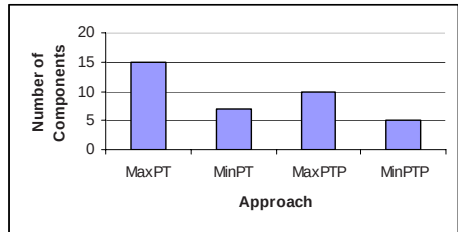


Fig. 11. Simplicity – Bioinformatics

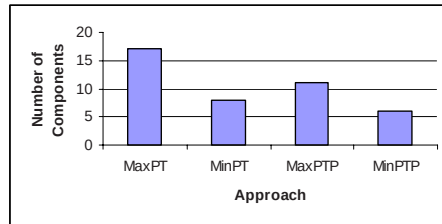


Fig. 12. Simplicity – Nanotechnology

5 Analysis of Results in Applications

5.1 Usefulness in Targeted Scientific Applications

1. Parameter Selection: In these applications users select process parameters in industry based on results of experiments conducted in the laboratory. They are analogous to transaction processing systems where data changes very frequently. Hence, to learn distance functions catering to rapidly changing data, efficiency is critical. Thus, MinPT which provides fast learning is likely to be useful here.

2. Simulation Tools: In these systems, simulations of real laboratory experiments are conducted to save resources. Simulations often take as long as the real experiment and require repeated access to the stored data. Hence issues such as storage, retrieval and indexing of the data are important since each access could take a long time. Thus

simplicity of the learned distance function is crucial. Hence, MinPTP which is optimal in terms of simplicity would be the most useful here.

3. *Intelligent Tutors*: These applications typically demonstrate scenarios analogous to human teachers to clarify concepts in a domain. Thus, it is important to emphasize the importance of different optimization criteria. Hence it would be desirable to use MaxPTP which balances accuracy, efficiency and simplicity.

4. *Decision Support*: In these applications, it is important to provide information for making long-term decisions for issues such as optimizing the concerned scientific processes. Hence, accuracy is critical, making MaxPT a seemingly obvious choice. However, some decision support systems tend towards day-to-day transaction processing, making efficiency an issue. In some systems, fast retrieval of data is critical for at-a-glance display of information. Thus storage, retrieval and indexing criteria become important giving a considerable concern to simplicity. Hence, it may be useful to choose MaxPTP to cater to all the needs of decision support systems.

5.2 Development of Trademarked Software

We developed a real application, namely, a decision support system for process optimization, based on graphical data mining. This system called AutoDomainMine [21] performs decision support in the Heat Treating of Materials. It is a trademarked software tool of the Center for Heat Treating Excellence that supported this research.

From our evaluation of selection approaches, MaxPTP proved to be almost as good as MaxPT in terms of providing optimal accuracy, while balancing well between the other criteria of simplicity and efficiency. We thus preferred the use of the MaxPTP approach in this application. MaxPTP was used for component selection in the LearnMet [20] algorithm which performed distance function learning. The learned distance function was used to cluster graphs obtained from laboratory experiments in heat treating. The clustering criteria, i.e., experimental input parameters leading to the clusters were learned by decision tree classifiers. These served as the basis for predicting the results of future experiments given their input parameters, thus helping to choose materials in order to optimize scientific processes. The accuracy of the prediction was based on the accuracy of the clustering which was dependent on the selection of appropriate components to capture the semantics of the graphical data.

The AutoDomainMine system was evaluated over real data by conducting formal user surveys with test sets distinct from training sets. It was found to give prediction accuracy of approximately 96% [21]. The users concluded that this system provided effective decision support as per their needs. It also showed a considerable improvement over its earlier versions prior to optimizing distance function learning.

6 Related Work

In [12] an overview of different types of distances useful for similarity search in multimedia databases is presented. However, they do not provide methods to learn a single distance function encompassing various distance types. Hinneburg et al. [9] propose a

learning method to find the relative importance of dimensions for n -dimensional objects. Their focus though, is on dimensionality reduction, rather than the semantics of the data. In [24] they learn which type of position-based distance is applicable for the given data starting from the general Mahalanobis distance. They do not consider other distance types besides position-based. Wavelets [23] are often used for image processing in order to compare and rank images. Wavelet coefficients need to be computed each time image comparison is made which is computationally expensive. Our feature-based learning on the other incurs a one-time cost of learning.

The FastMap algorithm [8] maps objects in multimedia databases to points in a user-defined k -dimensional space preserving similarities in the original objects. In [1], probabilistic measures are proposed for similarity search over images. The MindReader system [11] guesses the distance function in the mind of the users based on combining several examples given by users and their relative importance. In [15] they propose a human computer interaction approach to perform content based retrieval of images based on relevance feedback. Learnable similarity measures for strings are presented in [3]. Distance function learning can also be performed using genetic algorithms [7] and neural networks [2]. Our proposed approaches for component selection can be used in conjunction with such state-of-the-art methods to optimize the learning given factors such as accuracy, efficiency and simplicity.

Ensemble learning [22, 14, 25] trains multiple models from training data and their outputs are combined to generate the final predicted result. Popular ensemble based algorithms, such as hierarchical mixture of experts, often produce better results than single classifiers [14]. Techniques to enhance ensemble learning have been studied. Zhou et al. [25] train many neural networks first, then assign random weights to them and employ a genetic algorithm to evolve the weights to characterize fitness of the neural network. In our context each component could be a learner, thus in combining them we get an ensemble. Our selection approaches could be used to select learners in an ensemble, modifying heuristics as needed. Depending on the potential applications, learners could be selected using MaxPT, MinPT, MinPTP or MaxPTP approaches.

7 Conclusions

We address the problem of optimizing distance function learning for complex scientific data. We define a distance function as a weighted sum of components where each component depicts a feature of the data. We propose 4 component selection approaches: MaxPT, MinPT, MaxPTP and MinPTP. Evaluation with real data from scientific domains shows that: MaxPT is likely to be useful when the goal is to attain maximal accuracy in learning; MinPT is preferable when learning efficiency is of the highest priority; MinPTP is most desirable when simplicity of learned distance is important; MaxPTP is suitable when it is important to strike a good balance between the three criteria. Thus, different selection methods can be used based on the priorities of targeted applications. A notable outcome of this research is the development of a decision support system that is a trademarked software tool.

References

1. Aksoy, S., Haralick, R.: Probabilistic versus Geometric Similarity Measures for Image Retrieval. *IEEE CVPR* 2, 357–362 (2000)
2. Bishop, C.: *Neural Networks for Pattern Recognition*. Oxford University Press, England (1996)
3. Bilenko, M., Mooney, R.: Adaptive Duplicate Detection using Learnable String Similarity Measures. In: *KDD*, pp. 39–48 (August 2003)
4. Chen, L., Ng, R.: On the Marriage of Lp-Norm and Edit Distance. In: *VLDB*, pp. 792–803 (August 2004)
5. Das, G., Gunopulos, D., Mannila, H.: Finding Similar Time Series. In: Komorowski, J., Żytkow, J.M. (eds.) *PKDD 1997*. LNCS, vol. 1263, pp. 88–100. Springer, Heidelberg (1997)
6. Dougherty, S., Liang, L., Pins, G.: Precision Nanostructure Fabrication for the Investigation of Cell Substrate Interactions, Technical Report, Worcester Polytechnic Institute, Worcester, MA (June 2006)
7. Friedberg, R.: A Learning Machine: Part I. *IBM Journal* 2, 2–13 (1958)
8. Faloutsos, C., Lin, K.: FastMap: A Fast Algorithm for Indexing, Data Mining and Visualization of Traditional and Multimedia Datasets. *SIGMOD Record* 24(2), 163–174 (1995)
9. Hinneburg, A., Aggarwal, C., Keim, D.: What is the Nearest Neighbor in High Dimensional Spaces. In: Komorowski, J., Żytkow, J.M. (eds.) *PKDD 1997*. LNCS, vol. 1263, pp. 506–515. Springer, Heidelberg (1997)
10. Han, J., Kamber, M.: *Data Mining Concepts and Techniques*. Morgan Kaufmann, California (2001)
11. Ishikawa, Y., Subramanya, R., Faloutsos, C.: MindReader: Querying Databases through Multiple Examples. In: *VLDB*, pp. 218–227 (August 1998)
12. Keim, D., Bustos, B.: Similarity Search in Multimedia Databases. In: *ICDE*, pp. 873–874 (March 2004)
13. Mitchell, T.: *Machine Learning*. WCB McGraw Hill, USA (1997)
14. Polikar, R.: Ensemble Based Systems in Decision Making. *IEEE Circuits and Systems* 6(3), 21–45 (2006)
15. Rui, Y., Huang, T., Mehrotra, S.: Relevance Feedback Techniques in Interactive Content Based Image Retrieval. In: *SPIE*, pp. 25–36 (January 1998)
16. Sisson, R., Maniruzzaman, M., Ma, S.: Quenching: Understanding, Controlling and Optimizing the Process, CHTE Seminar (October 2002)
17. Sheybani, E., Varde, A.: Issues in Bioinformatics Image Processing, Technical Report, Virginia State University, Petersburg, VA (October 2006)
18. Traina, A., Traina, C., Papadimitriou, S., Faloutsos, C.: TriPlots: Scalable Tools for Multidimensional Data Mining. In: *KDD*, pp. 184–193 (August 2001)
19. Varde, A., Rundensteiner, E., Javidi, G., Sheybani, E., Liang, J.: Learning the Relative Importance of Features in Image Data. In: *ICDE's DBRank* (April 2007)
20. Varde, A., Rundensteiner, E., Ruiz, C., Maniruzzaman, M., Sisson, R.: Learning Semantics-Preserving Distance Metrics for Clustering Graphical Data. In: *KDD's MDM*, pp. 107–112 (August 2005)
21. Varde, A., Rundensteiner, E., Sisson, R.: AutoDomainMine: A Graphical Data Mining System for Process Optimization. In: *SIGMOD*, pp. 1103–1105 (June 2007)
22. Witten, I., Frank, E.: *Data Mining: Practical Machine Learning Algorithms with Java Implementations*. Morgan Kaufmann Publishers, San Francisco (2000)
23. Wang, J., Wiederhold, G., Firschein, O., Wei, S.: Content-Based Image Indexing and Searching Using Daubechies Wavelets. *International Journal of Digital Libraries* 1, 311–328 (1997)
24. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance Metric Learning with Application to Clustering with Side Information, *NIPS*, pp. 503–512 (December 2003)
25. Zhou, Z., Wu, J., Tang, W.: Ensembling Neural Networks: Many Could Be Better Than All. *Artificial Intelligence* 137(1), 239–263 (2002)

Emerging Pattern Based Classification in Relational Data Mining

Michelangelo Ceci, Annalisa Appice, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona, 4 - 70126 Bari - Italy
{ceci,appice,malerba}@di.uniba.it

Abstract. The usage of descriptive data mining methods for predictive purposes is a recent trend in data mining research. It is well motivated by the understandability of learned models, the limitation of the so-called “horizon effect” and by the fact that it is a multi-task solution. In particular, associative classification, whose main idea is to exploit association rules discovery approaches in classification, gathered a lot of attention in recent years. A similar idea is represented by the use of emerging patterns discovery for classification purposes. Emerging Patterns are classes of regularities whose support significantly changes from one class to another and the main idea is to exploit class characterization provided by discovered emerging patterns for class labeling. In this paper we propose and compare two distinct emerging patterns based classification approaches that work in the relational setting. Experiments empirically prove the effectiveness of both approaches and confirm the advantage with respect to associative classification.

1 Introduction

Discovering a characterization of classes has been a challenge for research in machine learning and data mining. Emerging patterns (EPs) discovery is a descriptive data mining task which aims at characterizing classes by detecting significant differences between objects of distinct classes. EPs are introduced in [6] as a particular kind of patterns (or multi-variate features) whose support significantly changes from one data class to another: the larger the difference of pattern support, the more interesting the pattern. Change in pattern support is estimated in terms of support ratio (or *growth rate*). EPs with sharp change in support (high growth rate) can be used to characterize object classes.

Originally, EPs discovery has been investigated for capturing difference between separate classes of objects which are stored in a single relational table. Each tuple in the table represents an attribute-value vector describing an object of one of the classes and each EP is discovered in form of a conjunction of attribute values. Anyway, real-world data typically involve complex and heterogeneous objects with different properties which are modeled by as many relational tables as the number of object types. Mining data scattered over the multiple tables of a relational database (*relational data*) is a challenge in several domains,

e.g., spatial domains, process mining and, in general, in domains where the effect of an attribute is not limited to a specific unit of analysis. In this case, when properties of some units (reference objects) are investigated, attributes of related units (task-relevant objects) must be taken into account as well. Traditional EPs discovery methods do not distinguish task-relevant from reference objects, nor do they allow the representation of any kind of interaction. Recently, EP discovery has been framed in the context of relational data mining [8] in order to deal with both relational data and relational patterns [2].

Although (relational) emerging patterns discovery is specially designed for a descriptive task, in this work, we exploit potentialities of EPs to deal with the (relational) classification task. The rationale behind this undertaking can be found in the goal of classification, that is, to learn the concept associated to each class by finding regularities which characterize the class in question and discriminate it from the other classes. The idea is to learn a relational classifier by exploiting the knowledge implicitly encoded in the (relational) EPs that permits to discriminate between different classes sharply.

In this paper, we propose and compare two EPs-based relational classifiers, namely Mr-CAEP and Mr-PEPC. The former upgrades the EP-based classifier CAEP [7] from the single relational table setting (propositional setting) to the relational setting. It computes a membership score of an object to each class. The score is computed by aggregating a growth rate based function of the relational EPs covered by the object to be classified. The largest score determines the object's class. The latter (Mr-PEPC) maximizes the posterior probability that an object belongs to a given class. Probability is estimated by using a set of relational EPs to describe an object to be classified and, then, to define a suitable decomposition of the posterior probability *à la* naive Bayesian classifier to simplify the probability estimation problem.

The paper is organized as follows. In the next section related works are discussed and the present work is motivated. The problem of relational emerging patterns discovery is faced in Section 3. The two relational EP-based classifiers are presented in Section 4. Lastly, experimental results are reported in Section 5 and then some conclusions are drawn.

2 Related Works and Motivations

Data mining research has provided several solutions for the task of emerging patterns discovery. In the seminal work by Dong and Li [6], a border-based approach is adopted to discover the EPs discriminating between separate classes. Borders are used to represent both candidates and subsets of EPs; the border differential operation is then used to discover the EPs. Zhang et al. [20] have described an efficient method, called ConsEPMiner, which adopts a level-wise generate-and-test approach to discover EPs which satisfy several constraints (e.g., growth-rate improvement). Recently, Fan and Ramamohanarao [9] have proposed a method which improves the efficiency of EPs discovery by adopting a CP-tree data structure to register the counts of both the positive and negative class. All these

methods assume that data to be mined are stored in a single data table. An attempt of upgrading the emerging pattern discovery to deal with relational data has been reported in [2], where the authors proposed to adapt the *levelwise* method described in [15] to the case of relational emerging patterns.

In the last years, studies on emerging patterns discovery have generally resulted in methods for building classifiers through a careful selection of high quality emerging patterns discovered in training data [7,10,11,13]. Indeed, the idea of using descriptive data mining methods for predictive purposes has its roots in studies on *associative classification*, where the goal is to induce a classification model on the basis of discovered association rules.

Differently from emerging patterns based classification, associative classification has been studied not only in the propositional setting [14,19], but also in the relational one [4]. Interesting aspects of associative classification can be summarized in the following points:

1. Differently from most of tree-based classifiers, whose univariate tests cause the “horizon effect”, associative classifiers are based on association rules that consider the simultaneous correspondence of values of different attributes, hence promising to achieve better accuracy [3].
2. Associative classification makes association rule mining techniques applicable to classification tasks.
3. The user can decide to mine both association rules and a classification model in the same data mining process [14].
4. The associative classification approach helps to solve the *understandability* problem [16] that may occur with some classification methods. Indeed, many rules produced by standard classification systems are difficult to understand because these systems often use only domain independent biases and heuristics, which may not fulfill user’s expectation. With the associative classification approach, the problem of finding understandable rules is reduced to a post-processing task [14].

All these points are also valid in the case of emerging patterns based classifiers. In addition, we argue that emerging pattern based classifiers take advantages from the fact that EPs provide features which better discriminate classes than association rules do.

3 Relational EPs Discovery

The problem of discovering relational emerging patterns can be formalized as follows:

Given:

- a database schema S with h relational tables $S = \{T_0, T_1, \dots, T_{h-1}\}$
- a set PK of primary key constraints on tables in S
- a set FK of foreign key constraints on tables in S
- a target relation $T \in S$ ¹

¹ Objects in T play the role of reference objects, while objects in $S - \{T\}$ play the role of task relevant objects.

- a target discrete attribute $y \in T$, different from the primary key of T , whose domain is $C = C_1, C_2, \dots, C_r$
- a couple of thresholds $0 < \minSup \leq 1$ and $\minGR \geq 1$

The problem is to find a set of relational emerging patterns.

Formally, a relational emerging pattern is a conjunction of predicates of two different types:

Definition 1 (Structural predicate). A binary predicate $p/2$ is a structural predicate² associated to a table $T_i \in S$ if a foreign key in S exists that connects T_i and a table $T_j \in S$. The first argument of p represents the primary key of T_j and the second argument represents the primary key of T_i .

Definition 2 (Property predicate). A binary predicate $p/2$ is a property predicate associated to a table $T_i \in S$ if the first argument of p represents the primary key of T_i and the second argument represents another attribute in T_i which is neither the primary key of T_i nor a foreign key.

Definition 3 (Relational Emerging Pattern).

A Relational Pattern is in the form:

$$\langle S \rangle \{ \langle attr(A) \rangle \}_{0..n} \{ \langle rel(A, R_k) \rangle \{ \langle attr(R_k) \rangle \}_{0..n} \}_{0..n} \\ \{ \langle rel(R_k, R_k) \rangle \{ \langle attr(R_k) \rangle \}_{0..n} \}_{0..n} \text{ where}$$

- $attr/1$ represents the predicate associated to the target table T (key predicate). The argument represents the primary key of T .
- $rel/2$ represents a generic structural predicate
- $attr/2$ represents a generic property predicate

A pattern P in this form is a relational pattern if the property of linkedness [12] is satisfied (e.g. each variable should be linked to the variable in the key predicate by means of structural predicates). A relational pattern P is a relational emerging pattern if $\exists t \in C \quad GR_{\bar{t} \rightarrow t}(P) > \minGR$ and $\sup_t(P) > \minSup$.

Definitions of support and growth-rate are formally provided below.

Definition 4. The support $\sup_t(P)$ of the pattern P for the class $t \in C$ is: $\sup_t(P) = |O_t(P)|/|O_t|$, where: O_t denotes the set of reference objects labeled with class t , while $O_t(P)$ denotes the subset of reference objects in O_t which are covered by the pattern P .

Definition 5. The growth rate $GR_{\bar{t} \rightarrow t}(P)$ of the pattern P for distinguishing the reference objects labeled with class t from the reference objects labeled with a class different from t (in $\bar{t} = C - \{t\}$), is computed as follows:

$$GR_{\bar{t} \rightarrow t}(P) = \frac{\sup_t(P)}{\sup_{\bar{t}}(P)} \quad (1)$$

$$GR_{\bar{t} \rightarrow t}(P) = \frac{0}{0} = 0 \text{ and } GR_{\bar{t} \rightarrow t}(P) = \frac{\geq 0}{0} = \infty.$$

² “/2” indicates the predicate arity.

The relational emerging pattern discovery is performed by exploring level-by-level the lattice of relational patterns ordered according to a generality relation ($\geq$) between patterns. Formally, given two patterns $P1$ and $P2$, $P1 \geq P2$ denotes that $P1$ ($P2$) is more general (specific) than $P2$ ($P1$). Hence, the search proceeds from the most general pattern and iteratively alternates the candidate generation and candidate evaluation phases (levelwise method). In [2], the authors propose an enhanced version of the level-wise method [15] to discover emerging patterns from data scattered in multiple tables of a relational database. Candidate emerging patterns are searched in the space of relational patterns satisfying linkedness, which is structured according to the θ -subsumption generality order [17].

Definition 6 (θ -Subsumption). *Let $P1$ and $P2$ be two relational patterns on a data schema S . $P2$ θ -subsumes $P1$ if and only if a substitution θ exists such that $P1 \theta \subseteq P2$.*

Having introduced θ -subsumption, generality order between relational patterns can be formally defined.

Definition 7 (Generality Order Under θ -Subsumption). *Let $P1$ and $P2$ be two relational patterns. $P1$ is more general than $P2$ under θ -subsumption, denoted as $P1 \geq_\theta P2$, if and only if $P2$ θ -subsumes $P1$, that is $P1 \theta \subseteq P2$ for some substitution θ .*

θ -subsumption defines a quasi-ordering, since it satisfies the reflexivity and transitivity property but not the anti-symmetric property. The quasi-ordered set spanned by $\geq_\theta$ can be searched according to a downward refinement operator which computes the set of refinements for a relational pattern.

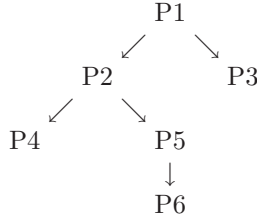
Definition 8 (Downward Refinement Operator Under θ -Subsumption). *Let $\langle G, \geq_\theta \rangle$ be the space of relational patterns ordered according to $\geq_\theta$. A downward refinement operator under θ -subsumption is a function ρ such that $\rho(P) \subseteq \{Q \in G \mid P \geq_\theta Q\}$.*

The downward refinement operator is a refinement operator under θ -subsumption. In fact, it can be easily proved that $P \geq_\theta Q$ for all $Q \in \rho(P)$. This makes possible to perform a levelwise exploration of the lattice of relational patterns ordered by θ -subsumption.

Example 1. Let us consider the relational patterns:

- P1: molecule(M).
- P2: molecule(M), atom(M,A).
- P3: molecule(M), logp(M,[1.8,3.7]).
- P4: molecule(M), atom(M,A), charge(A,[1.7,2.1]).
- P5: molecule(M), atom(M,A), bond(B,A).
- P6: molecule(M), atom(M,A), bond(B,A), type(B,covalent).

They are structured in a portion of a lattice ordered by θ -subsumption, that is:



Emerging patterns for distinguishing reference objects labeled with class t (target class) from reference objects labeled with class b (background class) are then discovered by generating the pattern space one level at a time starting from the most general emerging pattern (the emerging pattern that contains only the key predicate) and then by applying a breadth-first evaluation in the lattice of relational patterns ordered according to $\geq_\theta$.

When a level of the lattice is explored, the candidate pattern search space is represented as a set of enumeration trees (SE-trees)[20]. The idea is to impose an ordering on atoms such that all patterns in the search space are enumerated. Practically, a node g of a SE-tree is represented as a group comprising: the *head* ($h(g)$) that is the pattern enumerated at g , and the *tail* ($t(g)$) that is the ordered set consisting of the atoms which can potentially be appended to g by ρ in order to form a pattern enumerated by some sub-node of g . A child g_c of g is formed by taking an atom $i \in t(g)$ and appending it to $h(g)$, $t(g_c)$ contains all atoms in $t(g)$ that follow i (see Figure 1). In the case i is a structural predicate (i.e., a new relation is introduced in the pattern), $t(g_c)$ contains both atoms in $t(g)$ that follow i and new atoms that can be introduced only after i has been introduced (according to linkedness property). Given this child expansion policy, without any pruning of nodes or pattern, the SE-tree enumerates all possible patterns and avoids generation and evaluation of candidates equivalent under θ -subsumption to some other candidate.

As pruning criterion, the monotonicity property of the generality order $\geq_\theta$ with respect to the support value (i.e., a superset of an infrequent pattern cannot be frequent) [1] can be exploited to avoid generation of infrequent relational patterns. Let P' be a refinement of a pattern P . If P is an infrequent pattern

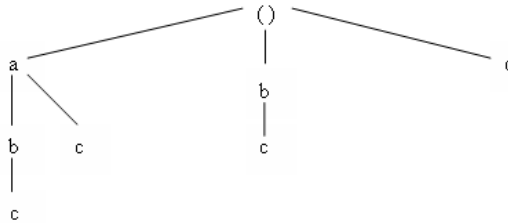


Fig. 1. The enumeration tree over the atoms $A = \{a, b, c\}$ to search the atomsets a, b, c, ab, ac, bc, abc

for a class $t \in C$ ($sup_t(P) < minSup$), then P' has a support on O_t that is lower than the user-defined threshold ($minsup$). According to the definition of emerging pattern, P' cannot be an emerging pattern for distinguishing O_t from $O_{\bar{t}}$, hence it is possible to avoid the refinement of patterns which are infrequent on O_t . Unluckily, the monotonicity property does not hold for the growth rate: a refinement of an emerging pattern whose growth rate is lower than the threshold $minGR$ may or may not be an emerging pattern. However, growth rate can be also used for pruning. In particular, it is possible to stop the search when it is not possible to increase the growth rate with additional refinements [2].

Finally, as stopping criterion, the number of levels in the lattice to be explored can be limited by the user-defined parameter $MAX_L \geq 1$ which limits the maximum number of predicates in a candidate emerging pattern.

4 Relational Classification

Once relational emerging patterns are extracted, they are used to mine classifiers. Two EPs-based relational classifiers are proposed in the following: Mr-CAEP and Mr-PEPC.

4.1 Mr-CAEP

Mr-CAEP (Multi-Relational Classification based on Aggregating Emerging Patterns) upgrades the EP-based classifier CAEP [7] to the relational setting. It computes a membership score of an object to each class. The score is computed by means of growth rate based function of the relational EPs covered by the object to be classified. The largest score determines the object's class.

The score is computed on the basis of the subset of relational emerging patterns that cover the object to be classified. Formally, let o be the description of the object to be classified (an object is represented by a tuple in the target table and all the tuples related to it according to foreign key constraints), $\mathcal{R}(o) = \{R_k \in \mathcal{R} | \exists \theta R_k \theta \subseteq o\}$ is the set of relational emerging patterns that cover the object o .

The score of o on the class C_i is computed as follows:

$$score(o, C_i) = \sum_{R_k \in \mathcal{R}(o)} \frac{GR_{\overline{C_i} \rightarrow C_i}(R_k)}{GR_{\overline{C_i} \rightarrow C_i}(R_k) + 1} sup_{C_i}(R_k) \quad (2)$$

This measure may result in an inaccurate classifier in the case of unbalanced datasets that is, when training objects are not uniformly distributed over the classes. In order to mitigate this problem, we follow the same solution proposed in [7] and we normalize this score on the basis of the median of the scores obtained from training examples belonging to C_i .

Formally, the classification is based on the following equation:

$$class(o) = argmax_{C_i \in C} \frac{score(o, C_i)}{median_{to \in TS}(score(to, C_i))} \quad (3)$$

where TS represents the training set.

4.2 Mr-PEPC

In Mr-PEPC (Multi-Relational Probabilistic Emerging Patterns Based Classifier), relational emerging patterns are used to build a naïve Bayesian classifier which classifies any object o by maximizing the *posterior probability* $P(C_i|o)$ such that o is of class C_i , that is, $class(o) = \arg \max_i P(C_i|o)$. By applying the Bayes theorem, $P(C_i|o)$ is reformulated as: $P(C_i|o) = \frac{P(C_i)P(o|C_i)}{P(o)}$. Since $P(o)$ is independent of the class C_i , it does not affect $class(o)$, that is, $class(o) = \arg \max_i P(C_i)P(o|C_i)$. Under the conditional independence assumption (*naïve Bayes assumption*), the likelihood $P(o|C_i)$ can be factorized: $P(o|C_i) = P(o_1, \dots, o_m|C_i) = P(o_1|C_i) \times \dots \times P(o_m|C_i)$, where $o_1, \dots, o_m$ represent the set of attribute values. Surprisingly, naïve Bayesian classifiers have been proved accurate even when the conditional independence assumption is grossly violated [5].

The formulation reported above for naïve Bayesian classifiers is clearly limited to propositional representations. In the case of structural representations, some extensions are necessary. The basic idea is that of using a set of relational emerging patterns to describe an object to be classified, and then to define a suitable decomposition of the likelihood $P(o|C_i)$ à la naïve Bayesian classifier to simplify the probability estimation problem. $P(o|C_i)$ is computed on the basis of the subset $\mathfrak{R}(o) \subseteq \mathfrak{R}$:

$$P(o|C_i) = P\left(\bigwedge_{R_k \in \mathfrak{R}(o)} R_k|o_i\right). \quad (4)$$

The straightforward application of the naïve Bayes independence assumption to all literals in $\bigwedge_{R_k \in \mathfrak{R}(s)} R_k$ is not correct, since it may lead to underestimate the probabilities for the case of classes for which several emerging patterns are found.

To prevent this problem we resort to the logical notion of factorization [18] which is given for clauses (i.e., disjunctions of literals) but we adapt it to the notion of relational pattern.

Definition 9. Let P be a relational pattern, which has a non-empty subset $Q \subseteq P$ of unifiable literals with most general unifier (mgu) θ . Then $P\theta$ is called a factor of P .

A factor of a pattern P is obtained by applying a substitution θ to P which unifies one or more literals in P , and then deleting all but one copy of these unified literals. In our context we are interested in particular factors, namely those that are obtained by substitutions θ which satisfy three conditions:

1. $Domain(\theta) = \bigcup_{R_k \in \mathfrak{R}(o)} Vars(R_k)$ that is, the domain of θ includes all variables occurring in the relational pattern $R_k \in \mathfrak{R}(o)$,
2. $Domain(\theta) \cap Range(\theta) = \emptyset$, that is, θ renames all variables occurring in the association rules $R_k \in \mathfrak{R}(o)$ with new variable names,

3. $\theta|_{Vars(R_k)}$ is injective, that is, the restriction of θ on the variables occurring in R_k is injective.

For each pattern P , a factor always exists. In the trivial case, it coincides with P up to a redenomination of variables in P . A factor $P\theta$ is minimal, when there are no other factors of P with less literals than $P\theta$.

As stated previously, a straightforward application of the naïve Bayes independence assumption may result in totally unreliable probability estimates because of the presence of redundant literals. For this reason, we impose that $P(o|C_i) = P(F|C_i)$ for any minimal factor F of $\bigwedge_{R_k \in \mathcal{R}(o)} R_k$.

By separating the contribution of the conjunctions of literals corresponding to structural predicates ($struct(F)$) from the contribution of the conjunction of literals corresponding to property predicates ($props(F)$) we have:

$$P(o|C_i) = P(struct(F)|C_i) \times P(props(F)|struct(F) \wedge C_i) \quad (5)$$

Under the naïve Bayes independence assumption, $P(struct(F)|C_i)$ can be factorized as follows:

$$P(struct(F)|C_i) = \prod_{rel_j(A,B) \in struct(F)} P(rel_j(A,B)|C_i), \quad (6)$$

where $P(rel_j(A,B))$ is computed on the basis of the relative frequency, computed on the training set, that two tuples are associated according to foreign key constraints.

The naïve Bayes conditional independence can also be assumed for the computation of $P(props(F)|struct(F) \wedge C_i)$, in which case

$$P(props(F)|struct(F) \wedge C_i) = \prod_{attr_j(A,v) \in props(F)} P(attr_j(A,v)|struct(F) \wedge C_i). \quad (7)$$

where $P(attr_j(A,v)|struct(F) \wedge C_i)$ is computed on the basis of the relative frequency, computed on the training set, that the property predicate is satisfied given $struct(F)$ and C_i .

5 Experimental Results

Mr-CAEP and Mr-PEPC have been implemented as modules of the multi-relational data mining system MURENA (MULTI RELational aNalyzer) which interfaces the Oracle 10g DBMS. We tested the methods on two real world data sets: the dibEmail dataset and the North-West England Census Data. Both data sets include numeric attributes, which are handled through an equal-width discretization to partition the range of values into a fixed number of bins. Experiments aim at both comparing Mr-CAEP vs. MrPEPC and comparing emerging patterns based classification against associative classification approaches in the context of relational data mining.

5.1 Data Sets

Dibemail Data Set. These data concern incoming emails processed by the mail server of the Department of Computer Science at University of Bari in the period between 20-Aug-2007 and 3-Oct-2007.

In all, there are 91,291 incoming messages. For each incoming message some information are stored. In particular, the client that dispatched the email, the ip address of the machine that sent the email, the email size, time and date the email was received, percentage of existing destination accounts, number of specified accounts. For each incoming message, multiple outgoing message can be generated: one for each specified account plus accounts automatically generated (mailing lists). In all, there are 111,787 outgoing message and for each outgoing message, the receiver is stored. For each receiver (there are 188 receivers, one for each employed) some information are available such as: role, number of taught courses (if lecturer). For each course (there are 1,044 stored courses), we considered the number of students attending the course, attending period and academic year.

The training set referred to emails received in the period 20-Aug-2007 : 15-Sep-2007 (52,920 incoming messages), while emails received in the period 16-Sep-2007 : 3-Oct-2007 (38,371 incoming messages) are considered as testing objects. In this application, the goal is to classify emails as spam or no-spam.

The North-West England Census Data. Data were obtained from both census and digital maps provided by the European project SPIN! (<http://www.ais.fraunhofer.de/KD/SPIN/project.html>). They concern Greater Manchester, one of the five counties of North West England (NWE). Greater Manchester is divided into into 214 census sections (wards). Census data are available at ward level and provide socio-economic statistics (e.g. mortality rate) as well as some measures of the deprivation of each ward according to information provided by Census combined into single index scores. We employed the Jarman score that estimates the need for primary care, the indices developed by Townsend and Carstairs to perform health-related analyses, and the DoE index which is used in targeting urban regeneration funds. The higher the index value the more deprived the ward. The analysis we performed was based on deprivation factors and geographical factors represented in topographic maps of the area. Vectorized boundaries of the 1998 census wards as well as of other Ordnance Survey digital maps of NWE are available for several layers such as urban area (115 lines), green area (9 lines), road net (1687 lines), rail net (805 lines) and water net (716 lines). Elements of each layer are stored as tuples of relational tables including information on the type (TYPE). For instance, an urban area may be either a “large urban area” or a “small urban area”. Topological relationships between wards and objects in these layers are materialized as relational tables expressing non-disjoint relations. The number of materialized “non disjoint” relationships is 5313.

In this application, the goal is to classify the DoE index for targeting urban regeneration funds. Results are obtained by means of a 10-fold cross validation.

5.2 Mr-CAEP vs Mr-PEPC

In Table 1, results on the dibEmail Data set are reported. This table reports accuracy results of Mr-CAEP and Mr-PEPC as well as the number of relational EPs discovered by varying values of *minGR* and *minSup*. Results are in favour of Mr-CAEP when *minGR* = 1. The situation changes for *minGR* = 1.5 when results do not show a clear advantage of one approach over the other. Indeed, it seems that Mr-PEPC suffers from problems coming from the high number of probabilities to be computed and takes advantage from highly discriminating emerging patterns.

Table 1. Accuracy results on the dibEmail testing set for different values of *minGR* and *minSup* ($MAX_L = 3$)

minGR	minSup	Mr-CAEP	Mr-PEPC	No of discovered EPs
1	0.05	97.93	82.7	254
1	0.1	97.97	83	214
1.5	0.05	98.11	95.53	21
1.5	0.1	97.97	97.7	14

From a descriptive data mining perspective, the following EP has been discovered for the class “spam”:

$$inmessage(A) \wedge inmessage\_outmessage(A, B) \wedge inmessage\_percentageExistingAccounts(A, [48.0..57.6]).$$

This pattern has support of 0.06 and growth rate 5.32. It captures the fact that when an incoming message is associated with an outgoing message and there is a high percentage of not existing destination accounts (percentage of existing account between 48% and 57.6%), it can be used to discriminate spam messages from non-spam messages.

The same behavior is observed on North West England Census Data. In particular, in Table 2, it becomes evident the fact that the two classifiers reach the same maximum predictive accuracy for different values of growth rate. Indeed, in Figure 2, we see that Mr-CAEP is able to take advantage from the high number of discovered emerging patterns, while Mr-PEPC reaches the best performance when *minGR* = 2. By considering only Emerging Patterns with growth rate equal to *Infinity* (Jumping Emerging Patterns), the accuracy of both classifiers decreases. This suggests us that some useful information is lost when working only with Jumping Emerging Patterns.

5.3 Associative Classification vs Emerging Patterns Based Classification

In this subsection we compare Mr-PEPC with its associative classification counterpart described in [4] in the context of spatial data mining. In that case, the

Table 2. North West England Census Data. Mr-CAEP Vs Mr-PEPC: 10-Cross Validation average accuracy and average number of discovered EPs for different values of $minGR$ ($minSup = 0.1$, $MAX_L = 3$).

minGR	Mr-CAEP	Mr-PEPC	No discovered Eps
1	93.18	90	1858
1.5	91.36	92.27	820.9
2	90.28	93.18	603.9
2.5	90.28	91.82	549.6
3	90.28	90.91	531.7
3.5	90.28	90.28	518.7
7	90.28	89.37	345
30	90.28	89.83	280.7
100	90.28	89.83	280.7

classification is based on extracted association rules and is performed by resorting to a naïve Bayesian classifier. The comparison is performed on the same cross validation framework.

Results of the associative classifier on North West England Census Data are reported in Table 3. Results vary on the basis of minimum support, minimum confidence and number of literals in extracted association rules. It is noteworthy, by comparing results in Table 2 with results in Table 3 in a predictive data mining perspective, that the emerging pattern based classifier outperforms its associative classification counterpart when $minGR \in [1.5, 2.5]$.

In a descriptive data mining perspective, it is not possible to compare results. Mr-PEPC returns relational emerging patterns, while the associative classifier returns association rules. In the following, an example of discovered relational emerging pattern for the class DoE index = low (zone to be addressed by regeneration funds) is reported:

$$ward(A) \wedge ward\_rails(A, B) \wedge ward\_carstairsidx(A, [-2.30..0.23]).$$

This emerging pattern has support of 0.24 and growth rate *Infinity*. It captures the fact that when a ward is crossed by a railway and has a low value of Carstairs index there is high demand for urban areas' regeneration projects.

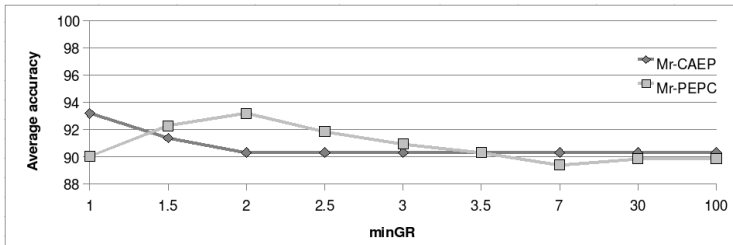


Fig. 2. Mr-CAEP Vs Mr-PEPC: average accuracy varying $minGR$ on North West England Census Data ($minSup = 0.1$, $MAX_L = 3$)

Table 3. DoE Index Associative Classification average accuracy

minsup	minconf	K	Associative Classification
0.2	0.8	5	90.3
0.2	0.8	6	88.3
0.2	0.8	7	87.4
0.1	0.6	5	91.2
0.1	0.6	6	91.2
0.1	0.6	7	91.2

6 Conclusions

In this paper, we presented two emerging patterns based classifiers that work in the multi-relational setting. The first approach classifies objects on the basis of an heuristic evaluation function. The second approach is based on a probabilistic evaluation. By comparing the two approaches, we noted that the probabilistic approach suffers from the high number of considered emerging patterns, but takes advantages from more discriminative patterns. The comparison with an associative classification approach, in fair conditions, confirm the advantage of relational emerging patterns discovery. This because associative classification considers many association rules that are not able to discriminate one class from the others.

As future work, we intend to evaluate scalability of the proposed approaches.

Acknowledgment

This work is partial fulfillment of the research objective of ATENEO-2008 project “Scoperta di conoscenza in domini relazionali”. The authors thank Costantina Caruso for providing dibEmail data set.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) International Conference on Management of Data, pp. 207–216 (1993)
2. Appice, A., Ceci, M., Malgieri, C., Malerba, D.: Discovering relational emerging patterns. In: Basili, R., Pazienza, M. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 206–217. Springer, Heidelberg (2007)
3. Baralis, E., Garza, P.: Majority classification by means of association rules. In: Lavrac, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 35–46. Springer, Heidelberg (2003)
4. Ceci, M., Appice, A.: Spatial associative classification: propositional vs structural approach. *Journal of Intelligent Information Systems* 27(3), 191–213 (2006)
5. Domingos, P., Pazzani, M.: On the optimality of the simple bayesian classifier under Zeo-Ones loss. *Machine Learning* 28(2-3), 103–130 (1997)

6. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: International Conference on Knowledge Discovery and Data Mining, pp. 43–52. ACM Press, New York (1999)
7. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: Classification by aggregating emerging patterns. In: Arikawa, S., Furukawa, K. (eds.) DS 1999. LNCS (LNAI), vol. 1721, pp. 30–42. Springer, Heidelberg (1999)
8. Džeroski, S., Lavrač, N.: Relational Data Mining. Springer, Heidelberg (2001)
9. Fan, H., Ramamohanarao, K.: An efficient singlescan algorithm for mining essential jumping emerging patterns for classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 456–462 (2002)
10. Fan, H., Ramamohanarao, K.: A bayesian approach to use emerging patterns for classification. In: Australasian Database Conference, vol. 143, pp. 39–48. Australian Computer Society, Inc. (2003)
11. Fan, H., Ramamohanarao, K.: A weighting scheme based on emerging patterns for weighted support vector machines. In: Hu, X., Liu, Q., Skowron, A., Lin, T.Y., Yager, R.R., Zhang, B. (eds.) IEEE International Conference on Granular Computing, pp. 435–440 (2005)
12. Helft, N.: Inductive generalization: a logical framework. In: Progress in Machine Learning, pp. 149–157. Sigma Press (1987)
13. Li, J., Dong, G., Ramamohanarao, K., Wong, L.: DeEPs: A new instance-based lazy discovery and classification system. *Machine Learning* 54(2), 99–124 (2004)
14. Liu, B., Hsu, W., Ma, Y.: Integrative classification and association rule mining. In: Proceedings of AAAI Conference of Knowledge Discovery in Databases (1998)
15. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* 1(3), 241–258 (1997)
16. Pazzani, M., Mani, S., Shankle, W.: Beyond concise and colorful: learning intelligible rules. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 235–238. AAAI Press, Menlo Park (1997)
17. Plotkin, G.D.: A note on inductive generalization. 5, 153–163 (1970)
18. Robinson, J.A.: A machine oriented logic based on the resolution principle. *Journal of the ACM* 12, 23–41 (1965)
19. Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In: SIAM International Conference on Data Mining (2003)
20. Zhang, X., Dong, G., Ramamohanarao, K.: Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In: Knowledge Discovery and Data Mining, pp. 310–314 (2000)

Rosso Tiziano: A System for User-Centered Exploration and Discovery in Large Image Information Bases

Giovanni Maria Sacco

Dipartimento di Informatica, Università di Torino, Corso Svizzera 185,
10149 Torino, Italy
sacco@di.unito.it

Abstract. Retrieval in image information bases has been traditionally addressed by two different and unreconciled approaches: the first one uses normal query methods on metadata or on a textual description of each item. The second one works on low-level multimedia features (such as color, texture, etc.) and tries to find items that are similar to a specific selected item. Neither of these approaches supports the most common end-user task: the exploration of an information base in order to find the “right” items. This paper describes a prototype system based on dynamic taxonomies, a model for the intelligent exploration of heterogeneous information bases, and shows how the system implements a new access paradigm supporting guided exploration, discovery, and the seamless integration of access through metadata with methods based on low-level multimedia features. Example interactions are discussed, as well as the major implications of this approach.

1 Introduction

Current research on image retrieval focuses on two different and unreconciled approaches for accessing multimedia databases: the *metadata* approach as opposed to the *content-based* approach. In the metadata approach, each image is described by metadata. Metadata types range from a set of keywords or a textual description, to standardized structures for metadata attributes and their relationships like the MPEG-7 standard [8], to ontologies [20]. While some metadata (e.g. image format) can be automatically derived from the image itself, the vast majority of them are manually entered. Once the items in the collection are described by metadata, the type and actual content of the item itself becomes irrelevant for browsing and retrieval and only needed when the item itself has to be “shown” to the user. From this point of view, it is intuitively appealing and straightforward to use techniques such as database queries on structured metadata or text retrieval queries on metadata consisting of a textual description of the multimedia item.

The content-based approach (CBIR, content-based image retrieval) describes the image through low-level features such as color, texture, shape, etc. which are automatically extracted from images. Retrieval is based on similarity among images: the user provides an item (selected through metadata or, in some cases, at random) and requests similar ones.

Both approaches suffer from significant drawbacks. The metadata approach relies on descriptions that are known to be inaccurate, heavily dependent on the specific

human classifier, ambiguous, etc. These problems can be alleviated by using ontological metadata rather than plain text descriptions, but a level of subjectivity remains. In addition, semantic annotation is costly, especially for large, existing databases.

CBIR originated as an attempt to overcome these problems, by stressing the automatic extraction of "descriptions" from the image itself. This process is inexpensive and parallelization can overcome any capacity problems. In addition, the characterization is completely objective, and does not suffer from the inconsistencies caused by human classification. However, despite significant improvements over the years, the accuracy of CBIR systems is admittedly less than satisfactory. The main reason for this is the semantic gap or "... the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for the user in a given situation" [21]. We believe that no CBIR system will be able to reconstruct all relevant information in all situations: in many practical cases, the information is just not there. As an example, a photo with a mountain scene is similar to many other mountain scenes, but it can come from different (mountain) countries. If the photo chosen as the cover for a leaflet by the Austrian Tourist Office turns out to be from France or Italy, a less than enthusiastic client can be expected.

In considering user access, it is obvious that the dichotomy between the two approaches (metadata vs. low-level features) forces the user to use different access strategies that only depend on the type of feature (conceptual or low-level) he is considering. Most importantly, from the user point of view, none of these approaches really supports an exploratory access to the image collection, which we believe to be the most common access paradigm. Consider our user looking for the cover photo for the Austrian Tourist Office. She would probably like to find out which types of photos on Austria the information base stores: e.g. mountains, towns, museums, etc. Once she focused on the most interesting type (according to her current requirements), say mountains, she might be interested in knowing that there are winter, spring, etc. photos on mountains, or there are photos with a specific dominant, etc.

In short, the user needs to explore the information base. We define *exploration* as an iterative activity in which the user must be able to:

1. have a systematic summary S of the entire universe U
2. freely focus on any subset of the entire universe $F \subseteq U$ and have a systematic summary S' of F
3. repeat 2 by setting additional, secondary foci until the number of selected items is sufficiently small for manual inspection.

The major difference between exploration and plain browsing rests on systematic summaries that provide concise descriptions of content that would otherwise require a time-consuming scan of the actual items. This implies that some sort of conceptual organization exists and that the exploration system is able to summarize any subset of the collection based on such organization. Closely linked to our definition of exploration is the notion of *discovery*: the user exploring the information base will often discover new and unexpected relationships among concepts. We want to stress here that exploration is not an additional, desirable feature of a multimedia information retrieval system. On the contrary, we believe that, in most practical cases, retrieval without exploration is just a trial-and-error task, with no guarantee of the quality of the user's final choice.

The access paradigm supported by most current image retrieval systems is quite different. Systems based on metadata use access techniques such as queries on structured data or text retrieval techniques that work on precise specifications and do not support exploration because they produce flat result lists. The inadequacy of these tools for exploration has been discussed in literature [14, 4]. CBIR systems use information retrieval techniques that are centered on retrieval by similarity. This type of access affords a very simple and intuitive user interaction, but offers no clue on what the information base contains. Systems that organize images through hierarchical clustering, e.g. [22], do offer an initial systematic summary of the collection, but do not account for the iterative refinement required by our working definition of exploration. From this point of view, they are similar to traditional taxonomies that offer conceptual but static summaries.

There are very few exceptions. Sacco [14] considered the application of dynamic taxonomies, a knowledge management model that supports exploration (and forms the backbone of the present approach) to multimedia databases described by conceptual metadata. The same model was used by Hearst et al. [4] to build a prototype system, *Flamenco*, that was successfully applied to a rather large image collection [24]. These works rely on conceptual metadata features and ignore low-level multimedia features. From another perspective, *El Niño*, a prototype system by Santini and Jain [19], focuses on browsing based on low-level features and textual descriptions. As we commented before, browsing is quite different from our definition of exploration. *El Niño*, in fact, works on a multi-feature weighted similarity measure and relies on a visual relevance feedback interface to modify these weights and try to match the user notion of similarity.

The approach we describe here extends the initial ideas reported in [15] and proposes a significant change in access paradigms, based on dynamic taxonomies. A system was implemented in order to make all the implications of the non-traditional design directions we are taking concrete. The prototype system discussed here has several goals. First, we provide a single, coherent framework that seamlessly integrates access by metadata and access by low-level features: it considerably simplifies user access, and each access method reinforces the effectiveness of the other one. Second, this framework is used to support the exploration of image collections, so that both metadata and low-level features can be used to express interest foci, and at the same time to systematically summarize them, in order to guide the user towards his goal. A number of problems that are relevant in this context are discussed in the following. Finally, such a prototype system, in which different low-level features approaches can be easily integrated and compared, can provide an excellent test bed for the future evaluation of different strategies, integrated access and human factors in general. At the present stage, we report the first informal findings of users experimenting with the system.

The information base used in the examples below consists of 251 images of five masters of the Italian Renaissance: Piero della Francesca, Masaccio, Antonello da Messina, Paolo Uccello and Raphael. Each work was thoroughly classified according to a number of topics that include, among others, the painter name, the type of painting (single, polyptic, etc.), the technique used (oil, tempera, etc.), the period in which it was painted, current locations, the themes (religious painting, portrait), etc. Differently from other test image databases, which usually exhibit widely different images,

the images in the sample collection are relatively similar and therefore harder to characterize. In addition, the collection is a good representative of one of the most important applications of image retrieval: museum and art collections. The sample infobase is available on-line at <http://tiziano.di.unito.it>, and is managed by Knowledge Processors' Universal Knowledge Processor [5].

Although we focus here on describing image information bases through low-level features and metadata, the dynamic taxonomy approach can be used in a number of variations, by considering metadata only or low-level features only, or by integrating traditional CBIR retrieval by similarity with a dynamic taxonomy metadata description in order to clarify contexts for similar objects.

2 Dynamic Taxonomies Reviewed

Dynamic taxonomies ([13, 14], also known as faceted search) are a general knowledge management model for complex, heterogeneous information bases. It has an extremely wide application range [18] that includes, among others, electronic commerce, e-government, human resource management and medical guidelines and diagnosis. The intension of a dynamic taxonomy is a taxonomy designed by an expert, i.e. a concept hierarchy going from the most general to the most specific concepts. A dynamic taxonomy does not require any other relationships in addition to subsumptions (e.g., IS-A and PART-OF relationships) and directed acyclic graph taxonomies modeling multiple inheritance are supported but rarely required.

Dynamic taxonomies work on conceptual descriptions of items, so that heterogeneous items of any type and format can be managed in a single, coherent framework. In the extension, items can be freely classified under several topics at any level of abstraction (i.e. at any level in the conceptual tree). This multidimensional classification is a generalization of the monodimensional classification scheme used in conventional taxonomies and models common real-life situations. First, an item is very rarely classified under a single topic, because items are very often about different concepts. Second, items to be classified usually have different independent features (e.g. Time, Location, etc.), each of which can be described by an independent taxonomy. These features are often called *perspectives* or *facets*.

By defining concepts in terms of instances rather than properties, a concept C is just a label that identifies all the items classified under C . Because of the subsumption relationship between a concept and its descendants, the items classified under C ($items(C)$) are all those items in the *deep extension* of C , i.e. the set of items identified by C includes the *shallow extension* of C (all the items directly classified under C) union the deep extension of C 's sons. By construction, the shallow and the deep extension for a terminal concept are the same.

There are two important consequences of our approach. First, since concepts identify sets of items, logical operations on concepts can be performed by the corresponding set operations on their extension. Second, dynamic taxonomies can infer all the concepts related to a given concept C , on the basis of empirical evidence: these concepts represent the conceptual summary of C . Concept relationships other than IS-A are inferred through the extension only, according to the following *extensional inference rule*: two concepts A and B are related if there is at least one item d in the

infobase which is classified at the same time under A (or under one of A's descendants) and under B (or under one of B's descendants). For example, an unnamed relationship between Raphael and Rome can be inferred if an item classified under Raphael and Rome exists in the infobase. At the same time, since Rome is a descendant of Italy, also a relationship between Raphael and Italy can be inferred.

The extensional inference rule can be easily extended to cover the relationship between a given concept C and a concept expressed by an arbitrary subset S of the universe: C is related to S if there is at least one item d in S which is also in items(C). In this way, the extensional inference rule can produce conceptual summaries not only for base concepts, but also for any logical combination of concepts and, most importantly, access through dynamic taxonomies can be easily combined with other retrieval methods such as information retrieval, etc.

2.1 Information Access through Dynamic Taxonomies

The user is initially presented with a tree representation of the initial taxonomy for the entire infobase. In general, each concept label has also a count of all the items classified under it (i.e. the cardinality of items(C) for all C's), because this statistical information is important to guide the search. The initial user focus F is the universe (i.e. all the items in the infobase). In the simplest case, the user can then select a concept C in the taxonomy and *zoom* over it. The zoom operation changes the current state in two ways. First, the current focus F is refined by intersecting it with C (i.e., with items(C)); items not in the focus are discarded. Second, the tree representation of the taxonomy is modified in order to summarize the new focus. All and only the concepts related to F are retained and the count for each retained concept C' is updated to reflect the number of items in the focus F that are classified under C'. The reduced taxonomy is a conceptual summary of the set of items identified by F, exactly in the same way as the original taxonomy was a conceptual summary of the universe. In fact, the term *dynamic taxonomy* is used to indicate that the taxonomy can dynamically adapt to the subset of the universe on which the user is focusing, whereas traditional, static taxonomies can only describe the entire universe.

The retrieval process is an iterative thinning of the information base: the user selects a focus, which restricts the information base by discarding all the items not in the current focus. Only the concepts used to classify the items in the focus, and their ancestors, are retained. These concepts, which summarize the current focus, are those and only those concepts that can be used for further refinements. From the human computer interaction point of view, the user is effectively guided to reach his goal, by a clear and consistent listing of all possible alternatives.

The advantages of dynamic taxonomies over traditional methods are dramatic in terms of convergence of exploratory patterns and in terms of human factors. Three zoom operations on terminal concepts are sufficient to reduce a ten million-item information base to an average ten items [17]. Dynamic taxonomies require a very light theoretical background: namely, the concept of a subject index (i.e. the taxonomic organization) and the zoom operation, which seems to be very quickly understood by end-users. Hearst et al. [4] and Yee et al. [24] conducted usability tests on a corpus of art images described by metadata only, showing a significantly better recall than

access through text retrieval and, perhaps more importantly, the feeling that one has actually considered all the alternatives in reaching a result.

3 Combining Conceptual Access with Low-Level Multimedia Features

Access by low-level multimedia features is usually based on clustering: items are grouped on the basis of the values of one or more features, say color, according to a measure of similarity between any two items. The goal is to create clusters in such a way that the similarity between any two items in a cluster K is higher than the similarity between any item in K and any item not in K . Techniques of this type derive from the ample body of research on clustering for textual information bases. Generally a vector space model is used, in which a item is represented by an n -dimensional vector $\mathbf{x}=(x_1, \dots, x_n)$. The similarity between any two items can then be computed as the distance $d(\mathbf{x}, \mathbf{y})$ between the two vectors that represent the items; the cosine of the angle between the two vectors is generally used, but other measures, such as Jaccard's coefficient, can be used. This geometric interpretation has an additional benefit, in that a cluster of items can be represented by its centroid, which is either the barycenter of the cluster or the item closest to it.

If we ignore the geometric interpretation of clusters, a cluster merely identifies a set of items that are grouped together by some affinity. This definition is, for all access purposes, equivalent to the definition of a concept in a dynamic taxonomy. In fact, in a dynamic taxonomy, a concept denotes a set of items classified under it, rather than a set of properties that instances of a concept must satisfy. Consequently, a rather straightforward strategy to integrate clusters in a dynamic taxonomy is by adding a top-most concept (or "facet") for each clustering scheme, with its actual clusters being the sons of this concept. For instance, if a clustering scheme by dominant color exists, a facet labeled "dominant color" will be added at the top-most level. Its sons will be all the clusters grouping items by dominant color similarity.

There are two obvious problems. The first one is how these clusters are to be labeled in such a way that their content is easily identifiable and understandable: labeling strategies are discussed in the following. The second problem is that, in most situations, the number of clusters for each feature will be large and difficult to represent in the interface and to manipulate by users: hierarchical clustering schemes can greatly simplify the effectiveness of interaction.

In the approach presented here, methods based on low-level features benefit from dynamic taxonomies in two basic ways. First, combinations of concepts can be used to supply a *conceptual context* to such methods, and consequently reduce noise. Such a context "is essential for determining the meaning of an image and for judging image similarity" [19]. Alternatively, when the user starts from retrieval based on low-level features, dynamic taxonomies can be used to quickly summarize the result according to the original conceptual taxonomy, thus increasing the precision of the result. This is especially important in order to quickly correlate low-level features with metadata.

As an example, consider a user starting from a low-level feature such as a blue dominant color. Focusing on it, the system will show in the conceptual summary that images with a blue dominant color include skies, sea, lakes, cars, etc. Conversely, a user

focusing on skies will find that images of skies may have a blue dominant color, but also a yellow one, a red one, etc. In both cases, the conceptual summary indicates which conceptual contexts are available to further refine user access.

As we mentioned before, the integration of metadata access with access by low-level features boosts the effectiveness of each type of access. Very simple features, such as dominant color, are probably not sufficient *per se* to provide an effective access to a large image base: however, when combined with other features, they can provide a useful discrimination. On the other hand, the availability of low-level features may make it useless to manually enter a large number of metadata descriptions, and as we show below, even a simple and small metadata structure can produce significant benefits in access and exploration.

In fact, an extremely important point in the application of dynamic taxonomies to image (and, in general, multimedia) information bases is the dramatic convergence dynamic taxonomies offer. Following the analysis reported in [17], assume that we organize the taxonomy for an image base containing D items as a faceted taxonomy with j facets. This means that the taxonomy is composed by j independent sub-taxonomies, e.g. topics, dominant colors, luminance, etc. For the sake of simplicity, assume that the set of terminal concepts T in the dynamic taxonomy is partitioned into j subsets of the same size. Under this assumption, each partition has T/j ($T/j \geq 2$) terminal concepts. Assume now that each image d is classified under one and only one leaf concept C in each partition, and that the probability of classifying d under C is uniform for each partition. The average number of images to be manually scanned R (i.e. the cardinality of the current focus F) after k zoom operations is

$$R = D \left(\frac{j}{T} \right)^k, T \geq 2j, 0 \leq k \leq j \quad (1)$$

Assume now an image base of ten million items, and a taxonomy of 1000 terminal concepts organized in 10 facets, each having 100 terminal concepts. In the case of low-level features, terminal concepts correspond to feature values (e.g. a specific level of luminance). According to (1), on the average, one zoom operation produces a focus with 100,000 images, two zooms produce a focus consisting of 1,000 images, and three zooms select 10 images.

The main indications of this analysis are two. First, we do not need many features as long as they are sufficiently uncorrelated [17] and even low-selectivity features can be used in practice. Second, the upward scalability of dynamic taxonomies is dramatic, even with compact taxonomies. In the example, a maximum of 10 zoom operations can be performed: they are sufficient to produce a focus of 10 images for an image base consisting of 10^{21} images.

4 Monodimensional vs. Multidimensional Clustering

Most current research accounts for the diversity of features by computing an overall similarity distance for the images in the database by linearly combining the similarity distance of each individual feature, such as low-level multimedia features (e.g. color) or conceptual features, such as a painter name. Clustering groups items together according to a single similarity measure, and consequently each item belongs to one and

only one cluster. We call this type of clustering a *monodimensional clustering* by analogy with classification theory. If a hierarchical clustering scheme is used, monodimensional clustering is similar to a traditional, monodimensional taxonomy.

In alternative, each feature can be considered independently: each feature will result, in general, into a different clustering scheme because, for instance, two items similar by texture may have different colors. In this case, an item will belong to different clusters. We call this *multidimensional clustering*¹.

Here we criticize the use of monodimensional clustering schemes by comparing them to multidimensional clustering schemes. By switching from monodimensional clustering to multidimensional clustering on F features, the cost of clustering increases by a factor F , because all the items have to be fully clustered for each feature. However:

1. the notion of similarity is inaccurate and ineffective in monodimensional clustering and classification, and
2. an exponential growth in the number of clusters is required if the number of items in a cluster is to be kept at a reasonable low level.

In monodimensional clustering, a given multimedia item is represented by a point in a multidimensional space, computed as the weighted combination of the item's low-level multimedia features. Similarity between any two items is then computed as the inverse of their distance in this space and depends on the weights used to combine low-level features. Consider two low-level features such as color and texture. Different users will generally use different weights to combine these features, according to their interests: user A may be more interested in color than in texture, whereas user B could be more interested in texture than in color. Different weights on these features imply different similarity functions, so that items a and b can be similar for user A and quite different for user B. However, since a single, predefined similarity function is used for clustering, the resulting clustering scheme only accommodates those users whose notion of similarity matches the predefined function. In order to accommodate other users, in this same framework, clustering should be performed dynamically on a similarity function given by the user himself. However, this is not feasible for two reasons:

1. cost, as the dynamic reclustering of large information bases requires substantial resources and time. Reclustering is required in El Niño [19]. A similar approach in textual databases, Scatter-Gather [1], is criticized in [14];
2. human factors, because it is unlikely that the average, unskilled user would be able to understand the effects of weights and hence come up with appropriate weights for different features. A similar problem occurs in product selection in e-commerce, and is discussed in [16].

Even validation of results can be seriously biased by a monodimensional clustering. In fact, weighting coefficients that combine feature similarity play a fundamental part on clustering and access by similarity, so that it may become difficult to

¹ Clustering theory is indeed defined in a multidimensional space: but in classic clustering an item a only belongs to single cluster, whereas clustering schemes based on a multidimensional classification can place the same item in different clusters, according to different similarity measures.

discriminate between the actual advantages offered by specific features and the random effects of a specific weighting scheme.

In addition, a hierarchical monodimensional clustering scheme is analogous to a traditional monodimensional taxonomy, and the analysis reported in [17] applies. In particular, the maximum resolution of a hierarchical clustering scheme with T terminal clusters is $MR=1/T$. Compare this resolution with the maximum resolution of a multidimensional scheme with the same number T of terminal clusters organized according to j facets, which is $MR=1/(T/j)^j$. The reducing power of a multidimensional clustering scheme is $j^j/T^{(j-1)}$ higher than the corresponding monodimensional scheme.

From another prospective, note that the average number of images to be manually scanned R is given by $R=D/T$. Therefore in order to have a reasonable result size, say $R \cong 10$, we need a number of terminal clusters that is just one order of magnitude less than the image collection size. From the one hand, such a high number of clusters is difficult to manage and access by end-users. From the other hand, there is no guarantee that such a fine subdivision can be produced.

Therefore, the advantages of multidimensional clustering over monodimensional scheme are analogous to those that we discussed for multidimensional vs. monodimensional taxonomies: namely, a dramatically better scalability and the ability to correlate different features. As an example, we can zoom on a specific texture cluster, and immediately see all the color clusters for that texture. In addition, custom similarities can be easily expressed.

In summary, we believe that multidimensional clustering strategies deserve close attention, because of their faster convergence and, most importantly, because they present visual similarities according to different perspectives (color, luminance, etc.), and allow for a more articulated exploration of the information base.

5 Representing Low-Level Features and Clusters

In addition to metadata, each image in the sample collection is automatically described by a number of independent low-level features. These include:

1. average image brightness
2. average image saturation
3. HSV histogram
4. clustering on average color in CIE L^*a^*b color space on a 4×4 grid

Each image was first reduced to a 250×250 size, while preserving the aspect ratio. For all features except the last, images were converted to the HSV color space, which is perceptually more accurate than the RGB color space. We record the image color histogram, using the color reduction proposed by Lei et al. [7]. A Gaussian blur is applied to the image before conversion to the HSV color space, in order to avoid noise, and HSV colors are mapped to 36 bins based on the perceptual characteristics of the color itself.

The last low-level feature records average colors. Each image axis is subdivided into 4 intervals, giving 16 areas: the average color of each area is computed. The aspect ratio is not preserved. Here, the image is first converted to the CIE L^*a^*b color

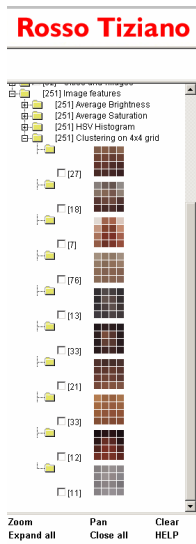


Fig. 1. Multidimensional low-level features: clustering of average color on a 4x4 grid. Clusters are labelled by their barycenter.

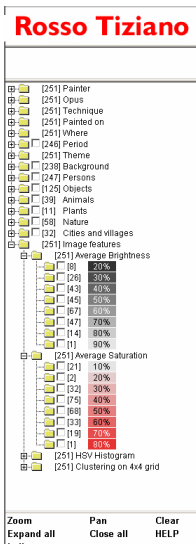


Fig. 2. Monodimensional low-level features: average brightness and average saturation

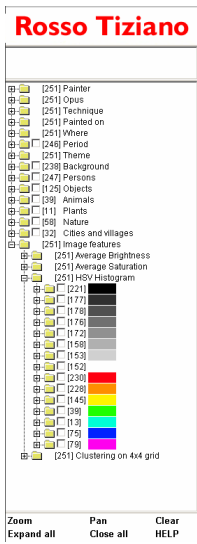


Fig. 3. Bidimensional low-level features: reduced HSV histogram

space, because linear color combinations in this space are perceptually more accurate than in other spaces. Each image is then represented by a vector on 16 dimensions and clustering is applied, based on the cosine measure. The entire collection was clustered into 10 clusters, by using an agglomerative complete-link clustering algorithm [6]. We predefined 10 clusters (see Fig. 1), a number chosen according to the guidelines from [14]: a larger collection would require a hierarchical clustering scheme.

Since the number and variety of low-level features that can be used to describe images is significant, we do not strive for an exhaustive coverage, and important features such as textures, regions, shapes, and salient points [23] are not considered here. Instead, we focus on simple features and problems common to significant classes of features.

Although the discussion in the previous sections focused on clustering, many low-level features can be represented by *a priori* fixed hierarchical organizations of the feature space. As an example, monodimensional image data such as average or dominant color, overall luminance, etc., can be easily represented by a facet with k sons, each representing a possible value. The label of each son can usually be provided by the value (color, luminance level, etc.) itself. In the example, we subdivided both average brightness and saturation in 10 intervals (figure 2). Although a much finer subdivision is supported by measurements, finer subdivisions would be useless from the perceptual point of view.

A similar organization applies also to a number of bidimensional features, such as color histograms [10]. The color histogram facet can be represented by a 2-level hierarchy, where colors are enumerated on the first level (immediate sons of the facet). Each color is further characterized, at the next level, by its normalized count, organized as range of values. The structure reported in the figures is slightly more complex than normal histograms. Here, in fact, the highest level reports principal colors (8 shades of gray and the 8 main colors, red to purple); at the lower level, each main color is subdivided into 4 actual colors, by using different luminance and saturation (figure 3).

More complex image representations such as spatial sampling of average color, salient points, etc. are not easily represented in this way. In these cases, hierarchical clustering is required, and the problem of conveying the “meaning” of each cluster has to be solved. In traditional clustering, clusters are usually labeled by their centroid: either the cluster barycenter or the item closest to it. Even with text-based clustering, the meaning of the cluster is often so unclear as to pose serious cognitive challenges to the user [14]. With image representations, these problems are considerably worse, because the image features used by clustering might be difficult to understand. In our example, we labeled clusters by their stylized barycenter rather than by the image closest to the barycenter, because a specific image might not convey the rationale used for clustering. For instance, a *tondo* (i.e. a circular painting) is rather uncommon, since the usual format is rectangular. If a *tondo* is used to label a cluster, users are likely to assume that the cluster contains *tondos*, whatever the correct interpretation might be.

6 Examples of Exploration

Figures 4 to 9 report three different explorative sessions that show the significant advantages of the current approach. In the first session, the user starts her exploration from a low-level feature, the average image brightness, and selects *dark* paintings (figure 4), i.e. paintings with a brightness of 20% or less. After the zoom operation, the reduced taxonomy in figure 5 indicates that only Antonello da Messina and Raphael painted dark paintings, and that almost all such paintings are portraits. If the user displays these portraits, he will notice that they have a black background. If you wonder why no other painters produced dark paintings, the explosion of the *Technique* topic will show you that both painters are the only masters in the collection that use oil painting. All the other masters used tempera, a technique using water and egg-yolk (instead of oil) as a binding medium. Tempera paintings tend to be much brighter.

In the second session, exploration starts from metadata: *Painter>Masaccio* and then *Theme>Sacred* are zoomed upon (figure 6). The result is then summarized according to the HSV histogram, and paintings that have an orange-ish color are displayed (figure 7). As you will notice, almost all the sacred paintings by Masaccio fall in this category: these are paintings with a golden background (the so-called *fondo oro*) that is typical of Italian sacred paintings of the fifteenth century.

Finally, in the third and last session, clustering on a 4x4 grid is used to explore Portraits by Antonello da Messina (figure 8). From the clusters in the reduced taxonomy, most portraits fall in a single cluster, which means they are visually very similar. In fact, almost all the portraits by Antonello have a very dark background (see the first session) and the face covers most of the painting (figure 9).

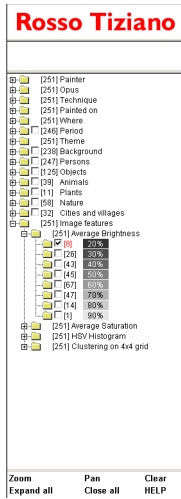


Fig. 4. Zooming on dark paintings



Fig. 5. Exploring dark paintings: only Raphael and Antonello have dark items, and almost all are portraits. Dark portraits are expanded.

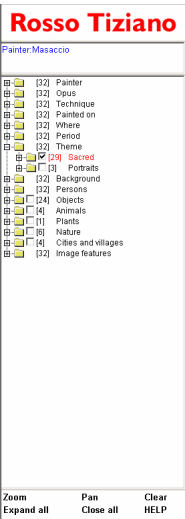


Fig. 6. Zooming on Sacred paintings after a zoom on Masaccio

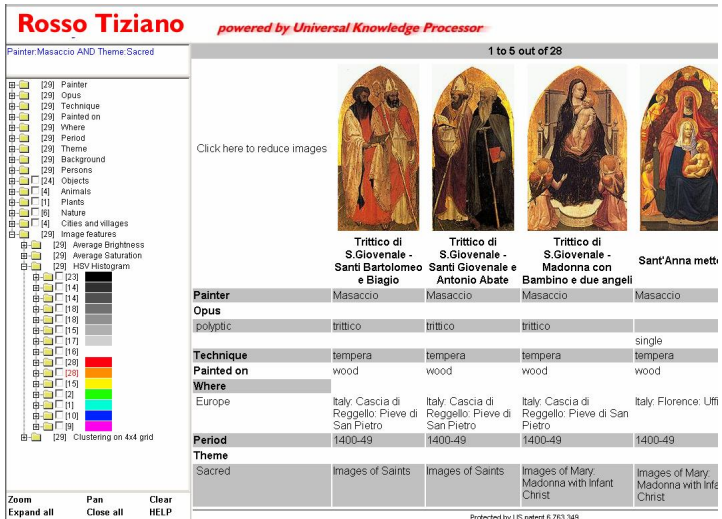


Fig. 7. Histogram summary of Masaccio's sacred paintings: paintings with orange-ish colors are displayed

Three simple and quick sessions show how the information base can be effortlessly explored, gaining insights on its contents: in fact, we discovered relationships and features of the image collection that no other access method would have made available.

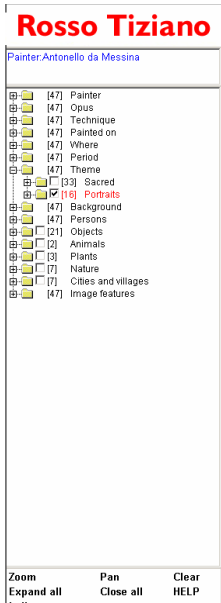


Fig. 8. Zooming on *Portraits* paintings, after a zoom on *Antonello*

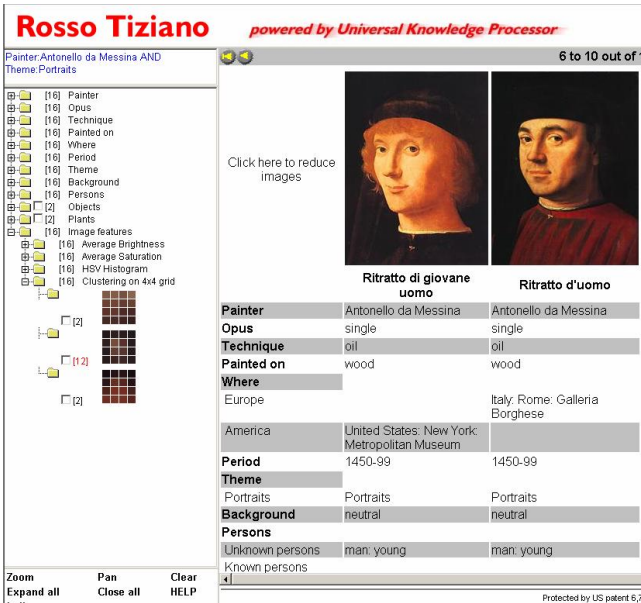


Fig. 9. Cluster summary of Antonello's portraits: displaying the selected cluster

7 Conclusions and Future Research

A unified intelligent browsing system for complex multimedia information bases was presented. Dynamic taxonomies allow users to explore complex and large multimedia information bases in a totally assisted way without unnecessary constrictions or asymmetries in search paths. They effectively integrate low-level multimedia features and conceptual metadata features in a unique, integrated visual framework, in which both features can be used to focus on and to conceptually summarize portions of the infobase.

The shift in the access paradigm has important implications on how low-level features are selected and used in the present context. Traditional CBIR systems strive to capture image “semantics” through a mix of high quality features. Here, instead, it is much more important that the features used are easily understood by users and that they capture image characteristics that are useful for exploration. In fact, we hypothesize that an array of simple features, such as the ones we used in the examples above, may be adequate, because of the very quick convergence of dynamic taxonomies, even for very large and more complex image collections. It is tempting to call RAIF (redundant array of inexpensive features) such an approach and to see a strong analogy with RAID techniques in disk technology. In both cases, the intelligent combination of very simple items (features in the present context, disks in RAID) produces a holistic result that is much better than the original components.

Although formal usability studies are required, the first informal tests on people reasonably familiar with the paintings in the information base show some interesting trends. First, most explorations start from metadata with low-level features used in later stages in order to find visual similarities among paintings characterized by semantic metadata descriptions. If confirmed by formal experiments, this would indicate that both access by metadata and by low-level features are required and must be dealt with in a uniform way. In addition, feature-only CBIR's that do not support metadata access would not seem to match user interactions and requirements.

Second, users found that the ability to see images clustered according to different and independent visual features quite important in exploring the information base, and in discovering effective visual similarities. Again, if confirmed, this would make a case for multidimensional clustering and simple, easy-to-understand low-level features.

References

1. Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W.: Scatter/Gather: a cluster-based approach to browsing large document collections. *ACM SIGIR*, 318–329 (1992)
2. Datta, R., Li, J., Wang, J.Z.: Content-Based Image Retrieval - Approaches and Trends of the New Age. In: *ACM MIR* (2005)
3. Gärdénfors, P.: *Conceptual Spaces – The Geometry of Thought*. MIT Press, Cambridge (2000)
4. Hearst, M., et al.: Finding the Flow in Web Site Search. *Comm. of the ACM* 45(9), 42–49 (2002)
5. Knowledge Processors, *The Universal Knowledge Processor* (1999), <http://www.knowledgeprocessors.com>
6. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. *ACM Comput. Surv.* 31(3), 264–323 (1999)
7. Lei, Z., Fuzong, L., Bo, Z.: A CBIR method based on color-spatial feature, *TENCON 1999*. In: *Proc of the IEEE Region 10 Conference*, pp. 166–169 (1999)
8. Martinez, J. (ed.): *MPEG 7 – Overview*, ISO/IEC JTC1/SC29/WG11 N4980 (2002)
9. McDonald, S., Tait, J.: Search strategies in content-based image retrieval. *ACM SIGIR*, 80–87 (2003)
10. Niblack, W., Barber, R., et al.: The QBIC Project: Querying Images By Content Using Color, Texture, and Shape. In: *SPIE*, vol. 1908, pp. 173–181 (1993)
11. Ranganathan, S.R.: *The Colon Classification*, vol. 4. Rutgers University Press, New Jersey (1965)
12. Sanderson, M., Croft, B.: Deriving concept hierarchies from text. *ACM SIGIR* (1999)
13. Sacco, G.M.: Navigating the CD-ROM. In: *Proc. Int. Conf. Business of CD-ROM* (1987)
14. Sacco, G.M.: Dynamic Taxonomies: A Model for Large Information Bases. *IEEE Transactions on Knowledge and Data Engineering* 12(3) (2000)
15. Sacco, G.M.: Uniform access to multimedia information bases through dynamic taxonomies. In: *IEEE 6th Int. Symp. on Multimedia Software Engineering* (2004)
16. Sacco, G.M.: The intelligent e-store: easy interactive product selection and comparison. In: *7th IEEE Conf. on E-Commerce Technology*, IEEE CEC 2005 (2005)
17. Sacco, G.M.: Analysis and Validation of Information Access through Mono, Multidimensional and Dynamic Taxonomies. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) *FQAS 2006*. LNCS (LNAI), vol. 4027, pp. 659–670. Springer, Heidelberg (2006)
18. Sacco, G.M.: Some Research Results in Dynamic Taxonomy and Faceted Search Systems. In: *SIGIR 2006 Workshop on Faceted Search* (2006)

19. Santini, S., Jain, R.: Integrated Browsing and Querying for Image Databases. *IEEE MultiMedia* 7(3), 26–39 (2000)
20. Schreiber, A.T., Dubbeldam, B., Wielemaker, J., Wielinga, B.J.: Ontology-based photo annotation. *IEEE Intelligent Systems* (2001)
21. Smeulders, A., et al.: Content-based image retrieval at the end of early years. *IEEE Trans on Pattern Analysis and Machine Intelligence* 22(12), 1349–1380 (2000)
22. Stan, D.: A New Approach for Exploration of Image Databases. In: *Proceedings of the Grace Hopper Celebration of Women in Computing* (2002)
23. Tian, Q., et al.: Image retrieval using wavelet-based salient points. *Journal of Electronic Imaging* 10(4), 835–849 (2001)
24. Yee, K., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: *Proc ACM SIGCHI Conf. CHI 2003* (2003)

NM-Tree: Flexible Approximate Similarity Search in Metric and Non-metric Spaces

Tomáš Skopal and Jakub Lokoč

Charles University in Prague, FMP, Department of Software Engineering,
Malostranské nám. 25, 118 00 Prague, Czech Republic
{skopal,lokoč}@ksi.mff.cuni.cz

Abstract. So far, an efficient similarity search in multimedia databases has been carried out by metric access methods (MAMs), where the utilized similarity measure had to satisfy the metric properties (reflexivity, non-negativity, symmetry, triangle inequality). Recently, the introduction of TriGen algorithm (turning any nonmetric into metric) enabled MAMs to perform also nonmetric similarity search. Moreover, it simultaneously enabled faster approximate search (either metric or nonmetric). However, a simple application of TriGen as the first step before MAMs' indexing assumes a fixed “approximation level”, that is, a user-defined tolerance of retrieval precision is preset for the whole index lifetime. In this paper, we push the similarity search forward; we propose the NM-tree (nonmetric tree) – a modification of M-tree which natively aggregates the TriGen algorithm to support *flexible approximate* nonmetric or metric search. Specifically, at query time the NM-tree provides a user-defined level of retrieval efficiency/precision trade-off. We show the NM-tree could be used for general (non)metric search, while the desired retrieval precision can be flexibly tuned on-demand.

1 Introduction

As the digital devices for capturing multimedia data become massively available, the similarity search in multimedia databases steadily becomes more important. The metadata-based searching (using text/keywords/URL attached to multimedia documents, e.g., as at images.google.com) provides either limited search capabilities or even is not applicable (for raw data). On the other hand, the *content-based similarity retrieval* provides a native solution. The multimedia objects are retrieved based on their similarity to a query object (i.e., we suppose the *query-by-example* modality). The similarity measure is domain-specific – we could measure similarity of two images based on, for example, color histogram, texture layout, shape, or any combination. In most applications the similarity measure is regarded as computationally expensive.

In order to search a multimedia database efficiently enough, the database has to be indexed so that the volume of explicitly computed similarity scores to answer a query is minimized. That is, we try to avoid sequential scan over all the objects in the database, and their comparing against the query object.

1.1 Metric Search

A few decades ago, the database-oriented research established a metric-based class of access methods for similarity search – the *metric access methods* (MAMs). The similarity measure δ (dissimilarity or distance, actually) is modeled by a metric distance function, which satisfies the properties of reflexivity, non-negativity, symmetry and triangle inequality. Based on these properties, the MAMs partition (or index) the metric data space into classes, so that only some of the classes have to be searched when querying; this results in a more efficient retrieval. To date, many MAMs were developed, addressing various aspects – main-memory/database-friendly methods, static/dynamic indexing, exact/approximate search, centralized/distributed indexing, etc. (see [21,15,4]). Although efficient in query processing, MAMs force their users to employ just the metric similarity measures, which is becoming a serious limitation nowadays.

1.2 Nonmetric Similarity

As the quantity/complexity of multimedia data grows, there is a need for more complex similarity measuring. Here the metric model exhibits its drawbacks, since the domain experts (being not computer scientists) are forced to “implant” metric properties into their nonmetric measures, which is often impossible.

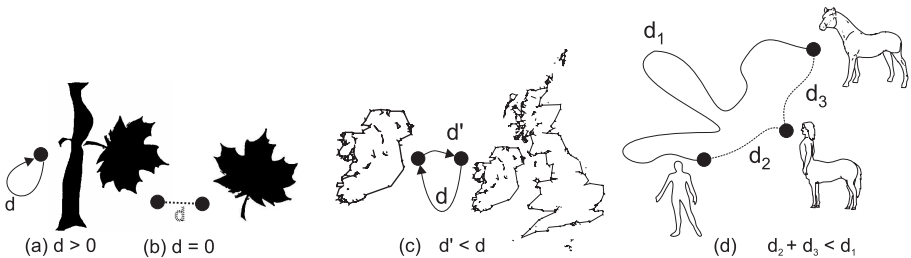


Fig. 1. Objections against metric properties in similarity measuring: (a) reflexivity (b) non-negativity (c) symmetry (d) triangle inequality

However, a nonmetric similarity has also a qualitative justification. In particular, the reflexivity and non-negativity have been refuted by claiming that different objects could be differently self-similar [11,19]. For example, in Figure 1a the leaf on a trunk can be viewed as positively self-dissimilar if we consider the less similar parts of the objects (here the trunk and the leaf). Or, alternatively, in Figure 1b the leaf-on-trunk and leaf could be treated as identical if we consider the most similar parts of the objects (the leaves). The symmetry was questioned by showing that a prototypical object can be less similar to an indistinct one than vice versa [13,14]. In Figure 1c, the more prototypical “Great Britain and Ireland” is more distant to the “Ireland alone” than vice versa. The triangle inequality is the most attacked property. Some theories point out the similarity has not to be transitive [1,20]. Demonstrated by the well-known example, a

man is similar to a centaur, the centaur is similar to a horse, but the man is completely dissimilar to the horse (see Figure 1d).

1.3 Related Work

When compared to the rich research output in the area of metric access methods, there exist only few approaches to efficient nonmetric search. They include mapping methods [2,12,7], where the nonmetric space is turned into a vector space (mostly Euclidean). The distances in the target space are preserved more or less approximately, while the approximation error is fixed and/or not known. Similar approximate results achieve classification techniques [8,10]. Recently, an exact search over a nonmetric database was introduced, assuming the query distribution is restricted to the database distribution [5], while the indexing is very expensive (all pairwise distances on database objects must be computed).

Universal Solution. In our previous work we have introduced the TriGen algorithm [17,16] – a universal approach to searching in (non)metric spaces – where we addressed exact metric search, approximate metric search, (almost) exact nonmetric search and approximate nonmetric search. All these retrieval types can be achieved by turning the input (non)metric δ into a distance which satisfies the triangle inequality to some degree (including the full metric case). Such a modified measure can be used by any MAM for indexing/querying. However, as the proposed solution separates the measure conversion from the subsequent MAM-related issues, querying on an index built using the modified measure can be used to retrieval that is unchangeable in retrieval precision, that is, a retrieval always exact or always approximate to some fixed extent. If the user wants to adjust the desired precision, the entire database has to be reindexed.

Paper Contribution. In this paper we propose the NM-tree, a nonmetric (and also metric) access method extending the M-tree. The NM-tree natively utilizes the TriGen algorithm and provides retrieval precision adjustable at query time.

2 TriGen

The metric access methods are efficient because they use metric properties to index the database, especially the triangle inequality. However, in nonmetric spaces a dissimilarity measure δ is not constrained by any properties, so we have no clue for indexing. A way how to enable efficient nonmetric search is a transformation to the (nearly) metric case by so-called T-bases. The reflexivity, non-negativity and symmetry can be easily added to any nonmetric δ used for similarity search (see [17]). We also assume the values produced by δ are scaled into $\langle 0, 1 \rangle$, which is achieved for free when fixing the reflexivity and non-negativity.

The hard task is enforcing the triangle inequality. The TriGen algorithm [17,16] can put more or less of the triangle inequality into any *semimetric* δ (i.e., into any reflexive, non-negative, symmetric distance), thus any semimetric distance can be turned into an equivalent full metric¹, or to a semimetric which

¹ In fact, the metric preserves the original query orderings (which is sufficient [17]).

satisfies the triangle inequality to some user-defined extent. Conversely, TriGen can also turn any full metric into a semimetric which preserves the triangle inequality only partially; this is useful for faster but only approximate search. For its functionality the TriGen needs a (small) sample of the database objects.

2.1 T-Bases

The principle behind TriGen is a usage of triangle triplets and T-bases. A triplet of numbers (a, b, c) is *triangle triplet* if $a + b \geq c, b + c \geq a, a + c \geq b$. The triangle triplets can be viewed as witnesses of triangle inequality of a distance δ – if all triplets $(\delta(O_i, O_j), \delta(O_j, O_k), \delta(O_i, O_k))$ on all possible objects O_i, O_j, O_k are triangle triplets, then δ satisfies the triangle inequality. Using triangle triplets we measure the *T-error* – a degree of triangle inequality violation, computed as the proportion of non-triangle triplets in all examined distance triplets.

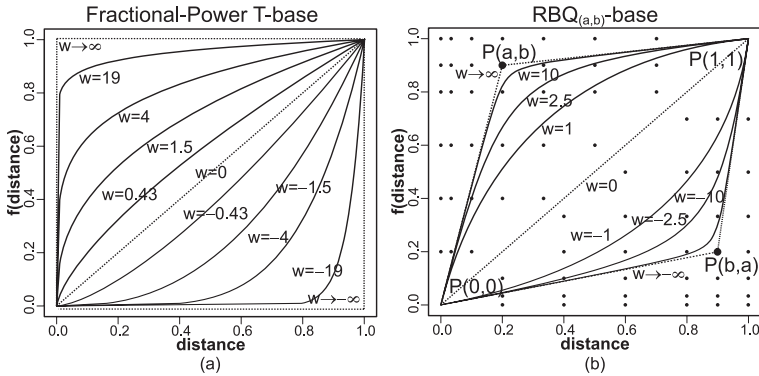


Fig. 2. T-bases: (a) FP-base (b) RBQ(a,b)-base; for their formulas see [17]

A *T-base* $f(x, w)$ is an increasing function (where $f(0, w) = 0$ & $f(1, w) = 1$) which turns a value $x \in \langle 0, 1 \rangle$ of an input (semi)metric δ into a value of a target (semi) metric δ^f , i.e., $\delta^f(\cdot, \cdot) = f(\delta(\cdot, \cdot), w)$. Besides the input distance value x , the T-base is parameterized also by a fixed weight $w \in \langle -\infty, \infty \rangle$ which determines how concave or convex f should be. The higher $w > 0$, the more concave f , which means also the lower T-error of any δ^f . Conversely, the lower $w < 0$, the more convex f and the higher T-error of any δ^f ($w = 0$ means f is identity). For example, in Figure 2 see two T-bases, the *fractional power T-base* (FP-base) and one of the *rational Bézier quadratic T-bases* (RBQ-bases).

2.2 Intrinsic Dimensionality

When choosing very high w (i.e., very concave f), we could turn virtually any semimetric δ into a full metric δ^f . However, such a modification is not very useful. The more concave f , the higher *intrinsic dimensionality* [4,3] of the data space – a characteristic related to the mean and variance computed on the set of

pairwise distances within the data space. Simply, a high intrinsic dimensionality of the data leads to poor partitioning/indexing by any MAM (resulting in slower searching), and vice versa. On the other hand, the more convex f , the lower intrinsic dimensionality of the data space but also the higher the T-error – this results in fast but only approximate searching, because now the MAMs’ assumption on fully preserved triangle inequality is incorrect. Hence, we have to make a trade-off choice – whether to search quickly but only approximately using a dissimilarity measure with higher T-error, or to search slowly but more precisely.

2.3 The TriGen Algorithm

Given a user-defined T-error tolerance θ , a sample $\mathcal{S}$ of the database, and an input (semi)metric δ , the TriGen’s job is to find a modifier f so that the T-error of δ^f is kept below θ and the intrinsic dimensionality of $(\mathcal{S}, \delta^f)$ is minimized². For each of the predefined T-bases the minimal w is found (by halving the weight interval), so that the weight w cannot be further decreased without T-error exceeding θ . Among all the processed T-bases and their final weights, the one is chosen which exhibits the lowest intrinsic dimensionality, and returned by TriGen as the *winning T-modifier* (for details of TriGen see [17]).

The winning T-modifier could be subsequently employed by any MAM to index and query the database using δ^f . However, at this moment a MAM’s index built using δ^f is not usable if we want to change the approximation level (the T-error of δ^f), that is, to use another f . This is because MAMs accept the distance δ^f as a black box; they do not know it is a composition of δ and f . In such case we have to throw the index away and reindex the database by a δ^{f_2} .

In this paper we propose the NM-tree, a MAM based on M-tree natively utilizing TriGen. In NM-tree, any of the precomputed T-modifiers f_i can be flexibly chosen at query time, allowing the user to trade performance for precision.

3 M-Tree

The *M-tree* [6] is a dynamic metric access method that provides good performance in database environments. The M-tree index is a hierarchical structure, where some of the data objects are selected as centers (references or local *pivots*) of ball-shaped regions, and the remaining objects are partitioned among the regions in order to build up a balanced and compact hierarchy, see Figure 3a. Each region (subtree) is indexed recursively in a B-tree-like (bottom-up) way of construction. The inner nodes of M-tree store *routing entries*

$$rout_l(O_i) = [O_i, r_{O_i}, \delta(O_i, Par(O_i)), ptr(T(O_i))]$$

where O_i is a data object representing the center of the respective ball region, r_{O_i} is a *covering radius* of the ball, $\delta(O_i, Par(O_i))$ is so-called *to-parent* distance

² As shown in [17,16], the real retrieval error of a MAM using δ^f is well estimated by the T-error of δ^f , hence, θ can be directly used as a retrieval precision threshold.

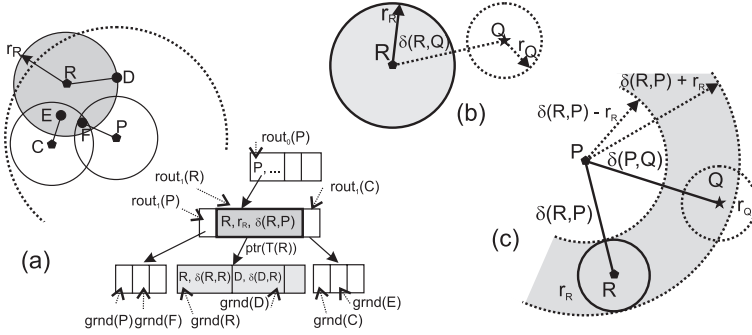


Fig. 3. (a) M-tree (b) Basic filtering (c) Parent filtering

(the distance from O_i to the object of the parent routing entry), and finally $ptr(T(O_i))$ is a pointer to the entry's subtree. The data is stored in the leaves of M-tree. Each leaf contains *ground entries*

$$grnd(O_i) = [O_i, id(O_i), \delta(O_i, Par(O_i))]$$

where O_i is the data object itself (externally identified by $id(O_i)$), and $\delta(O_i, Par(O_i))$ is, again, the to-parent distance. See an example of entries in Figure 3a.

3.1 Query Processing

The range and k nearest neighbors (kNN) queries are implemented by traversing the tree, starting from the root³. Those nodes are accessed whose parent regions (R, r_R) described by the routing entry are overlapped by the query ball (Q, r_Q) .

In case of a kNN query (we search for k closest objects to Q), the query radius (or range) r_Q is not known in advance, so we have to additionally employ a heuristic to dynamically decrease the radius during the search. The radius is initially set to the maximum distance in the metric space, that is, to 1.0 since we have assumed a dissimilarity measure scaled to $\langle 0, 1 \rangle$, see Section 2.

Basic filtering. The check for region-and-query overlap requires an explicit distance computation $\delta(R, Q)$, see Figure 3b. In particular, if $\delta(R, Q) \leq r_Q + r_R$, the data ball R overlaps the query ball, thus the child node has to be accessed. If not, the respective subtree is filtered from further processing.

Parent filtering. As each node in the tree contains the distances from the routing/ground entries to the center of its parent node, some of the non-relevant M-tree branches can be filtered out without the need of a distance computation, thus avoiding the “more expensive” basic overlap check (see Figure 3c). In particular, if $|\delta(P, Q) - \delta(P, R)| > r_Q + r_R$, the data ball R cannot overlap the query ball, thus the child node has not to be re-checked by basic filtering. Note $\delta(P, Q)$ was computed in the previous (unsuccessful) parent’s basic filtering.

³ We outline just the principles, for details see the original M-tree algorithms [6,18].

4 NM-Tree

The NM-tree is an extension of M-tree in terms of algorithms, while the data structure itself is unchanged. The difference in indexing relies in encapsulating the M-tree insertion algorithm by the more general NM-tree insertion (see Section 4.1). The query algorithms have to be slightly redesigned (see Section 4.2).

4.1 Indexing

The distance values (to-parent distances and covering radii) stored in NM-tree are all metric, that is, we construct a regular M-tree using a full metric. Since the NM-tree's input distance δ is generally a semimetric, the TriGen algorithm must be applied before indexing, in order to turn δ into a metric δ^{f_M} (i.e., searching for a T-modifier f_M under $\theta = 0$). However, because at the beginning of indexing the NM-tree is empty, there is no database sample available for TriGen. Therefore, we distinguish two phases of indexing and querying on NM-tree.

Algorithm 1 (*dynamic insertion into NM-tree*)

```

method InsertObject( $O_{new}$ ) {
  if database size < smallDBlimit then
    store  $O_{new}$  into sequential file
  else
    insert  $O_{new}$  into NM-tree (using original M-tree insertion algorithm under  $\delta^{f_M}$ )
  endif
  if database size = smallDBlimit then
    run TriGen algorithm on the database, having  $\theta_M = 0$ ,  $\theta_1, \theta_2, \dots, \theta_k > 0 \Rightarrow$ 
      obtaining T-bases  $f_M, f_{e_1}, f_{e_2}, \dots, f_{e_k}$  with weights  $w_M, w_{e_1}, w_{e_2}, \dots, w_{e_k}$ 
    for each object  $O_i$  in the sequential file
      insert  $O_i$  into NM-tree (using original M-tree insertion algorithm under  $\delta^{f_M}$ )
    empty the sequential file
  end if }

```

For the whole picture of indexing in NM-tree, see Algorithm 1. The first phase just gathers database objects until we get a sufficiently large set of database objects. In this phase a possible query is solved sequentially, but this is not a problem because the indexed database is still small. When the database size reaches a certain volume (say, $\approx 10^4$ objects, for example), the TriGen algorithm is used to compute f_M using the database obtained so far. At this moment we run the TriGen also for other, user-defined θ_i values, so that alternative T-modifiers will be available for future usage (for approximate querying). Finally, the first phase is terminated by indexing the gathered database using a series of the original M-tree insertions under the metric δ^{f_M} (instead of δ). In the second phase the NM-tree simply forwards the insertions to the underlying M-tree.

Notice: In contrast to the original TriGen [17], in NM-tree we require the T-bases f_i to be additionally *inversely symmetric*, that is, $f_i(f_i(x, w), -w) = x$. In other words, when knowing a T-base f_i with some weight w , we know also the inverse $f_i^{-1}(\cdot, w)$, which is determined by the same T-base and $-w$. The FP-base and all RBQ-bases (see Section 2.1) are inversely symmetric.

4.2 Query Processing

When querying, we distinguish two cases – exact search and approximate search.

Exact search. The exact case is simple, when the user issues a query with zero desired retrieval error, the NM-tree is searched by the original M-tree algorithms, because of employing δ^{f_M} for searching, which is the full metric used also for indexing. The original user-specified radius r_Q of a range query (Q, r_Q) must be modified to $f_M(r_Q)$ before searching. After the query result is obtained, the distances of the query object Q to the query result objects O_i must be modified inversely, that is, to $f_M^{-1}(\delta^{f_M}(Q, O_i))$ (regardless of range or kNN query).

Approximate search. The approximate case is more difficult, while here the main qualitative contribution of NM-tree takes its place. Consider user issues a query which has to be processed with a retrieval error $e_i \in \langle 0, 1 \rangle$, where for $e_i = 0$ the answer has to be precise (with respect to the sequential search) and for $0 > e_i \geq 1$ the answer may be more or less approximate. The e_i value must be one of the T-error tolerances θ_i predefined before indexing (we suppose the T-error models the actual retrieval error, i.e., $e_i = \theta_i$).

An intuitive solution for approximate search would be a modification of the required δ^{f_M} -based distances/radii stored in NM-tree into $\delta^{f_{e_i}}$ -based distances/radii. In such case we would actually get an M-tree indexed by $\delta^{f_{e_i}}$, as used in [17], however, a dynamic one – a single NM-tree index would be interpreted as multiple M-trees indexed by various $\delta^{f_{e_i}}$ distances. Unfortunately, this “online interpretation” is not possible, because NM-tree (M-tree, actually) stores not only direct distances between two objects (the to-parent distances) but also radii, which consist of aggregations. In other words, except for the two deepest levels (leaf and pre-leaf level), the radii stored in routing entries are composed from two or more direct distances (a consequence of node splitting). To correctly re-modify a radius into the correct one, we would need to know all the components in the radius, but these are not available in the routing entry.

Instead of “emulating” multiple semimetric M-trees as mentioned above, we propose a technique performing the exact metric search at higher levels and approximate search just at the leaf and pre-leaf level. In Figure 4 see all the distances/radii which are modified to semimetric ones during the search, while the modification is provided by T-bases associated with their user-defined retrieval errors. Besides the to-parent distances, we also consider the query radius and covering radii at the pre-leaf level, because these radii actually represent real distances to a furthest object in the respective query/data region. The query radius and entry-to-query distances (computed as $\delta(\cdot, \cdot)$) are not stored in NM-tree, so these are modified simply by f_{e_i} (where f_{e_i} is a T-base modifier obtained for retrieval error e_i). The remaining distances stored in NM-tree ($\delta^{f_M}(\cdot, \cdot)$)-based to-parent distances and covering radii, have to be modified back to the original ones and then re-modified using f_{e_i} , that is, $f_{e_i}(f_M^{-1}(\delta^{f_M}(\cdot, \cdot)))$.

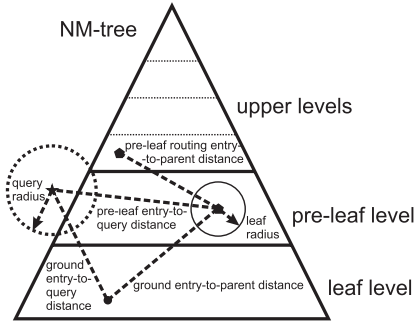
Algorithm 2 (*NM-tree range query*)

```

RangeQuery(Node  $N$ , RQuery  $(Q, r_Q)$ , retrieval error  $e_k$ ) {
  let  $O_p$  be the parent routing object of  $N$  // if  $N$  is root then  $\delta(O_i, O_p) = \delta(O_p, Q) = 0$ 

  if  $N$  is not a leaf then {
    if  $N$  is at pre-leaf level then {                                     // pre-leaf level
      for each  $rouT(O_i)$  in  $N$  do {
        if  $|f_{e_k}(\delta(O_p, Q)) - f_{e_k}(f_M^{-1}(\delta^{f_M}(O_i, O_p)))| \leq f_{e_k}(r_Q) + f_{e_k}(f_M^{-1}(r_{O_i}^{f_M}))$  then { // (parent filt.)
          compute  $\delta(O_i, Q)$ 
          if  $f_{e_k}(\delta(O_i, Q)) \leq f_{e_k}(r_Q) + f_{e_k}(f_M^{-1}(r_{O_i}^{f_M}))$  then // (basic filtering)
            RangeQuery( $ptr(T(O_i))$ ,  $(Q, r_Q)$ ,  $e_k$ )
          }
        } // for each ...
      } else {                                                         // higher levels
        for each  $rouT(O_i)$  in  $N$  do {
          if  $|f_M(\delta(O_p, Q)) - \delta^{f_M}(O_i, O_p)| \leq f_M(r_Q) + r_{O_i}^{f_M}$  then { // (parent filtering)
            compute  $\delta(O_i, Q)$ 
            if  $f_M(\delta(O_i, Q)) \leq f_M(r_Q) + r_{O_i}^{f_M}$  then // (basic filtering)
              RangeQuery( $ptr(T(O_i))$ ,  $(Q, r_Q)$ ,  $e_k$ )
            }
          } // for each ...
        }
      } else {                                                         // leaf level
        for each  $grnd(O_i)$  in  $N$  do {
          if  $|f_{e_k}(\delta(O_p, Q)) - f_{e_k}(f_M^{-1}(\delta^{f_M}(O_i, O_p)))| \leq f_{e_k}(r_Q)$  then { // (parent filtering)
            compute  $\delta(O_i, Q)$ 
            if  $\delta(O_i, Q) \leq r_Q$  then
              add  $O_i$  to the query result
            }
          } // for each ...
        }
      }
    }
  }
}

```

**Fig. 4.** Dynamically modified distances when searching approximately

In Algorithm 2 see the modified range query algorithm⁴. In the pseudocode the “metrized” distances/radii stored in the index are denoted as $\delta^{f_M}(\cdot, \cdot)$, $r_{O_i}^{f_M}$, while an “online” distance/radius modification is denoted as $f_{e_k}(\cdot)$, $f_M^{-1}(\cdot)$. If removed f_M , f_M^{-1} , f_{e_k} from the pseudocode, we would obtain the original M-tree range query, consisting of parent and basic filtering steps (see Section 3.1).

⁴ For the lack of space we omit the kNN algorithm, however, the modification is similar.

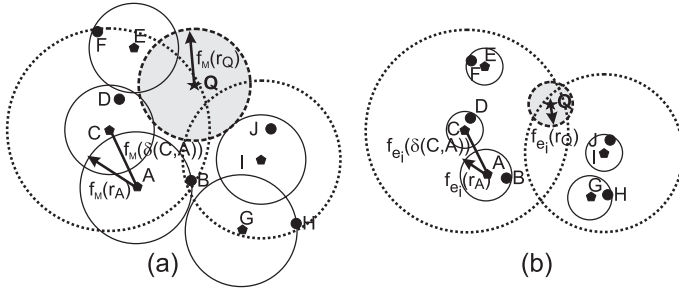


Fig. 5. (a) Exact (metric) search (b) Approximate search

In Figure 5 see a visualization of exact and approximate search in NM-tree. In the exact case, the data space is inflated into a (nearly) metric space, so the regions tend to be huge and overlap each other. On the other hand, for approximate search the leaf regions (and the query region) become much tighter, the overlaps are less frequent, so the query processing becomes more efficient.

5 Experimental Results

To examine the NM-tree capabilities, we performed experiments with respect to the efficiency and retrieval error, when compared with multiple M-trees (each using a fixed modification of δ , related to a user-defined T-error tolerance). We have focused just on the querying, since the NM-tree’s efficiency of indexing is the same as that of M-tree. The query costs were measured as the number of δ computations needed to answer a query. Each query was issued 200 times for different query objects and the results were averaged. The precision of approximate search was measured as the real retrieval error (instead of just T-error); the normed overlap distance E_{NO} between the query result QR_{NMT} returned by the NM-tree (or M-tree) and the correct query result QR_{SEQ} obtained by sequential search of the database, i.e. $E_{NO} = 1 - \frac{|QR_{NMT} \cap QR_{SEQ}|}{\max(|QR_{NMT}|, |QR_{SEQ}|)}$.

5.1 The TestBed

We have examined 4 dissimilarity measures on two databases (images, polygons), while the measures δ were considered as black-box semimetrics. All the measures were normed to $\langle 0, 1 \rangle$. The database of images consisted of 68,040 32-dimensional *Corel features* [9] (the color histograms were used). We have tested one semimetric and one metric on the images: the $L_{\frac{3}{4}}$ distance [17] (denoted **L0.75**), and the Euclidean distance (**L2**). As the second, we created a synthetic dataset of 250,000 2D polygons, each consisting of 5 to 15 vertices. We have tested one semimetric and one metric on the polygons: the dynamic time warping distance with the L_2 inner distance on vertices [17] (denoted **DTW**) and the Hausdorff distance, again with the L_2 inner distance on vertices [17] (denoted **Hausdorff**).

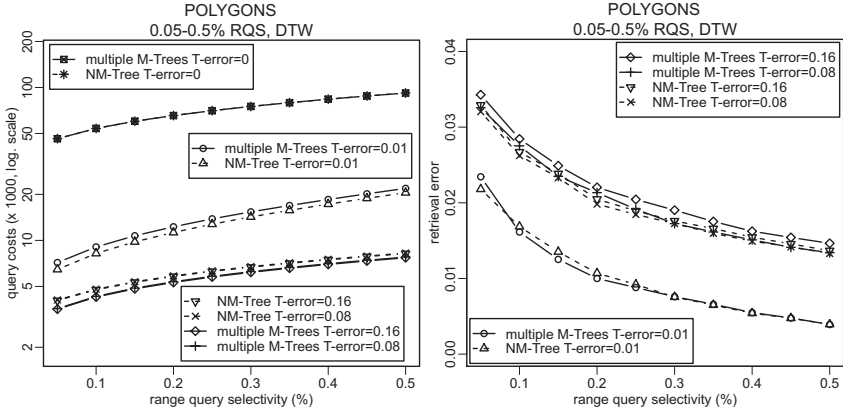
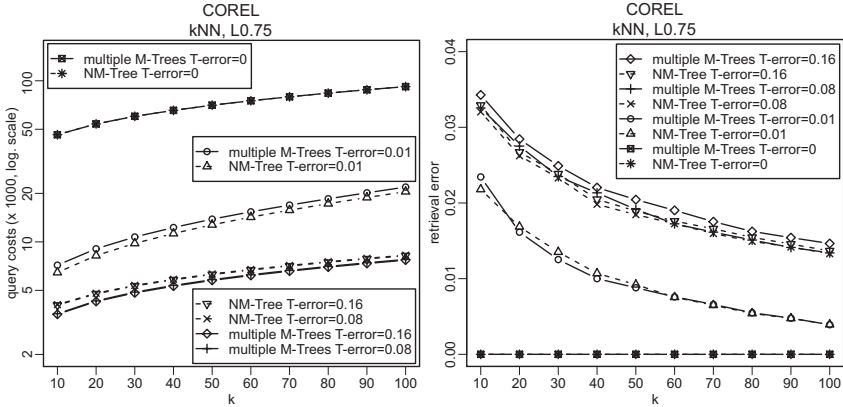


Fig. 6. Range queries on Polygons under DTW

Fig. 7. kNN queries on Corel under $L_{3/4}$

The TriGen inside NM-tree was configured as follows: the T-base pool consisting of the FP-base and 115 RBQ-bases (as in [17]), sample size 5% of Corel, 1% of Polygons. The NM-tree construction included creation of $4 \cdot 10 = 40$ T-modifiers by TriGen (concerning all the dissimilarity measures used), defined by T-error tolerances [0, 0.0025, 0.005, 0.01, 0.015, 0.02, 0.04, 0.08, 0.16, 0.32] used by querying. The node capacity of (N)M-tree was set to 30 entries per node (32 per leaf); the construction method was set to SingleWay [18]. The (N)M-trees had 4 levels (leaf + pre-leaf + 2 higher) on both Corel and Polygons databases. The leaf/inner nodes were filled up to 65%/69% (on average).

5.2 Querying

In the first experiment we have examined query costs and retrieval error of range queries on Polygons under DTW, where the range query selectivity (RQS) ranged

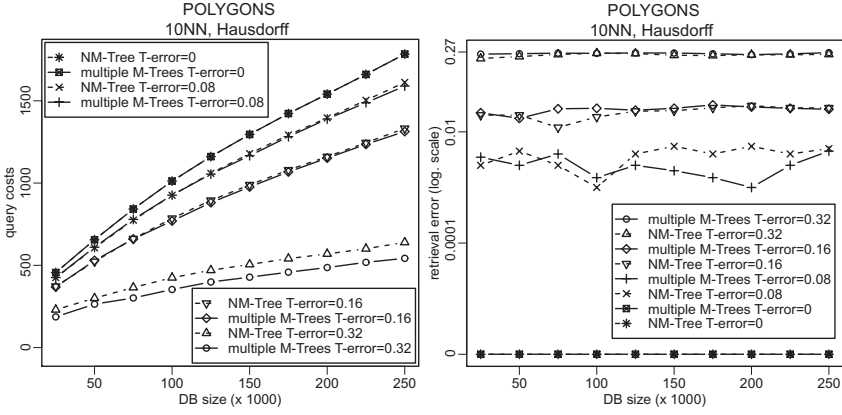


Fig. 8. 10NN queries on varying size of Polygons under Hausdorff

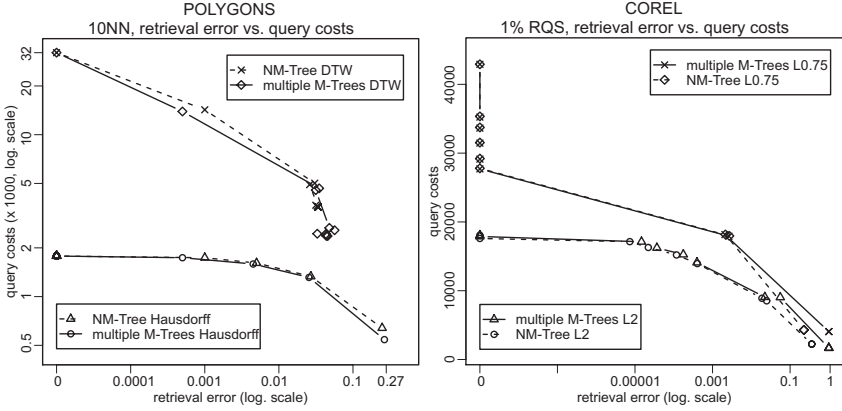


Fig. 9. Aggregated performance of 10NN queries for Polygons and Corel

from 0.05% to 0.5% of the database size (i.e., 125–1250 objects), see Figure 6. We can see that a single NM-tree index can perform as good as multiple M-tree indexes (each M-tree specifically created for a user-defined retrieval error). In most cases the NM-tree is even slightly better in both observed measurements – query costs and retrieval error.

Note that here the NM-tree is an order of magnitude faster than sequential file when performing exact (nonmetric!) search, and even two orders of magnitude faster in case of approximate search (while keeping the retrieval error below 1%). The Figure 6 also shows that if the user allows a retrieval error as little as 0.5–1%, the NM-tree can search the Polygons an order of magnitude faster, when compared to the exact (N)M-tree search.

In the second experiment we have observed the query costs and retrieval error for kNN queries on the Corel database under nonmetric $L_{\frac{3}{4}}$ (see Figure 7). The

results are very similar to the previous experiment. We can also notice (as in the first experiment) that with increasing query result the retrieval error decreases.

Third, we have observed 10NN queries on Polygons under Hausdorff, with respect to the growing database size, see Figure 8. The query costs growth is slightly sub-linear for all indexes, while the retrieval errors remain stable. Note the T-error tolerance levels (attached to the labels in legends) specified as an estimation of the maximal acceptable retrieval error are apparently a very good model for the retrieval error.

In the last experiment (see Figure 9) we have examined the aggregated performance of 10NN queries for both Polygons and Corel and all the dissimilarity measures. These summarizing results show the trade-off between query costs and retrieval error achievable by an NM-tree (and the respective M-trees).

6 Conclusions

We have introduced the NM-tree, an M-tree-based access methods for exact and approximate search in metric and nonmetric spaces, which incorporates the TriGen algorithm to provide nonmetric and/or approximate search. The main feature on NM-tree is its flexibility in approximate search, where the user can specify the approximation level (acceptable retrieval error) at query time. The experiments have shown that a single NM-tree index can search as fast as if used multiple M-tree indexes (each built for a certain approximation level). From the general point of view, the NM-tree, as the only access method for flexible exact/approximate nonmetric similarity search can achieve up to two orders of magnitude faster performance, when compared to the sequential search.

Acknowledgments

This research has been partially supported by Czech grants: "Information Society program" number 1ET100300419 and GAUK 18208.

References

1. Ashby, F., Perrin, N.: Toward a unified theory of similarity and recognition. *Psychological Review* 95(1), 124–150 (1988)
2. Athitsos, V., Hadjieleftheriou, M., Kollios, G., Sclaroff, S.: Query-sensitive embeddings. In: *SIGMOD 2005: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 706–717. ACM Press, New York (2005)
3. Chávez, E., Navarro, G.: A Probabilistic Spell for the Curse of Dimensionality. In: Buchsbaum, A.L., Snoeyink, J. (eds.) *ALENEX 2001*. LNCS, vol. 2153, pp. 147–160. Springer, Heidelberg (2001)
4. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. *ACM Computing Surveys* 33(3), 273–321 (2001)
5. Chen, L., Lian, X.: Efficient similarity search in nonmetric spaces with local constant embedding. *IEEE Transactions on Knowledge and Data Engineering* 20(3), 321–336 (2008)

6. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: VLDB 1997. LNCS, vol. 1263, pp. 426–435 (1997)
7. Farago, A., Linder, T., Lugosi, G.: Fast nearest-neighbor search in dissimilarity spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(9), 957–962 (1993)
8. Goh, K.-S., Li, B., Chang, E.: DynDex: a dynamic and non-metric space indexer. In: ACM Multimedia (2002)
9. Hettich, S., Bay, S.: The UCI KDD archive (1999), <http://kdd.ics.uci.edu>
10. Jacobs, D., Weinshall, D., Gdalyahu, Y.: Classification with nonmetric distances: Image retrieval and class representation. *IEEE Pattern Analysis and Machine Intelligence* 22(6), 583–600 (2000)
11. Krumhansl, C.L.: Concerning the applicability of geometric models to similar data: The interrelationship between similarity and spatial density. *Psychological Review* 85(5), 445–463 (1978)
12. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. *Psychometrika* 29(1), 1–27 (1964)
13. Rosch, E.: Cognitive reference points. *Cognitive Psychology* 7, 532–547 (1975)
14. Rothkopf, E.: A measure of stimulus similarity and errors in some paired-associate learning tasks. *J. of Experimental Psychology* 53(2), 94–101 (1957)
15. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, San Francisco (2006)
16. Skopal, T.: On fast non-metric similarity search by metric access methods. 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. 718–736. Springer, Heidelberg (2006)
17. Skopal, T.: Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Transactions on Database Systems* 32(4), 1–46 (2007)
18. Skopal, T., Pokorný, J., Krátký, M., Snášel, V.: Revisiting M-tree Building Principles. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 148–162. Springer, Heidelberg (2003)
19. Tversky, A.: Features of similarity. *Psychological review* 84(4), 327–352 (1977)
20. Tversky, A., Gati, I.: Similarity, separability, and the triangle inequality. *Psychological Review* 89(2), 123–154 (1982)
21. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach (Advances in Database Systems). Springer, Secaucus (2005)

Efficient Processing of Nearest Neighbor Queries in Parallel Multimedia Databases

Jorge Manjarrez-Sanchez, José Martinez, and Patrick Valduriez

INRIA and LINA, Université de Nantes

manjarrez@univ-nantes.fr, jose.martinez@univ-nantes.fr,
Patrick.Valduriez@inria.fr

Abstract. This paper deals with the performance problem of nearest neighbor queries in voluminous multimedia databases. We propose a data allocation method which allows achieving a $O(\sqrt{n})$ query processing time in parallel settings. Our proposal is based on the complexity analysis of content based retrieval when it is used a clustering method. We derive a valid range of values for the number of clusters that should be obtained from the database. Then, to efficiently process nearest neighbor queries, we derive the optimal number of nodes to maximize parallel resources. We validated our method through experiments with different high dimensional databases and implemented a query processing algorithm for full k nearest neighbors in a shared nothing cluster.

1 Introduction

Modern digital technologies are producing, in different domains, vast quantities of multimedia data, thus making data storage and retrieval critical. Retrieval of multimedia data can be modeled in a space where both the queries and the database objects are mapped into a multidimensional abstract representation, using signal processing and analysis techniques. Each dimension is a describing feature of the content. For instance, an image can be represented as a 256 dimensional feature vector where each value corresponds to a bin in a color histogram [12].

Content Based Retrieval (CBR) of multimedia objects is performed in a query by example approach. With CBR, the user supplies an example of the desired object which is used to find a set of the most similar objects from the database. This winning set can be obtained by a range approach or a best match approach. In the range approach, a threshold value is used to indicate the permissible similarity (or dissimilarity) of any object in the database from the query object and all qualifying objects within this range are returned. The similarity between the multidimensional representation of the query and each database object is estimated by computing their distance, usually the Euclidean distance (L_2). For the best match approach, the winning set is restricted to the *k-nearest neighbors (kNN)*[22]. *kNN* is the most useful kind of query in CBR. Conceptually, *kNN* query processing requires computing the distance from every feature vector in the database to the feature vector of the object example to determine the top best similarities. Obviously, performing a sequential scan of the database for CBR can be very inefficient for large data sets. Two main approaches have been proposed to avoid sequential scan and to speed up CBR: multimedia indexing and clustering.

Indexing of the feature vectors helps to speed up the search process. Using a data space or space partitioning approach [8], the feature vectors are indexed so that non interesting regions can be pruned. However, because of the properties of the multidimensional data representation (time complexity is exponential with the number of dimensions), the best recent results show that only queries up to 30 dimensions can be handled [18][19]. Above this limit, the search performance of indexing becomes comparable to sequential scan [24]. Other works propose to map the multidimensional data representation to another space with reduced dimensions. This allows dealing with a more manageable space at the expense of some loss of precision [2][3].

Clustering is a data partitioning approach. The aim is to form groups of similar objects, i.e. *clusters*. Most of the clustering schemes [16][25] construct a hierarchical structure of similar clusters or make use of an index structure to access rapidly some kind of cluster representative, such as the *centroid*, which typically is the mean vector of features for each cluster. In the searching process, this hierarchy or index is traversed by comparing the query object with the cluster representatives in order to find the clusters of interest. In both index and cluster-based strategies, the cost of computing the distances to determine similarity can be very high for large multimedia databases.

In this paper, we address the problem of CBR performance in large multimedia databases by exploiting parallelism, i.e. using a parallel database system. We assume a parallel (multiprocessor) system with the popular *shared-nothing* (SN) architecture [20], where each node has its own processor(s), local memory and own disk. Each node communicates with other nodes through messages over a fast interconnect. The main advantage of SN (e.g. over shared-disk) is excellent cost/performance and scalability. In this context, a major problem is data allocation on the different nodes in order to yield parallel content-based retrieval. The process of data allocation consists in data partitioning (producing a set of clusters) and data placement of the clusters onto the different SN nodes. The problem can be stated as follows: *given a database of n multimedia objects, find the optimal number of clusters and nodes to yield efficient processing of k NN queries.*

Assuming a clustering or any group forming partitioning process such as the k -means and based on a complexity analysis of CBR using clusters, we propose a data allocation scheme which computes an optimal number of clusters and nodes. Furthermore, the high dimensional representation of the multimedia objects is general enough to apply to other kinds of data. Thus we could apply our allocation scheme to any data that can be represented as multidimensional data objects, e.g. geographical maps, DNA sequences.

The rest of the paper is organized as follows. In Section 2, we present our allocation scheme, with its data partitioning and placement methods. In Section 3, we validate our proposal through simulation. In Section 4 we discuss related work and Section 5 concludes.

2 Data Allocation Scheme

Our data allocation scheme proceeds in two steps: (1) data partitioning which produces a set of clusters, and (2) data placement which places the clusters on the nodes

of the SN parallel system. This aims to reduce the time required to perform full search by partitioning the database among the nodes.

2.1 Data Partitioning

To overcome the problem introduced by the dimensionality of the data objects, which eventually deteriorates a search into a sequential scan or worst, one possibility is to rely on a clustering of the database. A priori, this process is computationally expensive but can be done off-line, in a preprocessing step before query execution. The goal is to minimize the number of objects distances to compute, by grouping them together based on their similarity. We do not suggest the use of any specific clustering process. This has been the focus of several works [13, 15, 5] which we can make use of to yield efficient CBR. Thus, we assume that a given database of size n , can be partitioned by using some clustering process, which produces a set C of clusters of similar data objects.

Based on this partitioning of the database, when executing a query, the set of clusters C can be pruned to a subset of candidate clusters C' containing the most similar objects. Our objective of this section is to propose an optimal number for $|C|$ based on the complexity analysis of the general searching process based on clustering.

For the case of searching via some data clustering, we can write the general complexity as:

$$O(f(C)) + O(g(C')) \quad (1)$$

In order to achieve optimal processing, this complexity should satisfy the following constraints:

- to minimize $O(g(C'))$ as a function of the number of candidate clusters, i.e., $|C'| \ll |C|$;
- to ensure that $O(f(C)) \leq O(g(C'))$;
- to ensure that $|C| \ll n$.

Let us consider the worst case of a search algorithm with:

- a linear selection of the candidate clusters;
- a sequential scan within each selected cluster;
- a full sort based on the merging of the results issued from the selected clusters.

Under these constraints then the general complexity (1) becomes:

$$O(C) + O\left(C' \frac{n}{C} + C' \frac{n}{C} \log_2 \left(C' \frac{n}{C}\right)\right) \quad (2)$$

Lemma 1. The search algorithm modeled by equation 2 has cost $O(\sqrt{n} \log_2 n)$ under the conditions:

$|C| = \sqrt{n} \log_2 n$; $|C'| \leq \log_2 n$ and classes of similar cardinalities.

Proof. First simplify by setting $C'=1$. Then propose $C = \sqrt{n}$ which gives a complexity in $O\left(\sqrt{n} + \frac{n}{\sqrt{n}} \log_2 \frac{n}{\sqrt{n}}\right) = O(\sqrt{n} \log_2 \sqrt{n}) = O(\sqrt{n} \log_2 n)$.

Second, let us propose $C = \sqrt{n} \log_2 n$ and the use of a multiplicative constant equal to $\frac{1}{2}$ which is the relation between $\mathbf{m}$, the set of features describing a data object and n , so that $m = \lambda \cdot n$. Then in the worst case, the complexity of equation 2 becomes:

$$O(\sqrt{n} \log_2 n) + O\left(\log_2 n \cdot \frac{n}{\sqrt{n} \log_2 n} + \log_2 n \cdot \frac{n}{\sqrt{n} \log_2 n} \cdot \log_2 \left(\log_2 n \cdot \frac{n}{\sqrt{n} \log_2 n}\right)\right),$$

i.e.,

$$O(\sqrt{n} \log_2 n) + O\left(\sqrt{n} + \sqrt{n} \log_2 \left(\frac{1}{2} \cdot \log_2 n\right)\right) \text{ which is certainly in } O(\sqrt{n} \log_2 n) \quad \blacksquare.$$

Our proposal for partitioning the database is then $|C| = \sqrt{n} \log_2 n$. Additionally, it can be derived some algorithmic variations, from optimal to suboptimal in $O(\sqrt{n} \log_2 n)$, under less restrictive conditions, $|C| \in [\sqrt{n}, \sqrt{n} \log_2 n]$ and $|C| \leq \log_2 n$, the optimal case can be obtained with a near multiplicative factor λ , since C' is small and independent of n .

2.2 Data Placement

Once the database is partitioned into clusters according to our proposal (above), we need to place them onto the nodes of the parallel system. Our solution for the placement of the clusters is to determine the number of nodes which yields the best (asymptotically) achievable performance. Thus we state the following theorem,

Theorem. Assume a shared nothing parallel system of at least $\log_2 n$ nodes, then the average complexity of any query is in $O(\sqrt{n})$.

Proof: The proof can be achieved as a result of Lemma 1 and of by a simple round robin placement of the clusters over the available nodes. That is, distributing $\sqrt{n} \cdot \log_2 n$ clusters over $\log_2 n$ nodes, each node will have $\sqrt{n}$ clusters each one containing in average $n/\sqrt{n} \cdot \log_2 n$ objects. With $|C'| \leq \log_2 n$ clusters selected, in the worst case the average number of selected classes on each node is only one. The local complexities, executed in parallel over all the participating nodes, are then:

$$\begin{aligned} \frac{n}{\sqrt{n} \log_2 n} \cdot \log_2 \frac{n}{\sqrt{n} \log_2 n} &= \frac{\sqrt{n}}{\log_2 n} \cdot \log_2 \frac{\sqrt{n}}{\log_2 n} = \frac{\sqrt{n}}{\log_2 n} \cdot \frac{1}{2} \cdot \log_2 \frac{n}{\log_2 n} \\ &= \frac{\sqrt{n}}{\log_2 n} \cdot \frac{1}{2} \log_2 n - \frac{\sqrt{n}}{\log_2 n} \cdot \frac{1}{2} \log_2 \log_2 n \end{aligned}$$

Since the last product factor is very small, we can ignore it. Thus we get the approximation.

$$\approx \frac{\sqrt{n}}{\log_2 n} \cdot \frac{1}{2} \cdot \log_2 n = \frac{1}{2} \cdot \sqrt{n} \in O(\sqrt{n})$$

Consolidation of results must be carried out by merging the local results, which are limited to the best k found data objects. In the worst case, each node returns as many results as there are objects in the treated class, that is $n/\sqrt{n} \cdot \log_2 n$, and all the nodes participate in the merging of the results, and the parallel complexity of the process is in :

$$\frac{n}{\sqrt{n} \log_2 n} \cdot \log_2 n \in O(\sqrt{n}).$$

The optimality is derived from the average size of the classes: increasing the number of nodes does not reduce the local complexities, they remain in $O(\sqrt{n})$. Assuming also a fast interconnection network which allows parallel transfers, then the merging process require only $\frac{n}{\sqrt{n} \log_2 n} \cdot \log_2 \log_2 n = \sqrt{n} \cdot \frac{\log_2 \log_2 n}{\log_2 n}$ operations, which becomes insignificant compared with local searching ■

The fact that the number of nodes obeys a logarithmic progression makes realistic its implementation, in particular because our proposal is conceived to deal with very large values of n . Also, as each node owns and works independently in a portion of the database, different queries can run in parallel.

3 Validation

To validate the efficiency of our proposal, we developed an experimental platform. It is written in Java 1.5 running on a shared-nothing cluster of Intel Xeon IA32 2.4GHz nodes with 2GB of main memory each one. In this section, we describe the datasets used, the k nearest neighbor searching process and the performance results obtained.

3.1 Experiments with Non Uniform Data Sets

Real data tends to form natural clusters of different sizes, which usually can be approximated by a Gaussian distribution. To validate our proposal we have crafted a synthetic cluster generator. The data generated simulate hyper-spherical clusters of feature vectors with uncorrelated features. Perhaps this has been a largely used approach as in [11][25] but we go far by varying some conditions at the interior of each cluster with the aim of providing a close to real workload.

As we propose an interval of validity for the number of clusters, we have generated several datasets. Each dataset is characterized by $\langle n, d, |C| \rangle$, which are the number of points, the size of the multidimensional space and the number of clusters respectively. Non uniform datasets were generated with the following range of values: $n \in \{10^6, 10^7\}$, $d \in \{10, 20, 50, 100, 200, 256, 500, 1000\}$, $C \in \{\sqrt{n}, \sqrt{n} \log_2 n\}$. Each cluster is characterized by $\langle r, \delta, \rho \rangle$. The radius r defines the space a cluster occupies; δ is the density, i.e. the number of points per unit of volume. The population ρ determines the number of d -points in the cluster. They have the following range of values: $r \in \{1, [1, 3]\}$, the first value indicates uniform radius and whereas the second one means radius is non uniform and is generated randomly in the interval using these

values as coefficients to the analytical cluster size. Similarly, the population is generated with $\rho \in \left\{1, \left[\frac{1}{3}, 3\right]\right\}$, so that clusters does not have the same population. For fairness of query processing, all generated datasets are normalized in $[0,1]^d$ by dividing all the vectors values by the highest dimension component value, irrespective of the dimension. After definition of all these parameters, to generate the clusters, first, $|C|$ d -dimensional centers are randomly generated with some random radius, providing the positioning of each cluster in the multidimensional space. Second, each cluster c is populated with ρ d -dimensional Gaussian points, this stands for a more near to real workload.

3.2 kNN Searching Process

We now describe our simple but effective algorithm to enable fast processing of k NN queries in the parallel system. Keep in mind that the process follows the parameters given in section 2.2 and it is disk based.

Step 1. Data Distribution

We did not try yet to develop a heuristic algorithm for optimizing the placement of the clusters. But the placement algorithm can take into consideration proximities between classes to distribute the nearest ones over different nodes, but as we show this is a questionable tradeoff of investing in a minimal gain at the cost of some (maybe) expensive process (cf 3.4). Here, they are distributed on the $\log_2 n$ machines in a round-robin way. Therefore, all the nodes have almost the same number of clusters.

Step 2. Cluster Selection

Query points are generated randomly under the same assumptions than the generated datasets. To find the set of k NN objects for a query $\mathbf{q}$, a sufficient number of classes must be selected in order to ensure to return the best k d -points. Firstly, a sequential disk-based search is executed to compare $\mathbf{q}$ with all the centroids, using the common Euclidean distance, and using the radiuses to find the nearest clusters¹. The first $\log_2 n$ clusters are considered. For each selected cluster, we know the node it has been placed during step 1.

With this simply heuristic, we aim at retrieving all the k NN which is equivalent to a full search. Indeed, in several research works on k NN query processing analysis [15, 16] propose a threshold which guarantees the retrieval of all or a high percentage of the NN. Here, taking the first ranked $\log_2 n$ clusters is a threshold large enough to achieve full precision.

Step 3. Sequential Scans of the Selected Clusters

Once known the node for each cluster of interest, a parallel search is executed on them to retrieve up to k objects. On each node, the scan of the locally selected class takes place in the form of a mere sequential search and the use of a bounded priority

¹ Let us note that even though the centroids of the clusters could fit into main memory, if we consider that the data is actually stored in a database and our aim is to validate our proposal for worst case conditions, then the process is disk-based.

queue to keep the best k responses. Therefore, each node returns a set of k sorted elements to the master node.

Step 4. Merging

The master node merges the various results, prunes the list of the first k answers, and returns it to the user or the application.

3.3 Performance Evaluation

In order to obtain experimental evidence of how well our data allocation method improves the processing of nearest neighbor queries, we processed 1000 k NN queries from the same data space for the above mentioned datasets. We measured the average response time, the number of distance comparisons and disk IO's. The results presented here are for databases of 10^7 data objects. The sizes of k for the queries are 50, 100 and 150.

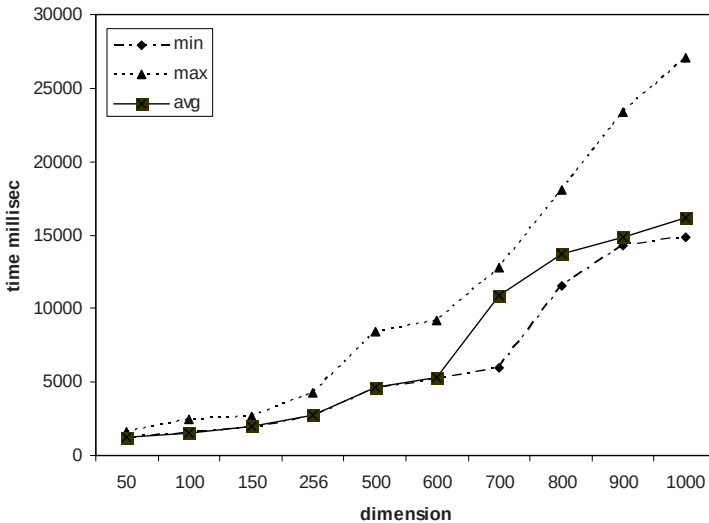


Fig. 1. Execution times for 1000 50 kNN queries for $n=107$ non uniform databases of different data dimensionalities

Figures 1 to 3 show the behavior of the query algorithm for non uniform datasets. Even when the trend to increase processing time with the dimensionality is exhibited, in none of our experiments, it is exponential. It has also a tendency to increase less rapidly as expected from our proposal.

The high difference between minimal values and maximal values is a java code implementation issue and it is due to the activation of the garbage collector of the JVM. We have removed some of the queries involved in the peak values and repeated the experiments for them and obtained a considerable decrease in time. This corroborates that when the JVM memory usage becomes near the assigned memory, garbage collection is activated, thus hurting performance.

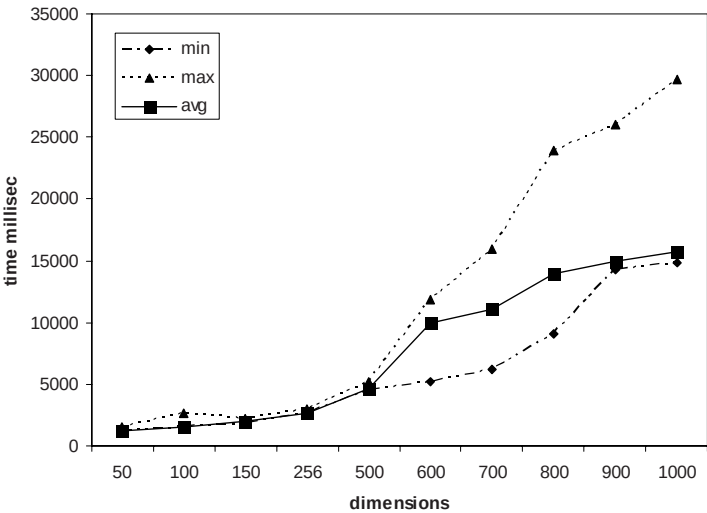


Fig. 2. Execution times for 1000 100 kNN queries for n=107 non uniform databases of different data dimensionalities

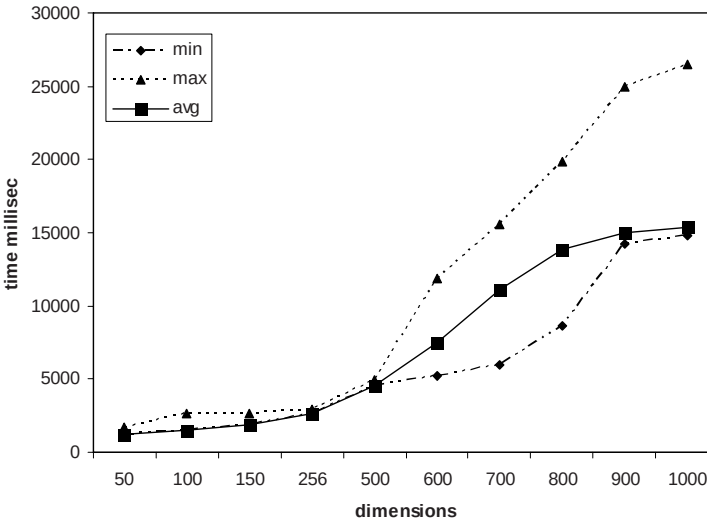


Fig. 3. Execution times for 1000 150 kNN queries for n=107 non uniform databases of different data dimensionalities

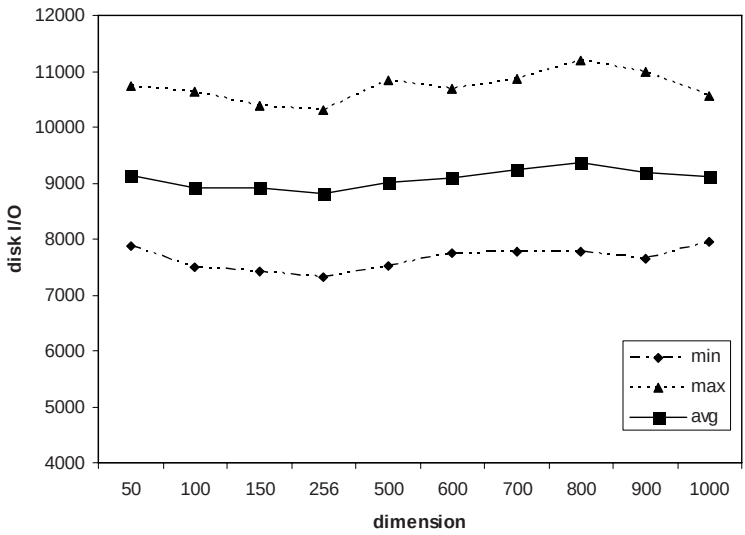


Fig. 4. Disk I/O count for the 50 kNN query processing over databases of different dimensionalities

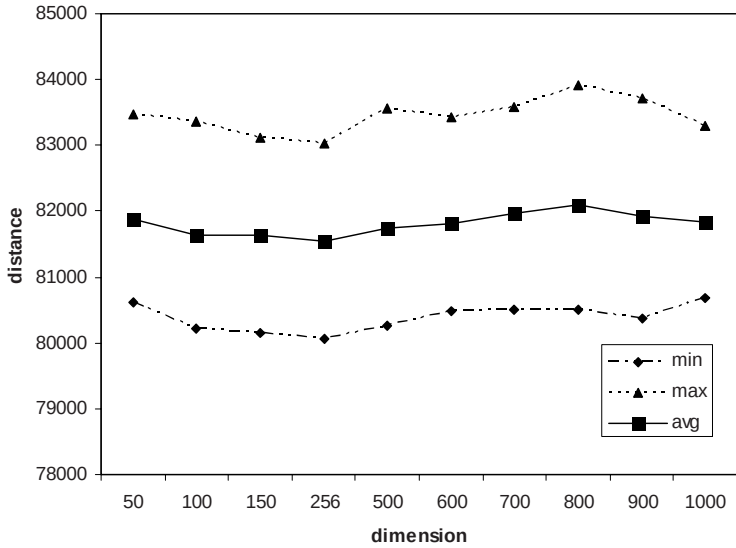


Fig. 5. Distance computations count for the 50 kNN query processing over databases of different dimensionalities

Figures 4 and 5 have almost steady curves. This is due to the query processing algorithm. In all cases, the parallel search is performed in the top $\log_2 n$ clusters, and each processor returns at most k results. Almost similar values are obtained for $k = 100$ and 150 , thus making unnecessary to present the curves. Here the cluster size is the influence factor. This allows us to conclude that relatively small sizes for the clusters help to reduce searching time. The key is then the cluster pruning process. Thus, it is not required to be concerned on efficiency issues of the search in the interior of the selected clusters. Notice that we use non uniform populations for the clusters as would happen with real data and that having to process queries for large values of k is also impractical. Even in the case when the CBR process is used for browsing; i.e. when the user is not sure of her needs and needs a variety of results, we consider better to wait and let the user lead the retrieval process as when relevance feedback is used.

3.4 Effect of the Size of k

For most related works the size of k is a factor of influence in the number of disk I/O and distances computed, directly affecting the response time. Under our scheme this is not a main concern. The heuristic allows selecting the most relevant clusters which are processed locally and only in the final merging result can have an effect; which is however small as the results are merged from sorted lists and finally pruned to k . For example, for 256 dimensional data sets from figure 1 the process time is 2709, 2697 and 2698 milliseconds for $k=50, 100$ and 150 respectively.

3.5 Effect of Data Placement

To maximize parallelism, the clusters selected C^s by the searching algorithm must be equally distributed on the nodes and no more than $\left\lceil \frac{C^s}{nodes} \right\rceil$ clusters should be retrieved from each one. That is, all the nodes are activated and contribute to speed up the processing of a query and as it is retrieved an equal number of clusters from each one, the workload is balanced. Theoretically, this should happen in an ideal situation, but given that one can not anticipate all the possible queries so that all the nearest clusters are distributed in different nodes, it is a NP-complete problem and it has been proved that the optimal can be achieved in a very few cases [1].

To improve parallelism, candidate clusters must be equally placed on the nodes and no more than $\left\lceil \frac{C^s}{nodes} \right\rceil$ clusters must be retrieved from each one. This has the effect of activating all the nodes to rapidly process the query (intra-query parallelism) and also to balance the processing load.

The proposed query processing algorithm prunes the clusters and selects the top $\log_2 n$ ranked by their similarity with respect to the query object, then just the involved nodes contribute to the final answer. The number of activated nodes in parallel are showed in figure 6. Notice that in this work, the proposal is based on a *worst case assumption*, i.e., processes are disk based and the placement algorithm is round robin, thus there is room to improve the performance, but what is important to note is the achieved performance behavior.

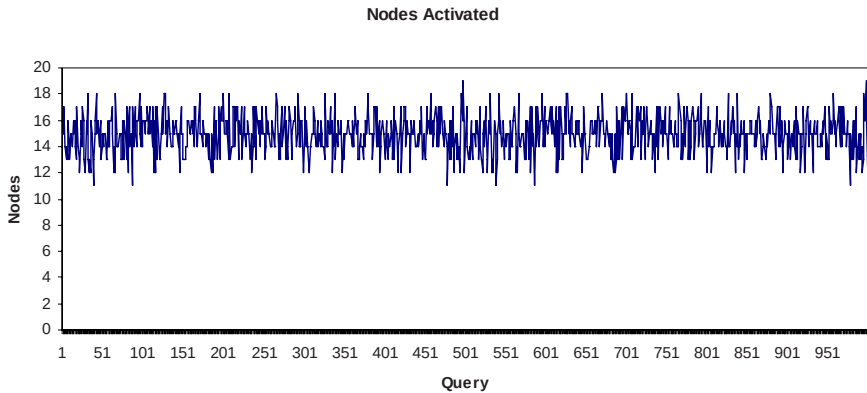


Fig. 6. Nodes activated simultaneously to process a query in 256 dimensions

Under Round Robin, the node more charged has to process four clusters, it can handle that in average in 300 milliseconds for 500- d data.

4 Related Work

The idea of indexing methods is the early elimination of as much as possible regions of the data space that are not of interest to the query, thus yielding a minimal subset to search in. However they are good in average for 30 dimensional data. Above 30, the problem known as curse of dimensionality arises and these methods are outperformed by simple linear search [24]. Some noticeable recent work is iDistance [18][19] which reports good performance up to 32 dimensions. Among the clustering methods, Clin-dex [16] and ClusterTree [25] aim to fight the so called curse of dimensionality which affects indexing methods. Even though they provide acceptable performance, there is no report of how these structures can be implemented or behave in parallel settings. In [14] it is presented an architecture of one processor-multiple disks to process range queries using an interesting concept called proximity index allocation which measures the similarity between nodes to later place them on the disks, avoiding to put similar nodes together, this allows a near optimal allocation process. A shared nothing based implementation is presented by [23], however they are based on the R-Tree which is used to spatial data. In [6] Berchthold et al proposed an allocation method for the X-Tree using a graph coloring algorithm. It has been reported that the Pyramid tree outperforms the X-Tree by a factor of 800 in response time in centralized settings [25]. In [21] it is presented a four disk architecture concerned mainly with the placement within each disk, i.e. a very low level placement scheme of wavelet coefficients. This architecture is used for browsing image thumbnails by exploiting parallelism (they use parallelism to retrieve the coefficients) but the final selected images must be reconstructed in the clients' side from the wavelet coefficients (kind of feature vector). The Parallel M-Tree [26] and one extension [4] address the problem of distributing the nodes of the M-Tree in the parallel system by computing the distance of each newly object placed in a node with all the objects already available, it is achieved by

performing a range query with a radius equal to the distance from the object to its parent. In [9][10], the authors propose an hybrid architecture of redundant multidisc and shared nothing, to improve disk IO based in the Parallel R-Tree. However, they provide results for 100, 000 data of up to 80 dimensions which are outperformed by our proposal with much less resources. Recently, Liu et al [17] propose a clustering based parallel spill tree altogether with a search method, in their study, they begin with an initial 1.5 billion images which after “cleaning” remains in around 200 million, a large enough image database but the dimensionality of the dataset is 100 and their proposal needs 200 machines. In the contrary, we use 1000 descriptors and $\log_2 n$ nodes, which in order to process the whole 200 millions images database would require only 27 nodes.

5 Conclusions

In this paper we proposed a data allocation scheme for efficient k NN query processing on multimedia databases in a shared-nothing architecture. Our proposal can be summarized to the following contribution: an upper and a lower bound on the number of fragments or clusters that can be obtained from the database and the number of nodes required to maximize resources utilization and to achieve optimal parallel k NN searching. The number of clusters is based on the complexity analysis of the general search problem for CBR and the number of nodes in the parallel architecture is proposed on the same principle but taking into consideration that it must be a feasible to implement architecture. We validated our method for different non uniform datasets under worst case considerations, any improvement such as a better placement scheme and putting into main memory the centroids must signify a performance improvement. Here we showed that our proposal stands for the derived algorithmic complexities.

Acknowledgements

The first author thanks the support of the CONACYT and IPN from Mexico, as well as INRIA France.

References

1. Abdel-Ghaffar, K.A.S., El Abbadi, A.: Optimal Allocation of Two-Dimensional Data. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 409–418. Springer, Heidelberg (1996)
2. Aggarwal, C.C.: On the Effects of Dimensionality Reduction on High Dimensional Similarity Search. In: ACM PODS 2001: Symposium on Principles of Database Systems Conference, pp. 256–266 (2001)
3. Aggarwal, C.C.: An efficient subspace sampling framework for high-dimensional data reduction, selectivity estimation, and nearest-neighbor search. IEEE Transactions on Knowledge and Data Engineering 16(10), 1247–1262 (2004)

4. Alpkocak, A., Danisman, T., Ulker, T.: A Parallel Similarity Search in High Dimensional Metric Space Using M-Tree. In: Grigoros, D., Nicolau, A., Tournel, B., Folliot, B. (eds.) IWCC 2001. LNCS, vol. 2326, pp. 166–171. Springer, Heidelberg (2002)
5. Attila Gürsoy, E.E.: Data Decomposition for Parallel K-means Clustering. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2004. LNCS, vol. 3019, pp. 241–248. Springer, Heidelberg (2004)
6. Berchtold, S., Böhm, C., Braunmüller, B., Keim, D.A., Kriegel, H.: Fast parallel similarity search in multimedia databases. In: SIGMOD Rec., vol. 26(2), pp. 1–12 (1997)
7. Berrani, S.-A., Amsaleg, L., Gros, P.: Approximate Searches: k-Neighbors + Precision. In: CIKM 2003: Proceedings of the 12th ACM International Conference on Information and Knowledge, pp. 24–31 (2003)
8. Böhm, C., Berchtold, S., Keim, D.A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Comput. Surv. 33(3), 322–373 (2001)
9. Bok, K.S., Seo, D.M., Song, S.I., Kim, M.H., Yoo, J.S.: An Index Structure for Parallel Processing of Multidimensional Data. In: Fan, W., Wu, Z., Yang, J. (eds.) WAIM 2005. LNCS, vol. 3739, pp. 589–600. Springer, Heidelberg (2005)
10. Bok, K.S., Song, S.I., Yoo, J.S.: Efficient k-Nearest Neighbor Searches for Parallel Multidimensional Index Structures. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 870–879. Springer, Heidelberg (2006)
11. Chavez, E., Navarro, G.: Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces. Information Processing Letters 85(1)(16), 39–46 (2003)
12. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P.: Query by Image and Video Content: The QBIC System. IEEE Computer 28(9), 23–32 (1995)
13. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
14. Kamel, I., Faloutsos, C.: Parallel R-trees. In: SIGMOD 1992: Proceedings of the ACM international Conference on Management of Data, pp. 195–204 (1992)
15. Kanungo, T., Mount, D.M., Netanyahu, N., Piatko, C., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Analysis and Machine Intelligence 24, 881–892 (2002)
16. Li, C., Chang, E., Garcia-Molina, H., Wiederhold, G.: Clustering for approximate similarity search in high-dimensional spaces. IEEE Transactions on Knowledge and Data Engineering 14(4), 792–808 (2002)
17. Liu, T., Rosenberg, C.R., Rowley, H.A.: Clustering Billions of Images with Large Scale Nearest Neighbor Search. In: 8th IEEE Workshop on Applications of Computer Vision (WACV 2007), p. 28 (2007)
18. Ooi, B.C., Tan, K.L., Yu, C., Zhang, R.: Indexing the Distance: An Efficient Method to KNN Processing. In: VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 421–430 (2001)
19. Ooi, B.C., Tan, K.L., Yu, C., Zhang, R.: iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. Journal of the ACM Transactions on Database Systems 30(2), 364–397 (2005)
20. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 2nd edn. Prentice-Hall, Englewood Cliffs (1999)
21. Prabhakar, S., Agrawal, D., El Abbadi, A., Singh, A., Smith, T.: Browsing and placement of multi-resolution images on parallel disks. Multimedia Systems 8(6), 459–469 (2003)

22. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest Neighbor Queries. In: SIGMOD 1995: Proceedings of the International Conference on Management of Data, San Jose, California, May 22-25, pp. 71–79 (1995)
23. Schnitzer, B., Leutenegger, S.T.: Master-Client R-Trees: A New Parallel R-Tree Architecture. In: SSDBM 1999: Proceedings of the 11th International Conference on Scientific and Statistical Database Management (1999)
24. Weber, R., Schek, H.J., Blott, S.: A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In: VLDB 1998: Proceedings of the 24th International Conference Very Large Data Bases, pp. 194–205 (1998)
25. Yu, D., Zhang, A.: ClusterTree: Integration of Cluster Representation and Nearest Neighbor Search for Large Datasets with High Dimensionality. *IEEE Transactions on Knowledge and Data Engineering* 15(5), 1316–1337 (2003)
26. Zezula, P., Savino, P., Rabitti, F., Amato, G., Ciaccia, P.: Processing M-trees with parallel resources. In: Research Issues In Data Engineering. Eighth International Workshop on Continuous-Media Databases and Applications, pp. 147–154 (1998)

OLAP for Trajectories

Oliver Baltzer¹, Frank Dehne²,
Susanne Hambrusch³, and Andrew Rau-Chaplin¹

¹ Dalhousie University, Halifax, Canada
obaltzer@cs.dal.ca, arc@cs.dal.ca
<http://www.cs.dal.ca/~arc>

² Carleton University, Ottawa, Canada
frank@dehne.net
<http://www.dehne.net>

³ Purdue University, West Lafayette, IN, USA
seh@cs.purdue.edu

<http://www.cs.purdue.edu/people/faculty/seh/>

Abstract. In this paper, we present an OLAP framework for trajectories of moving objects. We introduce a new operator GROUP_TRAJECTORIES for group-by operations on trajectories and present three implementation alternatives for computing groups of trajectories for group-by aggregation: *group by overlap*, *group by intersection*, and *group by overlap and intersection*. We also present an interactive OLAP environment for resolution drill-down/roll-up on sets of trajectories and parameter browsing. Using generated and real life moving data sets, we evaluate the performance of our GROUP_TRAJECTORIES operator. An implementation of our new interactive OLAP environment for trajectories can be accessed at <http://OLAP-T.cgmlab.org>.

1 Introduction

Global positioning (GPS) and RFID systems are creating vast amounts of spatio-temporal data for *moving objects*. Consider N moving objects on a 2D spatial grid. Each object is identified by a unique *tag* number (similar to EPC in RFID). Object movements are recorded through a set of readings $((x, y), i, t)$ indicating that object (tag) i was detected at time t within the grid cell located at (x, y) . The N moving objects are represented by a relational table *objects* with N records. Each record contains values *tag*, *name*, *size*, *color*, etc. describing one object according to a star schema. Among them is a value *trajectory* representing the movement of the respective object as a sequence $[(x_1, y_1, t_1), (x_2, y_2, t_2), \dots, (x_m, y_m, t_m)]$ of positions at time $t = t_1, t_2, \dots, t_m$. In order to efficiently *analyze* large scale data sets representing moving objects, it is important to have available the well established set of tools for OLAP analysis. In order to apply OLAP tools towards moving object datasets, it is necessary to aggregate with respect to *trajectory* as a *feature* dimension as well as a *measure* dimension.

We illustrate this with the example shown in Figure 1. Consider the trajectories shown in Figure 1a. We observe a number of individual objects that move

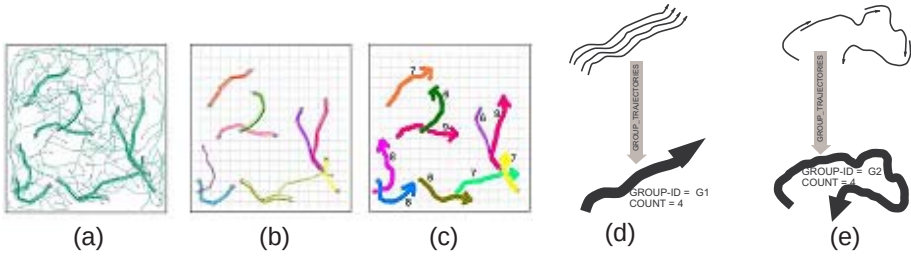


Fig. 1. *OLAP For Trajectories* Example. (a) Input data. (b) Groups with minimum support. (c) Aggregate results reported (aggregate trajectories and counts). (d) Illustration of operator `GROUP_TRAJECTORIES:Group by Intersection`. (e) Illustration of operator `GROUP_TRAJECTORIES:Group by Overlap`.

on random paths plus 10 *groups* of objects that move together on similar paths. Each group consists of more than five objects moving on similar paths which, taken together, appear to the human eye as “bold” paths.

Consider the following SQL query where *trajectory* is both, a *feature* dimension as well as a *measure* dimension:

```
SELECT AGGREGATE(trajectory) AS trajectory
      COUNT(trajectory) as count
FROM objects
GROUP BY GROUP_TRAJECTORIES(trajectory,
                             resolution)
HAVING COUNT(*) >= 5
```

For this example, the aim of the `GROUP BY` operation with respect to *feature* dimension *trajectory* is to group similar trajectories and eliminate groups with less than minimum support (less than 5 similar trajectories). The resulting set of groups is shown in Figure 1b. Once the groups of trajectories have been determined, we report for each group an *aggregate trajectory* representing the trajectories in the group. In this example, the aggregate trajectory is the average trajectory computed by calculating for each time t_i the average of the locations (x_i, y_i) of the trajectories in the group. The result is shown in Figure 1c, where each group is represented by the aggregate trajectory and size of the group (count).

The goal of *OLAP analysis for trajectories* is to answer aggregate queries with respect to the spatial movements of a set of objects represented in a relational table *objects*. The main problem arising is how to aggregate with respect to feature dimension *trajectory*. It is very unlikely that any two trajectories are exactly the same. Hence, standard aggregation of records with equivalent *trajectory* values is not very useful in most cases. We propose to partition the given trajectories into disjoint *groups* of trajectories using a new operator which we term `GROUP_TRAJECTORIES`. This operator returns for each trajectory a *group identifier*, and then OLAP can proceed with standard aggregation according to the group identifiers instead of the trajectories themselves.

The main problem addressed in this paper is how to define and compute the operator `GROUP_TRAJECTORIES` such that the resulting groups allow for a meaningful analysis of object movements via OLAP. We propose *three different versions* of the operator `GROUP_TRAJECTORIES` which compute groups of trajectories that are appropriate for OLAP analysis of trajectories for different circumstances and applications: *Group by Overlap*, *Group by Intersection* and *Group by Overlap and Intersection*.

Section 3 will show in detail how these three different versions of our `GROUP_TRAJECTORIES` operator are defined and computed. Our *Group by Intersection* method aggregates subsets of trajectories that correspond to similar or synchronous movements; see Figure 1d. Our *Group by Overlap* method aggregates subsets of trajectories that correspond to sequences of movements with sufficient overlap between subsequent trajectories; see in Figure 1e. The *Group by Overlap and Intersection* method aggregates subsets of trajectories that correspond to a combination of sequences of movements and similar or synchronous movements.

In Section 4, we present an *interactive OLAP environment* for the analysis of trajectories that allows resolution drill-down and roll-up as well as parameter browsing. An experimental evaluation is outlined in Section 5. An implementation of our new interactive OLAP environment for trajectories can be accessed at <http://OLAP-T.cgmlab.org>.

2 Related Work

There is a wealth of literature on spatiotemporal data analysis and aggregation. See e.g. [9] for a survey. This work studies aggregation by specific temporal dimensions such as "by day" or "by year", or by strict topological association such as "by location square" or "within 10 km of" (e.g. [11]). In our case, we wish to aggregate entire trajectories. For the detection of relationships among trajectories in a moving object database we found in the literature five groups of approaches: variations of frequent pattern or association rule mining (e.g. [4,5,6,16]), clustering techniques (e.g. [8,12]), Computational Geometry techniques (e.g. [7]), neural network based techniques (e.g. [15]), and edit distance, warping techniques and longest common subsequence (LCSS) extraction (e.g. [13,14,17,18]). A comparison of our work with these approaches is omitted due to page restrictions. It can be found in the extended version of this paper [2].

3 Computing Groups of Trajectories

In this section we present three different implementations of the operator `GROUP_TRAJECTORIES` which compute groups of trajectories that are appropriate for OLAP analysis of trajectories for different circumstances and applications: *Group by Overlap*, *Group by Intersection*, and *Group by Overlap and Intersection*. We first apply a time and space resolution mapping of our initial set $\mathcal{T}$ of trajectories. This allows for the resolution drill-down and roll-up within our interactive OLAP framework for trajectories to be discussed in Section 4. Next, we compute frequent

itemsets for the mapped set of trajectories and then apply a reverse mapping step. Here, we determine for each frequent itemset f , the corresponding *original* group c of trajectories and create a set $\mathcal{C}$ of resulting (f, c) pairs. A more detailed presentation of our method is contained in the extended version of this paper [2].

The most important part of our method is the group merging phase. In this paper, we present three different methods: (a) Group by Overlap (Sections 3.1), (b) Group by Intersection (3.2), and (c) Group by Overlap and Intersection (3.3).

3.1 Group by Overlap

Our *Group By Overlap* method introduces a tunable parameter *overlap ratio threshold* ORT which controls the strength of the grouping process. The interactive OLAP framework for trajectories discussed in Section 4 will allow for an interactive tuning of this parameter.

Our *Group By Overlap* method is based on an *overlap graph* Γ , where each vertex corresponds to a trajectory. For each frequent item set f and corresponding set c of trajectories, we consider all pairs of trajectories $t_i, t_j \in c$ and add for each pair an edge (t_i, t_j) with label *overlap ratio* $OS = \frac{2 \cdot |f|}{|t_i| + |t_j|}$. The *overlap ratio* measures the size of the overlap relative to the sizes of the trajectories. We then remove all edges where the *overlap ratio* OS is smaller than the chosen *overlap ratio threshold* ORT and compute the connected components of the remaining graph. These components correspond to the groups of trajectories that are reported. A more detailed presentation is contained in the extended version of this paper [2].

The nature of the obtained groups of trajectories is determined by two factors. (1) The *overlap ratio threshold* ORT determines how much two neighboring trajectories within a group have to overlap. (2) The graph connected component construction allows for an “adding up” of trajectories corresponding to a “relay” type of movement. Depending on the chosen *overlap ratio threshold* ORT , the “relay” parties will have to move in unison for more or less of their own individual movements.

3.2 Group by Intersection

Our *Group By Intersection* method introduces a tunable parameter *intersection ratio threshold* IRT which controls the strength of the grouping process. The interactive OLAP framework for trajectories discussed in Section 4 will allow for an interactive tuning of this parameter.

Our *Group By Intersection* method first creates an initial set $\mathcal{G}$ of groups of trajectories, where each group c corresponds to a frequent itemset f determined in the reverse matching in Section 3. Each group c is assigned a *group strength* $GS(c)$ which is initially set to the size of the respective frequent itemset. The remainder of our method merges groups in $\mathcal{G}$ by iterating the following loop. We compute for each pair $g_i, g_j \in \mathcal{G}$ a value *intersection ratio* $AS(g_i \cup g_j) = \min \left(\frac{|g_i \cap g_j|}{|g_i|}, \frac{|g_i \cap g_j|}{|g_j|} \right)$ which represents the number of trajectories that occur in

both g_i and g_j , relative to the sizes of g_i and g_j . We will consider as candidates for merging all pairs g_i, g_j whose intersection ratio is larger than our input parameter *intersection ratio threshold* IRT and compute for each such pair a value *merge strength* $MS(g_i \cup g_j) = \frac{GS(g_i) + GS(g_j)}{2}$ which is the average of their *group strength* values. All candidate pairs are ranked by their *merge strength* and we will merge the pair g_i^*, g_j^* with maximum merge strength, or one of the maximal pairs if there are multiple. The *group strength* $GS(g_i^* \cup g_j^*)$ of the new merged group will be the *merge strength* $MS(g_i^* \cup g_j^*)$. This process is repeated until there are no more pairs of groups with non zero *merge strength*, that is, until there are no more pairs of groups with *intersection ratio* larger than the *intersection ratio threshold* IRT . A more detailed presentation of our method is contained in the extended version of this paper [2].

Our *Group by Intersection* method aggregates subsets of trajectories that correspond to “marching band” style parallel movements. The nature of the obtained groups of trajectories is determined by two factors. (1) The *intersection ratio threshold* IRT determines how many shared trajectories between two groups are “sufficient” for them to be merged. (2) The merging process which is similar in nature to a minimum spanning tree calculation. We merge first the largest groups with sufficient shared trajectories and then work our way down to the smaller groups. Unlike the *Group by Overlap* method which combines sequences of movements, the *Group by Intersection* method combines parallel of movements.

3.3 Group by Intersection and Overlap

The goal of our *Group by Intersection and Overlap* method is to group both, sequences of movements and parallel movements. It is a combination of our methods in Sections 3.1 and 3.2. We create the same set $\mathcal{G}'$ of groups of trajectories as in Section 3.2 and the same overlap graph Γ as in Section 3.1. Then we add to Γ a clique for each $g \in \mathcal{G}'$ (i.e. edges between all pairs of trajectories $t_1, t_2 \in g$) and compute the connected components of the modified graph Γ . Each connected component corresponds to a group of trajectories.

The resulting groups are sequences of overlapping trajectories as in our *Group by Overlap* method to which we add parallel trajectories as in our *Group by Intersection* method. The aggregation is guided by two parameters, the *intersection strength threshold* IRT and the *overlap ratio threshold* ORT , which control the width and length, respectively, of the generated groups.

4 Interactive OLAP for Trajectories

The algorithms for the three different versions of operator GROUP_TRAJEC-TORIES presented in Section 3 are guided by the following parameters: space resolution, time resolution, minimum support, intersection ratio threshold and overlap ratio threshold. This allows to analyze groups of trajectories for various levels of resolution or connectedness, and provides another opportunity for

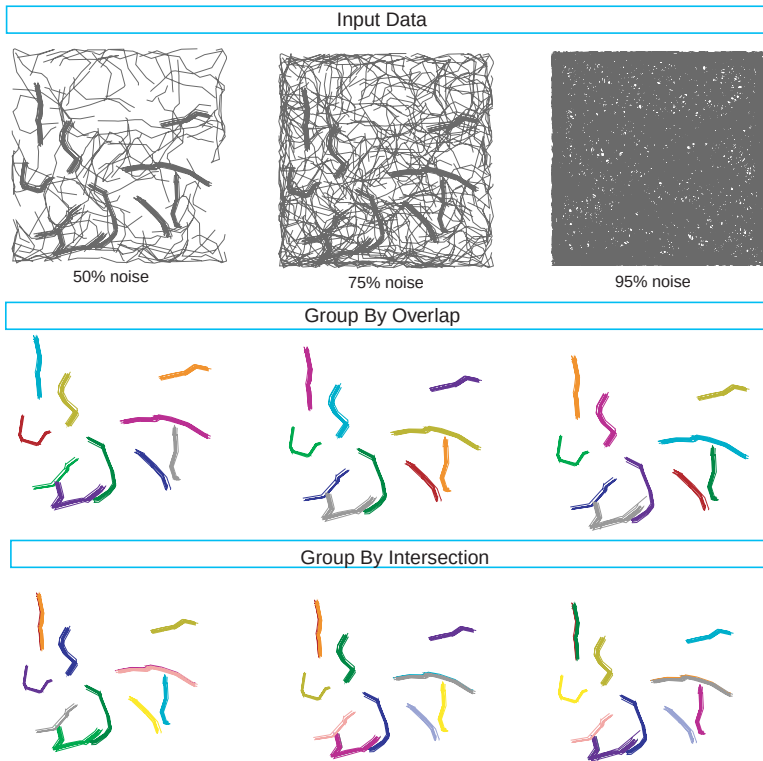


Fig. 2. Test of robustness against noise. Top row: input data consisting of 10 groups with 10 similar trajectories each and three levels of noise: 50%, 75% and 95%. Center row: Groups computed by GROUP_TRAJECTORIES: *Group By Overlap* ($ORT = 0.5$, $min\_support = 4$). Bottom row: Groups computed by GROUP_TRAJECTORIES: *Group By Intersection* ($IRT = 0.5$, $min\_support = 4$). Groups are identified by color ($group\_identifier = color$).

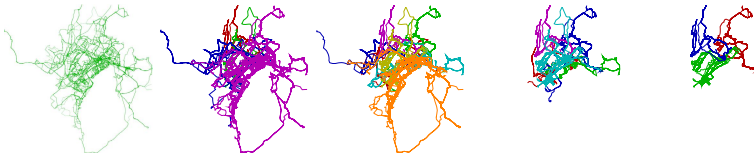


Fig. 3. School Buses Dataset and Groups reported (identified by color) using *Group by Overlap* and $ORT = 0.4, 0.5, 0.6, 0.7$, respectively ($min\_support = 5$, $min\_length = 30$)

OLAP analysis of trajectories. For example, for a high level analysis of GPS data for the movement of a fleet of ships, time granularity “day” may be sufficient. However, a drill-down to viewing the paths taken by a group of ships when

entering a port may require a time granularity “minute”. As an example for browsing a parameter like *overlap ratio threshold*, consider a set of trajectories representing movements of people who pass on a disease virus. The aggregate, using our *Group by Overlap* method, could be used to analyze the total movement of the virus. In this example, our parameter *overlap ratio threshold* would represent the amount of interaction between individuals required to pass on the virus. Changing the threshold value allows to evaluate how far the virus will spread based on different assumption about its transmission.

We have built a prototype *interactive environment for the analysis of trajectories* that allows resolution drill-down and roll-up as well as parameter browsing. It can be accessed at <http://OLAP-T.cgmlab.org>.

5 Experimental Evaluation

Our *Group by Overlap* and *Group by Intersection* methods have a surprising resilience against background noise. On the example shown in Figures 2, as well as many other examples that we tested, they have no trouble reporting the correct result for noise levels of 50%, 75% and even as high as 95%. At a noise level of 95%, the human eye can no longer visually detect the original groups of parallel paths but our methods have no problem reporting the correct result.

For the evaluation of our methods on real world data, we have chosen the school buses dataset that can be freely obtained from [1]. The dataset contains 145 trajectories of buses that are moving in and around an urban area. Due to page restrictions, we can not show the dataset here. It can be viewed by going to <http://OLAP-T.cgmlab.org> and selecting the dataset “buses”.

Frequent itemsets mining without aggregation, as e.g. in [4,10,3] (plus a minimum length cutoff as used in our methods), would result in 76 groups being identified. This large number of groups reported by frequent itemsets mining based methods is often a disadvantage because it does not lead to significant aggregation in an OLAP setting. Figure 3 shows the results obtained with our *Group by Overlap* method for *ORT* values 0.4, 0.5, 0.6, and 0.7. We observe that the parameter *ORT* in our *Group by Overlap* method allows for a much finer control over the grouping of trajectories reported and that the *Group by Overlap* method reports a considerably smaller number of groups.

A more detailed presentation of experimental results for our method is contained in the extended version of this paper [2].

References

1. R-tree Portal (Last accessed, November 16, 2007), <http://www.rtreeportal.org/>
2. Baltzer, O., Dehne, F., Hambrusch, S., Rau-Chaplin, A.: Olap for trajectories. Technical Report TR-08-11, School of Computer Science, Carleton University, <http://www.scs.carleton.ca>
3. Cao, H., Mamoulis, N., Cheung, D.W.: Mining frequent spatio-temporal sequential patterns. *icdm*, 82–89 (2005)

4. Gidófalvi, G., Pedersen, T.B.: Mining Long, Sharable Patterns in Trajectories of Moving Objects. In: STDBM 2006: Proceedings of the 3rd Workshop on Spatio-Temporal Database Management (2006)
5. Hwang, S.Y., Liu, Y.H., Chiu, J.K., Lim, E.P.: Mining mobile group patterns: A trajectory-based approach. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 713–718. Springer, Heidelberg (2005)
6. Kim, D., Kang, H., Hong, D., Yun, J., Han, K.: STMPE: An Efficient Movement Pattern Extraction Algorithm for Spatio-temporal Data Mining. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3981, pp. 259–269. Springer, Heidelberg (2006)
7. Laube, P., van Kreveld, M., Imfeld, S.: Finding REMO—detecting relative motion patterns in geospatial lifelines. In: Developments in Spatial Data Handling: Proceedings of the 11th International Symposium on Spatial Data Handling, pp. 201–214 (2004)
8. Li, Y., Han, J., Yang, J.: Clustering moving objects. In: KDD 2004: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 617–622. ACM, New York (2004)
9. López, I.F.V., Snodgrass, R.T., Moon, B.: Spatiotemporal Aggregate Computation: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 17(2), 271–286 (2005)
10. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., Cheung, D.W.: Mining, indexing, and querying historical spatiotemporal data. In: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 236–245 (2004)
11. Marchand, P., Brisebois, A., Bédard, Y., Edwards, G.: Implementation and evaluation of a hypercube-based method for spatiotemporal exploration and analysis. *ISPRS Journal of Photogrammetry and Remote Sensing* 59(1-2), 6–20 (2004)
12. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.* 27(3), 267–289 (2006)
13. Sclaroff, S., Kollios, G., Betke, M.: Motion mining: discovering spatio-temporal patterns in databases of human motion. In: Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2001)
14. Shim, C.B., Chang, J.W.: A new similar trajectory retrieval scheme using k-warping distance algorithm for moving objects. In: Dong, G., Tang, C.-j., Wang, W. (eds.) WAIM 2003. LNCS, vol. 2762, pp. 433–444. Springer, Heidelberg (2003)
15. Sumpter, N., Bulpitt, A.: Learning spatio-temporal patterns for predicting object behaviour (1998)
16. Verhein, F., Chawla, S.: Mining spatio-temporal patterns in object mobility databases. *Data Mining and Knowledge Discovery* (2007)
17. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: Proceedings. 18th International Conference on Data Engineering, 2002, pp. 673–684 (2002)
18. Zeinalipour-Yazti, D., Lin, S., Gunopulos, D.: Distributed spatio-temporal similarity search. In: CIKM 2006: Proceedings of the 15th ACM international conference on Information and knowledge management, pp. 14–23. ACM, New York (2006)

A Probabilistic Approach for Computing Approximate Iceberg Cubes^{*}

Alfredo Cuzzocrea, Filippo Furfaro, and Giuseppe M. Mazzeo

ICAR-CNR, I-87036 Cosenza, Italy

and

University of Calabria, I-87036 Cosenza, Italy

{cuzzocrea,furfaro,mazzeo}@si.deis.unical.it

Abstract. An *iceberg cube* is a refinement of a *data cube* containing the subset of cells whose measure is larger than a given threshold (*iceberg condition*). Iceberg cubes are well-established tools supporting fast data analysis, as they filter the information contained in classical data cubes to provide the most relevant pieces of information. Although the problem of efficiently computing iceberg cubes has been widely investigated, this task is intrinsically expensive, due to the large amount of data which must be usually dealt with. Indeed, in several application scenarios, efficiency is so crucial that users would benefit from a fast computation of even incomplete iceberg cubes. In fact, an incomplete iceberg cube could support preliminary data analysis by allowing users to focus their explorations quickly and effectively, thus saving large amounts of computational resources. In this paper, we propose a technique for efficiently computing iceberg cubes, possibly trading off the computational efficiency with the completeness of the result. Specifically, we devise an algorithm which employs a probabilistic framework to prevent cells which are probably irrelevant (i.e., which are unlikely to satisfy the iceberg condition) from being computed. The output of our algorithm is an incomplete iceberg cube, which is efficiently computed and prone to be refined, in the sense that the user can decide to go through the computation of the cells which were estimated irrelevant during the previous invocations of the algorithm.

1 Introduction

Iceberg cubes [3] are powerful tools for materializing interesting multidimensional cubes/views from very large fact tables stored in multidimensional database systems. Basically, iceberg cubes extend conventional data cubes [5], with the specialized condition that the groups of tuples from the source data set to be materialized into cells of the final cube are restricted to those satisfying a given *HAVING* clause. More formally, given a data set S , an iceberg cube $\mathcal{I}$ on S is

^{*} This work was supported by a grant from the Italian Research Project “Open-KnowTech: Laboratorio di Tecnologie per la Integrazione, Gestione e Distribuzione di Dati, Processi e Conoscenze”, funded by MUR.

defined by a *SELECT-GROUP-BY* aggregation on S with an additional clause (expressing the so-called *iceberg condition*) of the form *HAVING* $AggOp(A) > T$, where: (i) $AggOp$ is an SQL aggregation operator, (ii) A is an attribute from the fact table S , and (iii) T is a threshold value. In this paper, we consider the basic iceberg-cube computation problem where $AggOp$ is the *COUNT* operator, as iceberg cubes with complex aggregation predicates can be computed via meaningfully exploiting efficient solutions for the baseline case [7].

Similarly to what happens with traditional data cube computation [2,6,16], as real-life data are often characterized by large dimensionality, computing iceberg cubes incurs in the so-called *curse of dimensionality* problem [3,6]. Therefore, the problem of efficiently computing iceberg cubes has attracted the interest of a growing community of researchers, and is becoming appealing for a large number of modern applications such as *frequent pattern mining* [8,12].

The curse of dimensionality problem and excessive data cube sizes realistically limit the performance of state-of-the-art iceberg cubing algorithms. This limitation is mainly determined by the fact that all the solutions proposed so far aim at obtaining an *exact* computation of iceberg cubes, and neglect the opportunity of introducing some form of approximation. Indeed, in several application scenarios, efficiency is so crucial that users would benefit from a fast computation of even incomplete iceberg cubes. In fact, an incomplete iceberg cube could suffice to effectively support preliminary data analysis by allowing users to focus their explorations quickly and effectively, thus saving large amounts of computational resources. The idea of using approximation for supporting preliminary data explorations has been successfully exploited in several works dealing with OLAP applications. For instance, a number of techniques for providing fast but approximate answers to aggregate range queries have been proposed, such as *histograms* [10], *wavelets* [11], and *sampling* [1].

This paper has been inspired by the following question: *Can we define a method that allows us to efficiently compute iceberg cubes, possibly trading off the efficiency of the computation with the completeness of the result?* In this regard, starting from *BUC* [3] (one of the state-of-the-art techniques for the iceberg cube computation), we propose an iceberg cubing algorithm based on a probabilistic framework which is exploited to estimate whether groups of tuples satisfy the iceberg conditions before aggregating them. Specifically, the source of efficiency of our technique is that the probabilistic estimation prevents aggregations which are unlikely to satisfy the iceberg condition from being computed. Indeed, the probabilistic evaluation can result in “wrong” estimations, yielding a loss of cells w.r.t. the exact iceberg cube. However, this is not a drawback of our technique, as the user is given the possibility to progressively complete the computation of the iceberg cube in the regions of data she is interested in.

2 Preliminaries

In this section we formally introduce the problem of computing an iceberg-cube and provide the basic notations that will be adopted throughout the rest of the

paper. Let S be a data set defined over an n -dimensional domain whose dimensions are $D = \{d_1, d_2, \dots, d_n\}$. We will denote the cardinality of the domain associated with the i -th dimension (with $1 \leq i \leq n$) as w_i . A tuple of S has the form $\langle val_1, val_2, \dots, val_n, m \rangle$, where val_j is a value in the domain of d_j and m is a measure value. Given an aggregate operator op and a set of dimensions $G \subseteq D$, the cuboid on S w.r.t. op and G is the set of tuples resulting from the following SQL query:

```
SELECT G, op(m)
FROM S
GROUP BY G
```

Let op be an aggregation operator, T an integer value, and G a subset of D . The choice of op and G , along with the value of T , define the following *iceberg-query* $Q_{T,G}^{op}$:

```
SELECT G, op(m)
FROM S
GROUP BY G
HAVING count(*) ≥ T;
```

The condition expressed in the **HAVING** clause will be said to be *iceberg condition*. Basically, the answer of $Q_{T,G}^{op}$ on S is a subset of the cuboid on S w.r.t. op and G . Specifically, it results from first grouping the tuples of S by the dimensions in G , then selecting the groups consisting of at least T tuples, and finally evaluating the aggregate operator op on the measure attribute m of the tuples of each of the selected groups. In the following, when op and T are implied, we will denote the iceberg-query $Q_{T,G}^{op}$ as Q_G . The answer of an iceberg query will be said to be *iceberg cuboid*, and will be simply denoted by the set G .

The *iceberg-cube* on S with threshold T consists of the set of iceberg cuboids resulting from all the iceberg-queries defined on S (corresponding to all the different group-bys of the dimensions in D) and will be denoted as $\mathcal{I}_{S,T}^{op}$. Exploiting the **CUBE BY SQL** operator, the iceberg-cube $\mathcal{I}_{S,T}^{op}$ can be obtained by means of the following query:

```
SELECT D, count(*), op(m)
FROM S
CUBE BY D
HAVING count(*) ≥ T;
```

For instance, consider a 4-dimensional data set S representing the amount of sales of a company w.r.t. the dimensions $D = \{customer, product, city, month\}$. The iceberg cube on S consists of 16 (i.e., $2^{|D|}$) iceberg cuboids, corresponding to all the different ways to aggregate sales. All the possible groupings of dimensions of D define the lattice shown in Fig. 1. The lattice consists of $n + 1$ levels, $\{L_0, \dots, L_n\}$, where level L_i (starting to count from the bottom) contains the cuboids obtained by aggregating tuples in S w.r.t. all the possible $\binom{n}{i}$ subset of

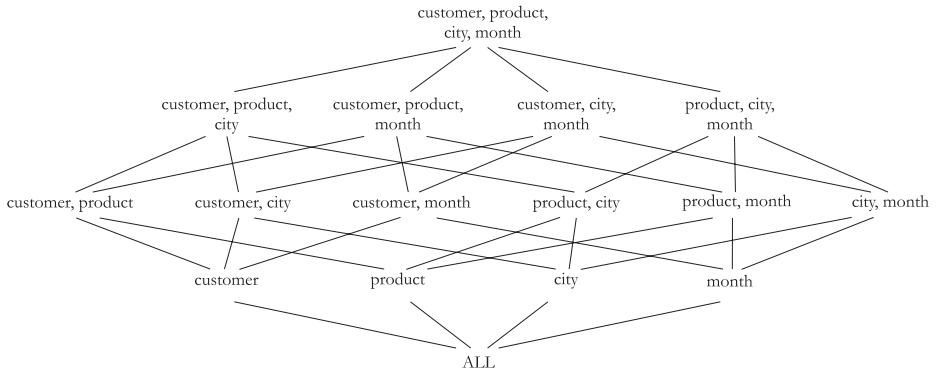


Fig. 1. Lattice of all the groupings of dimensions *customer*, *product*, *city*, *month*

D with cardinality i . We denote as *ALL* the singleton cell cuboid representing the aggregate result of the tuples in S .

3 The Bottom-Up Cubing Algorithm

In this section we briefly recall *BUC* algorithm [3], since our proposal can be viewed as a refinement of *BUC* employing a probabilistic estimation of the cells which should be materialized in the output iceberg cube.

BUC exploits the antimonotonicity property of the COUNT operator to avoid unnecessary computation. This is enabled by the bottom-up traversing of the cuboid lattice during iceberg cube computation, i.e., from the lowest cuboid (containing a single cell, aggregating all the tuples), towards the highest cuboid, where the source tuple are grouped by all the non-measure attributes.

Consider the case of a data set S representing sales defined on the set of dimensions $D = \{customer, product, city, month\}$. Fig. 1 shows the corresponding cuboid lattice. The arcs in the lattice represent the hierarchical relations between pairs of cuboids. For instance, in Fig. 1, the cuboid $\{customer, product\}$ is a super-cuboid of $\{customer, product, city\}$, meaning that cells in $\{customer, product, city\}$ can be obtained from cells in $\{customer, product\}$ by “disaggregating” tuples w.r.t the dimension *city*. Similarly, a cell in $\{customer, product\}$ can be obtained from cells in $\{customer, product, city\}$ by further aggregating them along the dimension *city*.

Fig. 2 shows the *BUC processing tree* for our running example, i.e., how *BUC* traverses the cuboid lattice, in order to compute the iceberg cube. Before going into details, a basic concept of the *BUC* traversing order needs to be introduced. A cell c of a given cuboid G is said to be *descendant cell* of a cell c' in a cuboid G' if G' is a super-cuboid of G and c is obtained from c' by means of disaggregation. In other words, with respect to the original notion of cuboid lattices (mentioned above), a descendant cell is a cell stored in a sub-cuboid, and thus it inherits the corresponding hierarchical data generation properties.

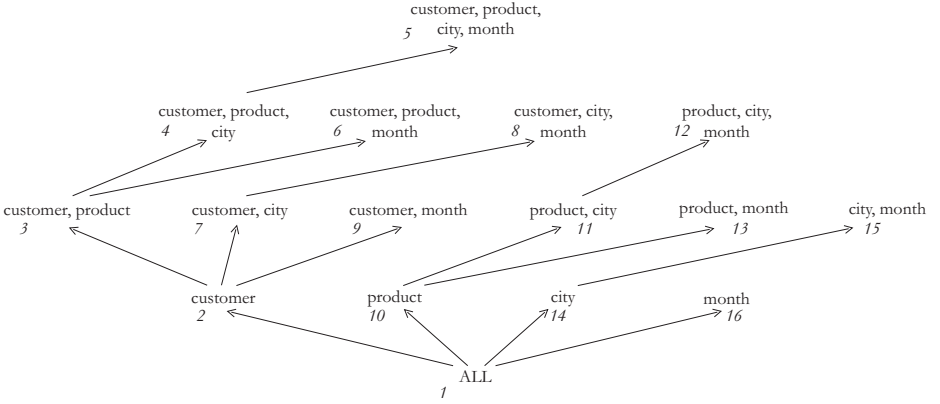


Fig. 2. Processing tree of *BUC* over the lattice of Fig. 1

BUC is driven in the computation by the processing tree as follows. It starts by computing the unique cell in the cuboid ALL by aggregating all the tuples in the data set. Then, it visits the processing tree and, for each node being visited, it evaluates the cells of the corresponding cuboid by disaggregating the cells of the super-cuboid. The antimonotonicity property of COUNT is exploited in *BUC* as follows. Assume, for instance, that $\{customer, product\}$ is the current cuboid during iceberg cube computation. If the iceberg predicate fails on a cell c of $\{customer, product\}$, then all its descending cells (in all the sub-cuboids of $\{customer, product\}$, i.e., $\{customer, product, city\}$, $\{customer, product, city, month\}$ and $\{customer, product, month\}$) are pruned from the computation, since the iceberg predicate will surely fail on these cuboid cells as well. This amenity is the key that allows *BUC* to avoid unnecessary computation, thus outperforming state-of-the-art top-down cubing algorithms (e.g., *MultiWay* [16]).

4 A New Efficient Probabilistic Approach

As explained before, algorithm *BUC* exploits the antimonotonicity of the count operator to gain efficiency in the computation of the iceberg cube. Specifically, at each step, *BUC* does not compute a cell of a cuboid Q_G if it is guaranteed, from the results obtained at the previous steps for the parent cuboid of Q_G , that this cell does not satisfy the iceberg condition. For instance, consider the case that, in our running example, grouping tuples by attribute *product* results in 8 tuples referred to the product *moving walkway*. If the threshold is $T = 10$, then *BUC* does not compute the cells of the cuboid $\{product, customer\}$ having *product*=“*moving walkway*”, as all of them are associated with a count value not greater than 8.

The idea underlying our approach is that of modifying the strategy implemented in *BUC* in the sense that aggregations are not computed when their evaluation is likely (instead of certain) to result in no cell whose count value

satisfies the iceberg condition. For instance, consider the case that, in our running example, grouping tuples by attribute *product* results in 1 000 tuples referred to the product *washer*. Moreover, assume that there are 10 000 customers. This means that, on the average, each customer bought 0.1 washers, and it is reasonable to estimate that there is no customer who bought at least 10 washers. It is worth noting that, in this case, *BUC* would accomplish the computation of the cells of the iceberg cuboid $\{product, customer\}$ with *product*="washer", and it would probably obtain no cell.

In order to implement our strategy, it is mandatory to define a framework for evaluating the probability that, given a cell c of the cuboid G and an attribute $A \notin G$, there is at least one non-null cell c' in the cuboid $G \cup \{A\}$ having the same values of the attributes in G as cell c . To this end, we investigate the following (equivalent) problem: *Given the threshold T and a cell c of a cuboid Q_G , if the count value of c is s , what is the probability that grouping the tuples aggregated in c w.r.t. an attribute $A \notin G$ of cardinality w results in at least one group consisting of at least T tuples?* This probability will be denoted as $\mathcal{P}(s, w, T)$, and its value is given by:

$$\mathcal{P}(s, w, T) = 1 - \frac{\sum_{\gamma=0}^{\lfloor \frac{s}{T} \rfloor} (-1)^\gamma \cdot \binom{w}{\gamma} \cdot \binom{w + s - \gamma \cdot T - 1}{s - \gamma \cdot T}}{\binom{w + s - 1}{s}} \quad (1)$$

The above-reported formula can be explained as follows. Let $F(s, w, T)$ the number of ways of distributing s tuples among w groups such that each group contains at most $T-1$ tuples, and let $H(s, w)$ the number of ways of distributing s tuples among w groups. It is easy to see that $\frac{F(s, w, T)}{H(s, w)}$ is the probability that the distribution of tuples among groups results in no group having at least T tuples, thus $\mathcal{P}(s, w, T)$ is given by $1 - \frac{F(s, w, T)}{H(s, w)}$. Therefore, formula 1 can be proved by showing that

$$F(w, s, T) = \sum_{\gamma=0}^{\lfloor \frac{s}{T} \rfloor} (-1)^\gamma \cdot \binom{w}{\gamma} \cdot \binom{w + s - \gamma \cdot T - 1}{s - \gamma \cdot T}$$

and

$$H(s, w) = \binom{w + s - 1}{s}.$$

The latter easily derives from the fact that $H(s, w)$ corresponds to the number of multisets of cardinality s which can be obtained from b distinct elements (this is equivalent to the number of combinations with repetitions of w elements from which s elements must be selected). The formula for $F(w, s, T)$ can be explained as follows. We denote the absolute value of the j -th addend of the sum (which corresponds to $\gamma = j - 1$) as S^{j-1} . Thus, the first term of the sum is S^0 and is

equal to $H(s, w)$. Hence, it counts all the ways of distributing s tuples among w groups. This value takes into account also the number F^1 of distributions where there exists at least one group containing more than $T - 1$ tuples. Hence, the value of $F(w, s, T)$ is given by $H(s, w) - F^1$. The second term of the sum in the formula of $F(w, s, T)$ (corresponding to $\gamma = 1$) provides an overestimation of F^1 . In fact, S^1 is the product of $\binom{w}{1}$ with $\binom{w + s - T - 1}{s - T}$. The former term represents the ways of choosing an integer in the interval $[1..w]$, while the latter is equal to $H(s - T, w)$. Thus, the rationale of this expression is that of evaluating F^1 by counting the choices of a group of tuples among b possible ones, assigning T tuples to it, and then counting the number of distributions of the remaining $s - T$ tuples among the b groups. Indeed, S^1 overestimates F^1 , as it counts twice each configuration where exactly two groups of tuples contain at least T tuples, three times each configuration where exactly three groups contain at least T tuples, and so on. Intuitively enough, adding (resp., subtracting) each term S^j (for $j > 1$) overcompensates (resp., undercompensates) the estimation of the overall number of configurations provided by the previous terms of the sum.

On the basis of the probabilistic framework introduced above, we propose an algorithm for computing an approximate iceberg cube (*AIC*). The algorithm implements the same strategy as *BUC*, except that it exploits formula (1) to estimate whether it is reasonable (w.r.t. to a probability threshold) to go through the computation of cells descending from an already computed one. The algorithm invokes the below-reported (recursive) function `computeDimension` on the source data set, and specifies the cardinalities of dimension domains, the count threshold, and the probability threshold. The last argument (the starting dimension) of the main invocation of `computeDimension` is set to 0 and is used to drive the recursive invocations through the processing tree, as it will be clearer in the following.

The generic invocation of function `computeDimension` takes as first argument a set of tuples S , which is the set of tuples aggregated into a cell c of the iceberg cube computed by the previous invocations of `computeDimension`, and computes (through recursive invocations) the cells of the iceberg cube descending from c . We recall that, at the first invocation, the input set S is the whole data set over which the iceberg cube must be computed (thus, it corresponds to the cell ALL of the iceberg cube). Function `computeDimension` performs its task as follows. First, it groups the tuples in S by attribute d , and stores these groups in an array of tuple sets g . Thus, the i -th cell of g contains the tuples of S where attribute d is equal to the i -th value of the domain of dimension d . Specifically, at the first invocation, $d = 0$ and g consists of a unique cell containing the whole data set S . After grouping the tuples, each cell stored in g contains the set of tuples corresponding to a candidate cell of the iceberg cube. Then, the array g is scanned and each cell $g[i]$ is added to the output iceberg cube iff it contains at least T tuples (i.e., it satisfies the iceberg condition). If this is the case, the computation proceeds on the tuples in cell $g[i]$, considering all the dimensions following d . Thus, for each $d' > d$ (with $d' \leq |D|$), the probability

Function computeDimension

Input: A data set S , the array w of cardinalities of dimension domains,
a count threshold T , a probability threshold p , a starting dimension d

Output: The set of cells of the approximate iceberg-cube on S w.r.t. T and p
obtained by walking through the processing tree starting from dimension d

begin

```

TupleSet[ ] g=Group( $S, d$ ); //  $g$  is the array of tuple sets resulting from grouping  $S$  by  $d$ ;
Cube l= new Cube(); // a new cube is instantiated as an empty set of cells;
for ( $i=1$ ;  $i \leq g.length$ ;  $i++$ )
    if ( $size(g[i]) \geq T$ ) {
        l.add( $g[i]$ ); // cell  $g[i]$  is added to the cube to be returned;
        for ( $j=d+1$ ;  $j \leq w.length$ ;  $j++$ )
            if ( $\mathcal{P}(size(g[i]), w[j], T) > p$ )
                l.union(computeDimension( $g[i]$ ,  $w$ ,  $T$ ,  $p$ ,  $j$ )); // the cube to be returned is
                                                                // augmented with the result of the recursive
                                                                // invocations of computeDimension;
            else
                l.mark( $g[i], j$ ); //  $g[i]$  is marked as a cell refinable w.r.t. dimension  $j$ 
    }
return l;
end;
```

Fig. 3. A bottom-up algorithm for computing iceberg-cubes based on a probabilistic approach

$\mathcal{P}$ that at least T tuples in $g[i]$ have the same value on d' is computed and, if $\mathcal{P}$ exceeds the input probability threshold p , tuples in $g[i]$ are processed by recursively invoking **computeDimension** w.r.t. d' . Otherwise, if $\mathcal{P} < p$, cell $g[i]$ is marked as a “cell refinable w.r.t. dimension d ”, that is a cell which can be possibly refined by further invocations of **computeDimension**. That is, during the exploration of the (possibly) incomplete iceberg cube returned by our algorithm, users can find a cell c marked as refinable and decide to force the evaluation of its descendant cells, by running the algorithm on the set of tuples underlying c w.r.t. the dimension d associated with c .

It is worth noting that our algorithm coincides with *BUC* when the probability threshold is set to 0, as in this case descending cells of an already computed cell are always computed if the parent cell has a count value satisfying the iceberg condition.

The cost of function **computeDimension** depends on the cost of function **group** and the cost of evaluating formula (1) for $\mathcal{P}$. As regards the former, tuples can be grouped in time linear w.r.t. the cardinality of S (as explained for *BUC* algorithm). Formula (1) can be efficiently computed by observing that it can be rewritten in the form: X

$$\mathcal{P}(s, w, T) = 1 - \sum_{\gamma=0}^{\lfloor \frac{s}{T} \rfloor} \frac{(-1)^\gamma \cdot \binom{w}{\gamma} \cdot \binom{w+s-\gamma \cdot T-1}{s-\gamma \cdot T}}{\binom{w+s-1}{s}} = 1 - \sum_{\gamma=0}^{\lfloor \frac{s}{T} \rfloor} a_\gamma \quad (2)$$

where the first term of the sum ($\gamma = 0$) is

$$a_0 = \frac{\binom{w+s-1}{s}}{\binom{w+s-1}{s}} = 1.$$

and the ratio between two subsequent terms (for $\gamma > 0$) of the sum is:

$$\begin{aligned} \frac{a_\gamma}{a_{\gamma-1}} &= \frac{(\gamma - w - 1) \cdot (w + s - \gamma \cdot T - 1)! \cdot (s - \gamma \cdot T + T)!}{\gamma \cdot (w + s - \gamma \cdot T - 1 + T)! \cdot (s - \gamma \cdot T)!} = \\ &= \frac{\gamma - w - 1}{\gamma} \cdot \frac{\prod_{k=1}^T (s - \gamma \cdot T + k)}{\prod_{k=1}^T (w + s - \gamma \cdot T - 1 + k)} = \frac{\gamma - w - 1}{\gamma} \cdot \prod_{k=1}^T \frac{s - \gamma \cdot T + k}{w + s - \gamma \cdot T - 1 + k} \end{aligned}$$

Thus, each term a_γ (with $\gamma > 0$) can be computed by means of $O(T)$ floating point operations. Since the value of $\mathcal{P}(s, w, T)$ results from $\lfloor \frac{s}{T} \rfloor$ terms, we obtain that the overall computation of $\mathcal{P}(s, w, T)$ can be accomplished by means of $O(s)$ floating-point operations. Each invocation of `computeDimension` computes $\mathcal{P}$ for each cell of g , thus the overall cost of computing the probabilities is $O(|S|)$. However the cost of computing the probabilities can be considered negligible w.r.t. that of grouping tuples, as the former can be always performed in main memory, while the latter task requires disk accesses to retrieve the tuples to be grouped.

Hence, the overall cost of a single invocation of `computeDimension` (which takes into account the cost of function `group` and the evaluation of $\mathcal{P}$, disregarding the cost of recursive invocations) is linear w.r.t. the cardinality of the input data set S . The overall cost of computing the (approximate) iceberg cube depends on the number of recursive invocations of `computeDimension` raised by its main invocation (taking as argument the whole data set). The impact of using a probability threshold to reduce the number of recursive invocations of `computeDimensions` cannot be determined but experimentally. This is the matter investigated in the following section.

5 Experimental Results

In order to test the effectiveness of our proposal, we performed experiments comparing the efficiency and the accuracy of our algorithm with *BUC* (the state-of-the-art technique for the exact computation of iceberg cubes), for different

values of probability and count thresholds. We measured the efficiency of both the algorithms by counting the number of calls of the **group** function for different probability and count thresholds. Then, we computed the speed-up for different values of the parameters. Specifically, being p the probability threshold and T the count threshold, the speed-up of the probabilistic algorithm w.r.t. the classical *BUC* algorithm was measured as $S(p, T) = C(0, T)/C(p, T)$, where $C(x, y)$ denotes the number of calls of function **group** in the case that the probability and count thresholds are x and y , respectively¹.

We adopted the number of calls of the **group** function as measure of the computation efficiency because this is the operation with the largest cost. In fact, when the database is large, its tuples cannot fit main memory, and their grouping must be performed by means of several disk accesses². Therefore, it is possible to disregard the cost of the other operations, including the computation of $\mathcal{P}(w, s, T)$, which can be always performed in main memory by means of $O(s)$ floating point operations, as discussed in Sect. 4.

As regards the accuracy of the computation, it was measured as $A(p, T) = M(p, T)/M(0, T)$, where $M(x, y)$ denotes the number of materialized cells in the case that the probability and count thresholds are x and y , respectively. That is, $A(p, T)$ is the ratio between the number of cells materialized by our algorithm and the number of cells in the exact iceberg cube returned by *BUC*.

We tested the performances of *AIC* on several synthetic data sets. Here we present the results obtained on two different data sets, which are sufficient to discuss the performances of *AIC*. The synthetic data sets are 4-dimensional, each representing 1 million sales defined on the following dimensions (in brackets, the cardinality of each dimension domain is reported): *customer* (50 000), *product* (10 000), *month* (60), *city* (24). In order to simulate correlation between attributes, we assumed that each customer is assigned a *peak product* and a *peak city*, both chosen according to a zipf distribution (on the product domain and on the city domain, respectively), and each product is assigned a *peak month* chosen according to a uniform distribution (on the month domain). Then, we generated each of the million sales as follows: (i) the customer c is chosen according to a zipf distribution, (ii) the city is chosen according to a normal distribution centered in the peak month of c , (iii) the product p is chosen according to a normal distribution centered in the peak product of c , and (iv) the month is chosen according to a normal distribution centered in the peak month of p . The parameters of the zipf distributions were set to 0.3 and, respectively, 1.0 for the first and the second data set, that will be denoted as $D_{0.3}$ and $D_{1.0}$ in the following. The variances σ^2 of the normal distributions were chosen such that a range of size $6 \cdot \sigma$ is at least as large as one half of the considered domain size.

¹ $C(0, T)$ represents the number of tuple groupings performed by *BUC*, since, as remarked before, our algorithm coincides with *BUC* in the case that $p = 0$.

² A more detailed measure of efficiency should consider the number of disk accesses. However, we did not adopt this measure because the grouping operation can be performed differently by different DBMSs.

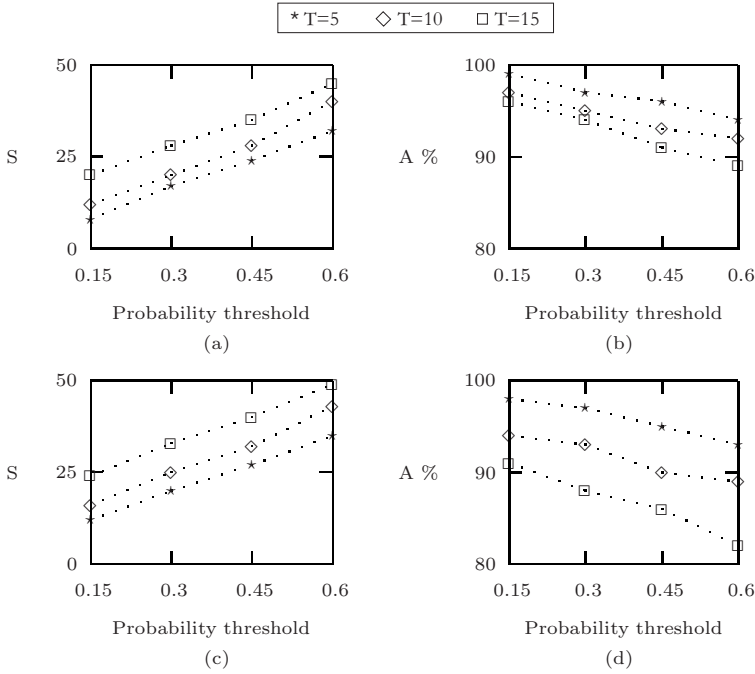


Fig. 4. Speedup (a,c) and accuracy (b,d) of *AIC* on a $D_{0.3}$ (a,b) and $D_{1.0}$ (c,d)

Fig. 4 depicts the results obtained by running *AIC* on $D_{0.3}$ (a,b) and $D_{1.0}$ (c,d). As shown in Fig. 4 (a,c), the employment of *AIC* results in a speed-up w.r.t. *BUC*, ranging from 8 to 50 in the considered cases. These diagrams also show that the speed-up increases with both the count and the probability threshold. Correspondingly, the accuracy of *AIC*, as shown in Fig. 4 (b,d), decreases with both the count and the probability threshold. Intuitively enough, as the speedup increases, the probability of losing cells that must be materialized gets higher, since a larger number of groupings which might result in cells satisfying the iceberg condition is not performed. Therefore, a larger number of cells are likely to be lost. This effect becomes more relevant as T increases, since $\mathcal{P}(s, w, T)$ (see formula 1) decreases more rapidly than the number of cells whose count exceeds T , due to the correlation among attributes and the data skewness. As a matter of fact, the sensitivity on the data skewness can be observed by comparing the results shown in Fig. 4 (b,d): for the same T , the accuracy on $D_{0.3}$ is better than that on $D_{1.0}$, being $D_{0.3}$ less skewed than $D_{1.0}$.

The effectiveness of our approach can be evaluated by simultaneously analyzing the results reported in the above-mentioned diagrams. From these diagrams, it turns out that the gain in terms of groupings saved is large compared with the number of “lost” cells (which is between the 1% and the 18% of the overall number of cells of the actual iceberg cube). This means that our probabilistic framework effectively estimates, in most cases, whether a cell satisfies the iceberg

condition. Furthermore, the majority of the lost cells have count slightly larger than the threshold (almost all lost cells have count within the range $[T..2 \cdot T]$). It is worth noting that losing a number of cells is not a severe drawback, due to the possibility of progressively refining the result starting from marked cells.

We remark that the order in which dimensions were processed by both *BUC* and *AIC* was the descending order of domain cardinalities, as proposed in [3]. However, we also tried other orderings, obtaining larger execution times for *BUC* (about twice, in the case of dimensions processed in ascending order), whereas *AIC* turned out to be almost insensitive to the dimension ordering (for the considered values of probability threshold). This can be considered as a further benefit of *AIC*, which, differently from *BUC*, does not require a data preprocessing for determining the actual cardinality of dimension domains.

6 Related Work

In this section, we provide an overview of state-of-the-art iceberg cubing algorithms. Basically, state-of-the-art approaches pursue two main goals: (i) *sharing computation overheads as much as possible*; (ii) *applying the iceberg condition in the cuboid lattice as deep as possible*. These two principles have inspired both top-down and bottom-up iceberg cube computation strategies. For instance, the computation strategy of *BUC* is based on the second principle. On the other hand, it is worth noting that these principles are pretty controversial, and cannot be accommodated simultaneously. This evidence has stimulated the proliferation of a novel class of iceberg cube computation methods, the so called *hybrid methods*. Devising innovative strategies beyond capabilities and performance of *BUC* is the main goal of these proposals. Nevertheless, even though with a dependency lower than first-generation proposals [12], the performance of hybrid methods is still heavily affected by the curse of dimensionality problem and excessive data cube sizes.

Top-down strategies do not perform well with iceberg cubes, being more targeted to the computation of conventional data cubes. The reason of this is that, contrary to bottom-up ones, these algorithms do not take any advantage from the antimonotonicity property, as discussed in Section 1. However, top-down algorithms can be still used to compute iceberg cubes, as follows: (i) compute the full-materialized data cube first, and (ii) apply the iceberg predicate then in order to prune (materialized) cells that do not satisfy the latter predicate. Obviously, this approach is plausible only for small-sized cubes and low dimension numbers, whereas it does not scale well when cubes grow in dimension number and size.

MultiWay [16] is one of the most representative top-down iceberg cubing algorithm. The main idea that underlies *MultiWay* consists in using a compressed sparse array to load tuples from the target fact table, and then partitioning this array in ad-hoc *chunks* supporting the simultaneous computation of multiple cuboids in one pass only. The key of this approach relies in the chunk computation order, which must be arranged meaningfully [16].

Among the hybrid iceberg cubing algorithms, *StarCubing* [12] is the most representative one. The innovation carried out by *StarCubing* relies in the amenity of combining bottom-up and top-down iceberg cubing methodologies in order to devise a novel approach that efficiently supports simultaneous multidimensional aggregations, which is a useful feature for several Data Warehousing and OLAP applications. The resulting iceberg cubing methodology that underlies *StarCubing* is indeed quite complex. First, *StarCubing* introduces the notion of *shared dimensions* of the cuboid lattice in order to achieve an “intermediate” *spanning tree* of the lattice that combines the strengths of both bottom-up and top-down traversing orders. The relationships among cuboids define the so-called *star-tree*, a novel data structure that efficiently supports both a-priori pruning like *BUC*, which avoids unnecessary computation, and also simultaneous multidimensional aggregations like *MultiWay*, which is instead not supported by *BUC*.

Apart from the previous proposals, which, in summary, constitute the state-of-the-art iceberg cubing algorithms, other interesting research initiatives that focus on research aspects related to iceberg cubes are the following. [7] deals with the problem of computing iceberg cubes with complex OLAP-like aggregation predicates. [4] advocates *high-performance distributed and parallel computation methodologies* to solve the problem of computing very large iceberg cubes. [14] introduces *BP-Cubing*, a novel *bound prune* approach for iceberg cubing that is able to handle *non-antimonotone* aggregation operators via devising meaningful *bounding formulas* for both simple and complex SQL aggregate functions. This framework has also been successfully exploited for efficiently computing *iceberg quotient cubes* [15]. [13] moves the attention from the classical exclusive pruning strategy to novel *inclusive* and *anti-pruning* strategies. Finally, [9] considers the interesting case of computing iceberg cubes on top of XML multidimensional data repositories.

7 Conclusions and Future Work

A novel iceberg cube computation paradigm has been proposed and experimentally assessed in this paper. While state-of-the-art cubing algorithms accomplish the exact computation of the cube, we pursue an incomplete but faster computation, which results in an iceberg cube refinable under user control. Our approach is driven by a probabilistic framework which is used to estimate whether a cell satisfies the iceberg condition (that is, whether it should be materialized) on the basis of the aggregate data associated with already computed cells.

Due to the possibility of progressively refining the iceberg cube under user control, our method is highly flexible, so that it can be effectively used as baseline component for advanced OLAP-based analysis tools.

Future work will be mainly devoted to investigate how to take into account correlation among attributes in the probabilistic evaluation of the cells which are likely to satisfy the iceberg condition. Furthermore, we will focus on extending our approach to deal with aggregate operators other than *count*, which may be useful in a number of OLAP-like application contexts.

References

1. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: Join Synopses for Approximate Query Answering. In: ACM SIGMOD (1999)
2. Agarwal, S., Agrawal, R., Deshpande, P.M., Gupta, A., Naughton, J.F., Ramakrishnan, R., Sarawagi, S.: On the Computation of Multidimensional Aggregates. In: VLDB (1996)
3. Beyer, K., Ramakrishnan, R.: Bottom-Up Computation of Sparse and Iceberg Cubes. In: ACM SIGMOD (1999)
4. Chen, Y., Dehne, F., Eavis, T., Rau-Chaplin, A.: PnP: Parallel And External Memory Iceberg Cube Computation. In: IEEE ICDE (2005)
5. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Crosstab, and Sub-Total. In: IEEE ICDE (1996)
6. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing Data Cubes Efficiently. In: ACM SIGMOD (1996)
7. Han, J., Pei, J., Dong, G., Wang, K.: Efficient Computation of Iceberg Cubes with Complex Measures. In: ACM SIGMOD (2001)
8. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidate Generation. In: ACM SIGMOD (2000)
9. Ming-fei Jian, F., Pei, J., Wai-chee Fu, A.: IX-Cubes: Iceberg Cubes for Data Warehousing and OLAP on XML Data. In: ACM CIKM (2007)
10. Poosala, V., Ioannidis, Y.E.: Selectivity Estimation without the Attribute Value Independence Assumption. In: VLDB (1997)
11. Vitter, J.S., Wang, M., Iyer, B.: Data Cube Approximation and Histograms via Wavelets. In: ACM CIKM (1998)
12. Xin, D., Han, J., Xiaolei, L., Shao, Z., Wah, B.W.: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: The StarCubing Approach. *IEEE Trans. on Knowledge and Data Engineering* 19(1) (2007)
13. Zhang, X., Lienhua Chou, P.: Multiway Pruning for Efficient Iceberg Cubing. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 203–212. Springer, Heidelberg (2006)
14. Zhang, X., Lienhua Chou, P., Dong, G.: Efficient Computation of Iceberg Cubes by Bounding Aggregate Functions. *IEEE Trans. on Knowledge and Data Engineering* 19(7) (2007)
15. Zhang, X., Lienhua Chou, P., Ramamohanarao, K.: Computing Iceberg Quotient Cubes with Bounding. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 145–154. Springer, Heidelberg (2006)
16. Zhao, Y., Deshpande, P.M., Naughton, J.F.: An Array-based Algorithm for Simultaneous Multidimensional Aggregates. In: ACM SIGMOD (1997)

Noise Control Boundary Image Matching Using Time-Series Moving Average Transform

Bum-Soo Kim, Yang-Sae Moon, and Jinho Kim

Department of Computer Science, Kangwon National University
192-1, Hyoja2-Dong, Chuncheon, Kangwon 200-701, Korea
{bskim, ysmoon, jhkim}@kangwon.ac.kr

Abstract. To achieve the noise reduction effect in boundary image matching, we exploit the *moving average transform* of time-series matching. Our motivation is that using the moving average transform we may reduce noise in boundary image matching as in time-series matching. We first propose a new notion of *k-order image matching*, which applies the moving average transform to boundary image matching. A boundary image can be represented as a sequence in the time-series domain, and our *k-order image matching* identifies similar boundary images in this time-series domain by comparing the *k-moving average* transformed sequences. Next, we propose an index-based method that efficiently performs *k-order image matching* on a large image database, and prove its correctness. Moreover, we present its index building and *k-order image matching* algorithms. Experimental results show that our *k-order image matching* exploits the noise reduction effect, and our index-based method outperforms the sequential scan by one or two orders of magnitude.

1 Introduction

Owing to advances in storage and computing power, there have been many research efforts on time-series matching to exploit large time-series databases [1, 4, 7, 10]. In addition, there have been several recent attempts to apply these time-series matching techniques to practical applications such as handwritten recognition, image matching, query by humming, and biological sequence matching [7, 14, 15, 9]. Among these applications, in this paper we focus on the *boundary image matching* that converts boundary images to time-series and identifies similar images using time-series matching on those time-series [7, 14].

Using the moving average transform in boundary image matching we reduce distortions of matching results caused by noise. The traditional research of reducing noise has focused on an image itself rather than a time-series [2, 5]. In contrast, we reduce the noise in a time-series converted from an image rather than the image itself. For this time-series approach, we exploit the moving average transform used in time-series matching as a preprocessing technique [11, 12]. This idea is derived from an observation that the moving average transform reduces noise of time-series. That is, we are motivated by an intuition that using the moving average transform we may reduce noise in the image domain as well as in the time-series domain.

To exploit the moving average transform in boundary image matching, we propose a new notion of *k-order similar*. We say two boundary images are *k-order similar* if their transformed time-series become similar after performing *k-order* moving average transform. We now use this notion of *k-order similar* in boundary image matching, which we call *k-order image matching*. After then, we propose an efficient solution for this *k-order* image matching in large image databases.

We propose an index-based method for the efficient *k-order* image matching. This index-based approach uses a multidimensional index after transforming high-dimensional time-series to low-dimensional points. In this paper, we formally prove the correctness of the index-based method by showing that it incurs no false dismissal. We then present the index-building and the *k-order* image matching algorithms for the index-based method.

Through extensive experiments, we show the effectiveness of *k-order* image matching and the superiority of our index-based method. Experimental results show that, as the order *k* increases, the more images with noise are identified as similar to the given query image. This means that our *k-order* image matching works well even though boundary images contain some noise. To show the performance superiority of our index-based method, we next compare its elapsed time with that of the sequential scan. Compared with the sequential scan, our index-based method reduces the elapsed time by one or two orders of magnitude since it prunes many unnecessary images through the index-based search. According to these results, we believe that our *k-order* image matching and its index-based solution provide a very practical way of realizing the noise control boundary image matching.

2 Related Work

2.1 Time-Series Matching

A time-series is a sequence of real numbers representing values at specific time points. (Hereafter, we use *time-series* and *sequences* interchangeably.) Finding data sequences similar to the given query sequence from the database is called *time-series matching* [1, 4, 6, 7, 10]. We say two sequences *X* and *Y* are *similar* if the distance $D(X, Y)$ is less than or equal to the user specified tolerance ϵ [1, 4, 10]. In this paper we focus on the simplest similarity model that uses the Euclidean distance and the ϵ -based range query. Given two sequences $X = \{X[1], X[2], \dots, X[n]\}$ and $Y = \{Y[1], Y[2], \dots, Y[n]\}$, the Euclidean distance $D(X, Y)$ is defined as $\sqrt{\sum_{i=1}^n (X[i] - Y[i])^2}$. Besides this Euclidean distance-based model, several similarity models were proposed. For these other models, readers refer to [11, 12] for preprocessing techniques, [6, 8] for dynamic time warping (DTW) distance, and [6, 7] for *k*-NN queries.

In this paper we use the whole matching, where the lengths of data and query sequences are all identical. The first solution of whole matching by Agrawal et al. [1] consists of preprocessing, range search, and post-processing steps. In

the preprocessing step, each data sequence of length n is transformed into an f -dimensional point ($f \ll n$, we call it *lower-dimensional transformation*), and the transformed points are stored in an f -dimensional R*-tree [1, 11]. In the range search step, a query sequence is similarly transformed into an f -dimensional point, and a range query is constructed using that point and the tolerance ϵ . By evaluating the range query on the index, the *candidates* that are potentially similar to the query sequence are identified. The lower-dimensional transformation guarantees there is no *false dismissal*, but may cause *false alarms* [1, 4, 10]. Thus, in the post-processing step, for each candidate sequence obtained, the actual data sequence is accessed from the disk; the distance from the query sequence is computed; and the candidate is discarded if it is a false alarm.

Moving average transform is useful in finding the trend of time-series data by reducing the effect of noise [8, 11]. For a given sequence $S = \{S[1], \dots, S[n]\}$, the k -moving average transformed sequence $S^k = \{S^{(k)}[1], \dots, S^{(k)}[n - k + 1]\}$ is computed as $S^{(k)}[i] = \frac{1}{k} \sum_{j=i}^{i+k-1} S[j]$. Here, k is called the *moving average order* or simply the *order*, and it can be varied according to the type of applications and the degree of noise reduction [11]. Rafiei and Mendelzon [12] proposed a solution to whole matching that supports the moving average transform of arbitrary order. For a sequence of length n and the order k , they computed the last $(k-1)$ entries of the k -order moving average transformed sequence using the first $(k-1)$ entries of the original sequence in a circular manner. Recently, Moon and Kim [11] proposed a single-index solution to subsequence matching that supports the moving average transform of arbitrary order. They presented a notion of *poly-order moving average transform* and used it in supporting arbitrary orders.

2.2 Boundary Image Matching

Image matching [5] that finds similar data images to the given query image is one of the most important research topics in image processing areas. In image matching, colors [13], textures [3], or shapes [7, 14] were used as major features. Among these features, we focus on the shape features of an image. Main considerations in the shape-based image matching are object boundaries contained in an image. In this paper we use the centroid contour distance (*CCD* in short) [7, 14], which maps a boundary image to a time-series using the distance of each boundary point from the centroid. Figure 1 shows an example of converting an image to a time-series by CCD.

Recently, two novel works were reported to use time-series of boundary images. First, using the rotation-invariant property of DFT magnitudes Vlachos et al. [14] proposed a novel solution to rotation-invariant image matching. Their method, however, has a weak point that matching accuracy may be worse due to not using DFT phase information. Second, Keogh et al. [7] showed that their tight lower bound LB-Keogh [8] could also be used in rotation-invariant image matching and provided a novel solution for the DTW distance. Their solution is excellent, but it does not consider noise reduction to be solved in this paper.

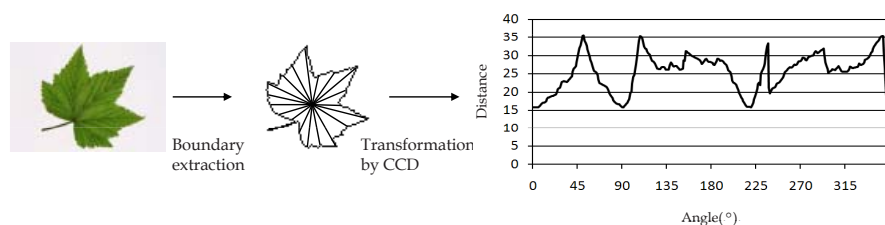


Fig. 1. An example of converting an image to a time-series by CCD

3 Motivation of the Research

To use time-series matching for boundary images, we formally define similarity between two boundary images and boundary image matching.

Definition 1. Let A and B be boundary images and X and Y be their corresponding time-series. Boundary images A and B (or time-series X and Y) are said to be similar if $D(X, Y) \leq \epsilon$. $\square$

Definition 2. Time-series converted from boundary images of the image database are called data sequences, and a database that stores those time-series is called the image time-series database. A time-series converted from a given query boundary image is a query sequence. Given a query sequence and the tolerance, finding data sequences similar to the query sequence from the image time-series database is called boundary image matching. $\square$

Several papers on time-series matching used preprocessing techniques to reduce distortions of time-series [11, 12]. Examples of preprocessing techniques include moving average transform, shifting & scaling, and normalization. We note that, if we reduce distortions of time-series using the preprocessing techniques, we may reduce distortions of original boundary images. That is, by using preprocessing techniques in the time-series domain, we may get an effect of reducing distortions in the image domain. This is our motivation. In particular, we focus on the moving average transform since it reduces noise of time-series data.

Figure 2 shows how boundary images and their time-series are changed by the moving average transform. Figure 2(a) shows original images with some noise and their boundary images. Figure 2(b) shows the corresponding time-series, and Figure 2(c) its 8-moving average transformed time-series. Figure 2(d) shows the reconstructed boundary images from the transformed time-series of Figure 2(c). Comparing Figure 2(a) with Figure 2(d), we can graphically confirm that image boundaries are smoothened by the moving average transform, which means that noise of images are partly eliminated by the moving average transform.

The moving average order k plays an important role in controlling the degree of noise reduction. Figure 3 shows the change of k and its influence on the noise reduction. Figure 3(a) shows an original image with some noise, and Figure 3(b) its boundary image. Figures 3(c) to 3(e) show the reconstructed boundary images from the k -moving average transformed sequences, where k is 2, 4, and 8.

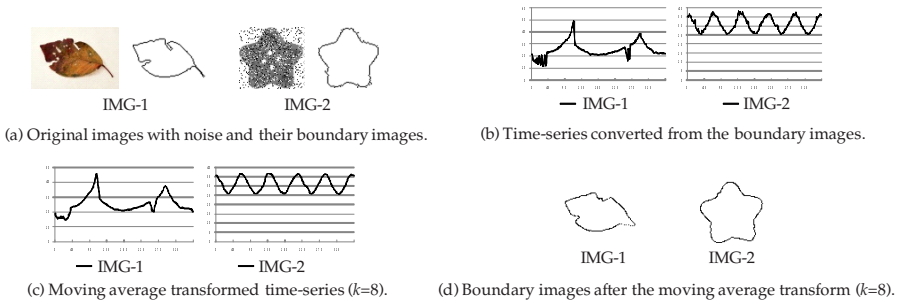


Fig. 2. Effect of reducing noise of boundary images by the moving average transform

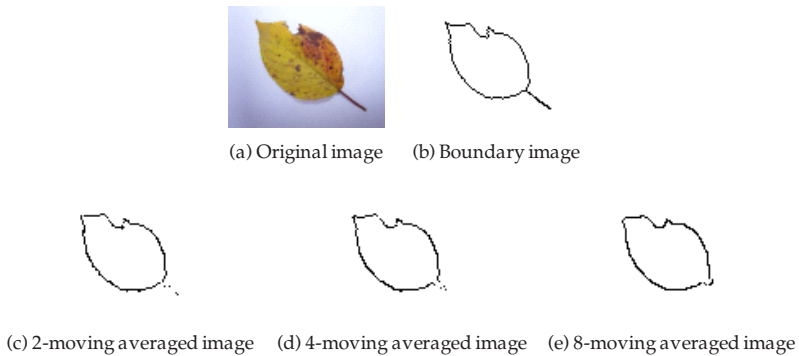


Fig. 3. Noise reduction effect of a boundary image by the moving average order

As shown in Figure 3, as the order k increases, the noise reduction effect also increases. Thus, supporting arbitrary orders is necessary, and our image matching system permits users to control the order k as an input value.

4 Boundary Image Matching Using Moving Average Transform

4.1 The Concept

In Section 3, we already defined similarity between boundary images using their time-series, and described boundary image matching based on that similarity. In Definitions 1 and 2 of Section 3, however, we do not consider the noise reduction effect by the moving average transform. In particular, the degree of noise reduction may vary by the order k . Thus, we need to redefine similarity between boundary images by considering the order k . For this purpose, we first redefine k -moving average transform for the time-series of boundary images.

Definition 3. If $X = \{X[1], \dots, X[n]\}$ is a time-series of a boundary image, its k -moving average transformed sequence $X^{(k)} = \{X^{(k)}[1], \dots, X^{(k)}[n]\}$ is defined as Eq. (1):

$$X^{(k)}[i] = \frac{1}{k}(X[i\%n] + X[(i+1)\%n] + \dots + X[(i+k-1)\%n]) = \frac{1}{k} \sum_{j=i}^{i+k-1} X[j\%n], \quad (1)$$

where $1 \leq i \leq n$, $1 \leq k \leq n-1$, and ‘ $\%$ ’ is a modular operator. $\square$

Definition 3 is the same as Rafiei et al.’s circular definition. In a boundary image, the subsequent boundary point of the last boundary point is the first boundary point, and this means that, in its corresponding time-series, the subsequent entry of the last entry is the first entry. Therefore, the extended moving average transform is reasonable for handling the time-series of boundary images.

We next extend the definition of similarity by considering the moving average transform.

Definition 4. Let A and B be boundary images, X and Y be the corresponding time-series of length n , and $X^{(k)}$ and $Y^{(k)}$ be the k -moving average transformed sequences. Boundary images A and B (or time-series X and Y) are said to be k -order similar if $D(X^{(k)}, Y^{(k)}) \equiv \sqrt{\sum_{i=1}^n (X^{(k)}[i] - Y^{(k)}[i])^2} \leq \epsilon$ holds. $\square$

Previous works [7, 14] on boundary image matching did not consider any pre-processing technique, and they used the distance itself as the similarity measure. In contrast, in order to consider the moving average transform, in Definition 4 we compute the distance between the k -moving average transformed sequences rather than the original sequences. Using the k -order similarity we now extend the definition of boundary image matching.

Definition 5. Given a query sequence Q of a query image, the tolerance ϵ , and the moving average order k , finding data sequences that are k -order similar to the given query sequence from the image time-series database is called k -order image matching. $\square$

To efficiently handle a large image time-series database, we need to use an index as many time-series matching solutions did. For this, we show that the lower-dimensional transformation of time-series matching can also be used in our k -order image matching.

Lemma 1. Let the transformation F convert sequences S and Q of length n to sequences S_F and Q_F of length f , and the lower-bounding condition of Eq. (2) hold for S and Q . Then, F also satisfies the lower-bounding condition of Eq. (3) for time-series X and Y of boundary images.

$$D(S, Q) \geq D(S_F, Q_F) \equiv \sqrt{\sum_{i=1}^f (S_F[i] - Q_F[i])^2} \quad (2)$$

$$D(X^{(k)}, Y^{(k)}) \geq D(X_F^{(k)}, Y_F^{(k)}) \equiv \sqrt{\sum_{i=1}^f (X_F^{(k)}[i] - Y_F^{(k)}[i])^2} \quad (3)$$

In Eq. (3), $X_F^{(k)}$ and $Y_F^{(k)}$ are f -dimensional sequences transformed from $X^{(k)}$ and $Y^{(k)}$ by F , respectively.

Proof. If we replace S and Q in Eq. (2) with $X^{(k)}$ and $Y^{(k)}$, Eq. (2) becomes identical to Eq. (3). Thus, if Eq. (2) holds for S and Q , Eq. (3) also holds for $X^{(k)}$ and $Y^{(k)}$. $\square$

Next, to support arbitrary orders, we use the poly-order moving average transform (*poly-order transform* in short) proposed by Moon and Kim [11]. They defined the poly-order transform for subsequence matching, however, we handle image time-series instead of subsequences. Thus, we redefine the poly-order transform for image time-series rather than subsequences.

Definition 6. Given a time-series X of a boundary image and a set of orders $\mathbb{K} = \{k_1, k_2, \dots, k_m\}$, the poly-order (moving average) transformed set of X on $\mathbb{K}$, denoted by $X^{\mathbb{K}}$, is defined as $\{X^{(k_i)} \mid 1 \leq i \leq m\}$. $\square$

Using the poly-order transform incurs no false dismissal as follows.

Lemma 2. If an order k is an element of a set $\mathbb{K}$ (i.e., $k \in \mathbb{K}$), and X and Y are image time-series, then the lower-bounding condition of Eq. (4) holds.

$$D(X^{(k)}, Y^{(k)}) \geq D(X^{(k)}, \text{MBR}(Y^{(\mathbb{K})})) \quad (4)$$

In Eq. (4), $\text{MBR}(Y^{(\mathbb{K})})$ is an n -dimensional minimum bounding rectangle (MBR) that bounds all sequences contained in $Y^{(\mathbb{K})}$.

Proof. $Y^{(k)}$ is contained in $Y^{(\mathbb{K})}$ by Definition 6. The n -dimensional point $Y^{(k)}$ is contained in the n -dimensional MBR $\text{MBR}(Y^{(\mathbb{K})})$ by the definition of MBR. Thus, the distance from an arbitrary point to $Y^{(k)}$ is greater than or equal to the distance from that point to $\text{MBR}(Y^{(\mathbb{K})})$. $\square$

We finally show that only one index is enough to support arbitrary moving average orders in k -order image matching.

Theorem 1. If an order k is an element of a set $\mathbb{K}$ (i.e., $k \in \mathbb{K}$), the transformation F satisfies the lower-bounding condition of Eq. (2), and X and Y are image time-series, then the lower-bounding condition of Eq. (5) holds.

$$D(X^{(k)}, Y^{(k)}) \geq D(X_F^{(k)}, \text{MBR}(Y_F^{(\mathbb{K})})) \quad (5)$$

In Eq. (5), $Y_F^{(\mathbb{K})}$ is a set of f -dimensional sequences that are transformed from n -dimensional sequences in $Y^{(\mathbb{K})}$ by F , and $\text{MBR}(Y_F^{(\mathbb{K})})$ is an f -dimensional MBR that bounds all sequences in $Y_F^{(\mathbb{K})}$.

Proof. We omit the proof since Eq. (5) of the theorem trivially holds by Eqs. (3) and (4). $\square$

Theorem 1 means that, if two image time-series X and Y are k -order similar, the f -dimensional point transformed from X is in ϵ -distance with the f -dimensional MBR $MBR(Y_F^{(\mathbb{K})})$ transformed from Y . In other words, Theorem 1 guarantees that the candidate set consisting of the time-series Y such that the query point $X_F^{(k)}$ is in ϵ -distance with the stored MBR $MBR(Y_F^{(\mathbb{K})})$ contains no false dismissal.

4.2 Index-Building and k -Order Image Matching Algorithms

Figure 4 shows the index-building algorithm. The inputs to the algorithm are an image time-series database and a set of moving average orders, and the output is a multidimensional index. In Lines (2) to (5), we store each data sequence in the index through the poly-order transform and the lower-dimensional transformation. In Line (2) we construct a set of n -dimensional sequences by performing the poly-order transform to the data sequence Y on the set of orders $\mathbb{K}$. In Line (3) we transform those n -dimensional sequences to f -dimensional sequences by the lower-dimensional transformation F . In Line (4) we construct an f -dimensional MBR by bounding those f -dimensional sequences. In Line (5) we finally store that MBR into the multidimensional index with Y 's identifier $Y-ID$. We construct the multidimensional index by repeating these Lines (2) to (5) for each data sequence (Lines (1) and (6)). In summary, we map each high-dimensional data sequence to a low-dimensional MBR and store that MBR into the index with its sequence identifier.

Procedure *BuildIndex*(Image time-series database DB , A set of orders $\mathbb{K}$)

- (1) **for** each data sequence Y in DB **do**
- (2) Make a set $Y^{(\mathbb{K})}$ of n -dimensional sequences by using the *poly-order moving average transform* on $\mathbb{K}$;
- (3) Construct a set $Y_F^{(\mathbb{K})}$ of f -dimensional sequences by using the transformation F ;
- (4) Construct an f -dimensional MBR $MBR(Y_F^{(\mathbb{K})})$ by bounding all f -dimensional sequences;
- (5) Make a record $\langle Y-ID, MBR(Y_F^{(\mathbb{K})}) \rangle$, and store it into the index;
- (6) **end for**

Fig. 4. The index-building algorithm

Figure 5 shows the k -order image matching algorithm. The inputs to the algorithm are a query sequence X converted from a query image, a given tolerance ϵ , and a moving average order k , and the outputs are the k -order similar data sequences. In the algorithm, we first construct a set of candidate sequences by searching the index, and then identify the true k -order similar sequences by accessing the image time-series database. In Line (1) we construct an n -dimensional sequence $X^{(k)}$ from the n -dimensional query sequence X through the k -order moving average transform. In Line (2) we obtain an f -dimensional sequence $X_F^{(k)}$ from $X^{(k)}$ through the lower-dimensional transformation F . After then, in Line (3) we construct an f -dimensional range query using $X_F^{(k)}$ and the

Procedure *k-OrderImageMatching*(Query sequence X , Tolerance ϵ , Order k)

- (1) Make an n -dimensional sequence $X^{(k)}$ from X by using the k -order moving average transform;
- (2) Transform $X^{(k)}$ to an f -dimensional sequence $X_f^{(k)}$ by using the transformation F ;
- (3) Make a range query using $X_f^{(k)}$ and ϵ ;
- (4) Construct a candidate set by evaluating the range query on the index;
- (5) Identify the true k -order similar images from the candidate set through the post-processing step;

Fig. 5. The k -order image matching algorithm

tolerance ϵ . In Line (4) we evaluate the range query on the multidimensional index and construct a set of candidate sequences that are potentially k -order similar to the query sequence. This candidate set contains false alarms as well as the true k -order similar sequences. Therefore, in Line (5) we finally perform the post-processing step that discards false alarms by retrieving the real data sequences from the database and by computing their k -order distance to the query sequence.

5 Experimental Evaluation

5.1 Experimental Data and Environment

We implemented a client-server system for k -order image matching. We also implemented three noise addition methods: a blur method, a random-noise method, and a mixed method. See Figure 6 for the screenshots explaining the blur method and the random-noise method. The mixed method makes a new image by exploiting these two methods one by one. We regarded these blur or random-noise effects as a kind of noise, and we used these noise images as data and query images in the experiments.

In the experiments, we constructed an image database consisting of total 90 thousand images. For this, we first collected total 10 thousand original images from the Web. Figure 7 shows examples of these original images. For the experiments, we generated eight more noise images from each original image. Figure 6 explains how we generate these eight noise images from an original image. As a result, we used total 90 thousand images consisting of 10 thousand original images and 80 thousand noise images. About 100 thousand time-series are generated from 90 thousand images, and those time-series are stored in the image time-series database. As shown in Figures 7(b) and 7(d), one image may contain two or more boundary objects, and in these cases two or more time-series may be extracted from one image.

The hardware platform of the server was a SUN Ultra 25 workstation equipped with an UltraSPARC IIIi CPU 1.34GHz, 1.0GB RAM, and a 80GB hard disk. The software platform was the Solaris 10 operation system. As a multidimensional index, we used the R*-tree and set its index and data page sizes to 4,096 bytes. We converted each boundary image to a time-series of length 360 and extracted eight DFT features from each time-series.

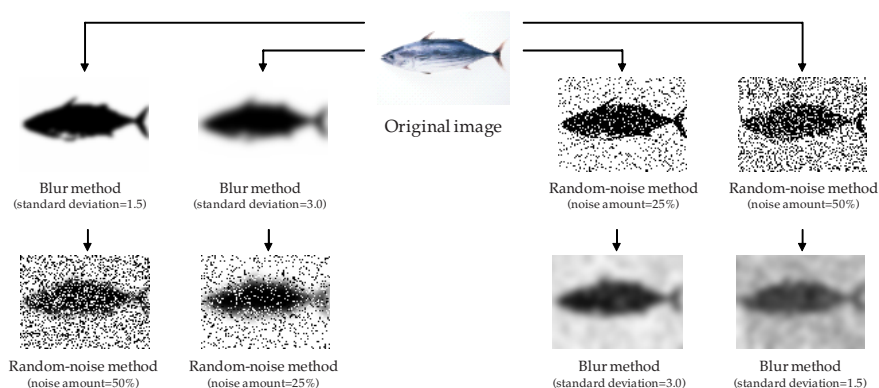


Fig. 6. Examples of eight different noise images generated from an original image



Fig. 7. Examples of original images

5.2 Experimental Results

We performed extensive experiments using a variety of images to confirm the effectiveness of k -order image matching. Figure 8 shows the experimental result when a pot image is used as a query. In the experiment, we set the query tolerance ϵ to the maximum among the tolerance values which returns two images as the 1-order image matching result. For example, in 1-order image matching of Figure 8, we get one or two images if $\epsilon \leq 13.5$ and three images if $\epsilon > 13.5$, so we set ϵ to 13.5. As shown in the figure, the larger moving average order causes the more number of images similar to the query image. In particular, as the order k increases, the noise pot images, which are generated from the same original pot image, are returned as similar ones. This means that our k -order image matching works well even for the large database having noise images.

Figure 9 summarizes the experimental results for other various images. In Figure 9, the *matching results* mean the returned images through k -order image matching, and the *similar results* the actual similar images that are generated from the same original image. As shown in the figure, the larger k increases the number of matching results in all cases, and all these matching results are identified as the similar results. As a result, we can say that our k -order image matching retrieves similar images relatively correctly, and the order k can be used as a measure of controlling the degree of noise reduction. However, too large k may return wrong images as similar ones. This is because non-similar images as well as similar ones can be smoothened by the moving average transform.

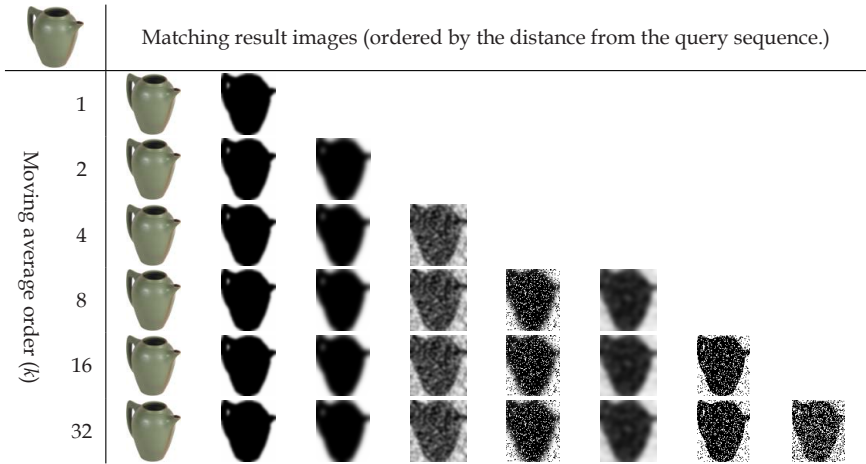


Fig. 8. The k -order image matching result of a pot image (tolerance=13.5)

Query image	Type	Noise method	The number of images	Moving average order (k)					
				1	2	4	8	16	32
	Cap	Not used	Matching results	2	3	4	5	6	7
			Similar results	2	3	4	5	6	7
	Rabbit	Not used	Matching results	2	5	7	7	8	8
			Similar results	2	5	7	7	8	8
	People	Blur method (Standard deviation=1.5)	Matching results	2	5	5	5	6	7
			Similar results	2	5	5	5	6	7
	Bulb	Blur method (Standard deviation=3.0)	Matching results	2	4	4	4	5	7
			Similar results	2	4	4	4	5	7
	Butterfly	Random-noise method (Noise amount=50%)	Matching results	2	2	3	4	5	6
			Similar results	2	2	3	4	5	6

Fig. 9. Summary of k -order image matching results for various query images

Therefore, we need to use k -order image matching in an interactive manner by controlling the order k , i.e., we need to investigate the more returned images step by step by increasing the order k .

We next compare the elapsed times of the proposed index-based method with those of a sequential scan. To explain the experimental results, we define *selectivity* [4, 6, 10] as the following Eq. (6), which means how many sequences are returned as the 1-order image matching result. In the experiments, we first determine a tolerance for each of selectivity $10\text{E-}4$, $10\text{E-}3$, and $10\text{E-}2$, and then measure the elapsed time by changing the order k for that determined selectivity.

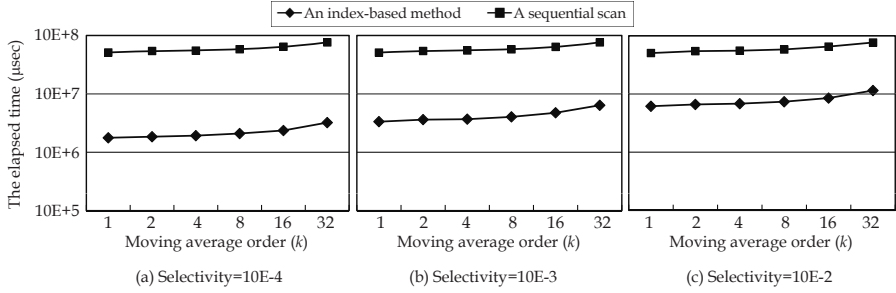


Fig. 10. The elapsed times for the original set

$$\text{Selectivity} = \frac{\text{Number of returned sequences by the 1-order image matching}}{\text{Number of all sequences in the image time-series database}} \quad (6)$$

As query images, we used four image sets: 1) an *original set* consists of 20 original images; 2) a *blur set* 20 noise images by the blur method with standard deviation of 1.5 and 3.0; 3) a *random-noise set* 20 noise images by the random-noise method with noise amount of 25% and 50%; and 4) a *mixed set* 20 images randomly selected from the image time-series database. We measured the elapsed times of 20 images and used their average as the experimental result.

Figure 10 shows the experimental results of the original set. As shown in the figure, our index-based method significantly reduces the elapsed time compared with the sequential scan. (Note that Y-axis is a log-scale.) Using a multidimensional index causes this improvement. In Figure 10, we note that, as the order k increases, the elapsed time slightly increases in both methods. This is because, as k increases, the number of candidates and their computation time for the moving average transform also increase. In summary, the index-based method improves the matching performance by 6.7 to 29.4 times over the sequential scan.

We next perform the experiments on the blur set, the random-noise set, and the mixed set. These experimental results show a very similar trend to those of the original set in Figure 10. That is, even though we use other image sets, the index-based method significantly outperforms the sequential scan, and this means that our index-based method shows better performance than the sequential scan regardless of image types.

6 Conclusions

In this paper we proposed a new approach in boundary image matching that achieved the noise reduction effect by exploiting the moving average transform. Contributions of the paper can be summarized as follows. First, considering the moving average transform we proposed new concepts of *k-order similar* and *k-order image matching*. The previous work did not consider the moving average transform, so we newly defined these concepts by extending the traditional definitions. Second, in Theorem 1 we formally proved that we could use a

multidimensional index in k -order image matching without incurring any false dismissal. Third, we presented the index-building and the image matching algorithms for k -order image matching. Fourth, through extensive experiments, we showed the effectiveness of k -order image matching and the superiority of the index-based method. Experimental results indicate that k -order image matching and its index-based solution provide a very practical way of noise-controlled image matching. In particular, to our best knowledge, this is the first attempt to solve some problems of the image domain through appropriate techniques of the time-series domain, and we believe that our approach can be widely used in removing other types of distortions in image matching areas.

Acknowledgments. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2007-331-D00381).

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search in Sequence Databases. In: Proc. the 4th Int'l Conf. on Foundations of Data Organization and Algorithms, pp. 69–84. Chicago, Illinois (October 1993)
2. Brailean, J.C., et al.: Noise Reduction Filters for Dynamic Image Sequences: A Review. In: Proceedings of the IEEE, vol. 83(9) (September 1995)
3. Do, M.N.: Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance. IEEE Trans. on Image Processing 11(2) (February 2002)
4. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast Subsequence Matching in Time-Series Databases. In: Proc. Int'l Conf. on Management of Data, pp. 419–429. ACM SIGMOD, Minneapolis, Minnesota (May 1994)
5. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd edn. Prentice Hall, Englewood Cliffs (2002)
6. Han, W.-S., Lee, J., Moon, Y.-S., Jiang, H.: Ranked Subsequence Matching in Time-Series Databases. In: Proc. the 33rd Int'l Conf. on Very Large Data Bases, Vienna, Austria, pp. 423–434 (September 2007)
7. Keogh, E., et al.: LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures. In: Proc. the 32nd Int'l Conf. on Very Large Data Bases, Seoul, Korea, September 2006, pp. 882–893 (2006)
8. Keogh, E., Ratanamahatana, C.A.: Indexing and Mining Large Time Series Databases. In: Proc. The 12th Int'l Conf. on Database Systems for Advanced Applications, Tutorial, Bangkok, Thailand (April 2007)
9. Lee, A.J.T., et al.: A Novel Filtration Method in Biological Sequence Databases. Pattern Recognition Letters 28(4), 447–458 (2007)
10. Moon, Y.-S., Whang, K.-Y., Han, W.-S.: General Match: A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows. In: Proc. Int'l Conf. on Management of Data, pp. 382–393. ACM SIGMOD, Madison, Wisconsin (June 2002)
11. Moon, Y.-S., Kim, J.: Efficient Moving Average Transform-Based Subsequence Matching Algorithms in Time-Series Databases. Information Sciences 177(23), 5415–5431 (2007)

12. Rafiei, D., Mendelzon, A.O.: Querying Time Series Data Based on Similarity. *IEEE Trans. on Knowledge and Data Engineering* 12(5), 675–693 (2000)
13. Theoharatos, C.: A Generic Scheme for Color Image Retrieval Based on the Multivariate Wald-Wolfowitz Test. *IEEE Trans. on Knowledge and Data Engineering* 17(6), 808–819 (2005)
14. Vlachos, M., Vagena, Z., Yu, P.S., Athitsos, V.: Rotation Invariant Indexing of Shapes and Line Drawings. In: *Proc. of ACM Conf. on Information and Knowledge Management*, Bremen, Germany, October 2005, pp. 131–138 (2005)
15. Zhu, Y., Shasha, D.: Warping Indexes with Envelope Transforms for Query by Humming. In: *Proc. Int'l Conf. on Management of Data*, June 2003, pp. 181–192. *ACM SIGMOD*, San Diego, California (2003)

Approximate Range-Sum Queries over Data Cubes Using Cosine Transform

Wen-Chi Hou¹, Cheng Luo², Zhewei Jiang¹, Feng Yan¹, and Qiang Zhu³

¹ Computer Science Department in Southern Illinois University Carbondale,
Carbondale, IL 62901, U.S.A

{hou, zjiang, fyan}@cs.siu.edu

² Department of Mathematics and Computer Science in Coppin State University,
2500 West North Avenue, Baltimore, MD, 21216, U.S.A

cluo@coppin.edu

³ Department of Computer and Information Science in University of Michigan-
Dearborn, Dearborn MI 48128, U.S.A

qzhu@umich.edu

Abstract. In this research, we propose to use the discrete cosine transform to approximate the cumulative distributions of data cube cells' values. The cosine transform is known to have a good energy compaction property and thus can approximate data distribution functions easily with small number of coefficients. The derived estimator is accurate and easy to update. We perform experiments to compare its performance with a well-known technique - the (Haar) wavelet. The experimental results show that the cosine transform performs much better than the wavelet in estimation accuracy, speed, space efficiency, and update easiness.

1 Introduction

Data warehouse is a large collection of integrated data, built to assist knowledge workers, such as executives, managers, analysts, etc., to make better and faster decisions. It is often required that the data be summarized at various levels of detail and on various combinations of attributes for on-line analytical processing (OLAP), which allows analysts to gain insight into the data through a variety of views. Typical OLAP applications include product performance and profitability, effectiveness of sales programs or marketing campaigns, sales forecasting, capacity planning, etc. Data warehousing and OLAP have increasingly become a focus of the database industry. OLAP systems generally support a multidimensional data model, known as a data cube [6]. A range-sum query, a very common and useful type of query over data cubes, is to compute the sum of measure attribute values of data cube cells that fall in the ranges specified by the query. It is very useful in finding trends and discovering relationships between attributes in OLAP. To facilitate range-sum query processing, prefix-sum cubes are proposed [3,4,5,7]. Although these methods generally can answer range-sum queries quickly, updates to data cubes can propagate to large portions of the prefix-sum cubes, incurring tremendous overheads. In addition, these approaches all require at least as much space as the original data cubes to store the prefix-sum cubes.

Approximate query answering present an appealing alternative to conventional query processing when exact answers are too slow or costly to derive. Fast approximate answers are very useful in exploratory data analyses, such as OLAP, decision support, and data mining. They provide quick summary information to users to help refine search in a potentially tedious mining process or in an ad-hoc drill-down and roll-up in OLAP [1]. In this research, we attempt to find a method that not only can provide fast approximate answers to range-sum queries, but also uses very little space. In addition, the approach is dynamically updatable in the presence of updates to data cubes.

In this research, we propose to use discrete cosine transform (DCT) to approximate prefix-sum cubes. The discrete cosine transform is known to have a good energy compaction property and thus can approximate the distribution of data cube cells' values easily using only a few low frequency terms. The derived estimator is accurate and easy to update. We perform experiments to compare its performance with a well-known technique, the (Haar) wavelet [11,15]. Experimental results show that DCT performs better than the wavelet in estimation accuracy, speed, space efficiency, and update easiness.

The paper is organized as follows. Section 2 reviews previous studies on data cube compression and prefix-sum cubes. Section 3 introduces the notations. Section 4 discusses approximation using cosine transform. Section 5 presents the range-sum query estimation. Section 6 discusses the updatability of the cosine estimator. Section 7 presents the experimental results. Section 8 concludes the paper.

2 Related Work

The condensed cube [16] condenses tuples from different cuboids that are aggregated from the same set of tuples into one tuple. The quotient cube method [8] partitions a cube into classes of cells with identical aggregate values to save storage space. Dwarf [14] accomplishes size reduction of data cubes by factoring redundant prefixes and suffixes out of the data warehouse. Unfortunately, the constructions of such cubes are generally complex and the effectiveness of these reduction methods heavily depend upon the properties of the data themselves. For range-sum queries, substantial portions of the size-reduced cubes may have to be accessed. Li et al. [10] indicated that poor query performance was observed in condensed and quotient cubes.

Quasi-cubes [2] slice the data cubes and approximate the distributions of subcubes by linear functions. It is not clear how range-sum queries can be answered when subcubes partially intersect with the ranges of queries. In addition, updates are difficult to handle on the fly and periodical reconstructions of the linear functions may be required.

To facilitate range-sum query processing, Ho et al. [7] computed the prefix sums of data cubes. Although this method can answer queries fast, an update in the worst case can propagate to the entire prefix-sum cube, which is as large as the original data cube. To control the cascading of updates, Geffner et al.

[4,5] decomposed the prefix-sum cubes recursively and Chan et al. [3] organized the prefix-sum cubes hierarchically; but the complexity of update still increases exponentially with the number of dimensions. In general, update propagation is a common problem for all these prefix-sum data cube approaches. Note that all these approaches require tremendous amounts of space, at least as large as the sizes of the original data cubes, to store the prefix-sum cubes.

The wavelet transforms [13] decompose the original signal by applying high-pass and low-pass filters repeatedly until a predefined decomposition level is reached. The Haar transform is conceptually simple and fast. It is proved that the largest coefficients in absolute value carry the most important information of the original signal in the Haar transform. Thus, the original signal can be compressed using a small number of coefficients that have largest absolute values. The wavelet transform has been used to compress histograms for selectivity estimation [11]. Vitter et al. [15] compressed the data cubes using the Haar wavelet and showed that estimates of aggregation queries can be derived quickly and accurately. Matias et al [12] have enhanced the method with a dynamic update scheme for its coefficients.

In this study, we use the discrete cosine transform (DCT) to approximate the cumulative distribution of the measure attribute values in the data cubes. DCT is known to have a good energy compaction property and thus can approximate a distribution easily using a few (low-frequency) coefficients. DCT also has a simple and efficient update method. Since the domains of dimension attributes are all discrete or already discretized, we shall use “discrete cosine transform and “cosine transform” interchangeably in the paper for simplicity.

The wavelet method [15] is most relevant to our approach as both attempt to apply mathematical techniques to compress the data cubes. We shall have more in depth comparisons of the two approaches in subsequent sections.

3 Notations

3.1 Attribute Value Normalization

Consider a data cube with d dimension attributes $X_1, \dots, X_d$. By converting a categorical domain to numerical, all dimension attributes can be viewed as numerical. Let M be a measure attribute whose values are of interest. We assume M has the set of real numbers R as its domain.

To simplify notations and algorithm implementation, dimension attribute values are normalized to a predetermined domain $[0, 1]$. Let $\max X_i$ and $\min X_i$ be the maximal and minimal values of attribute X_i , respectively. Then, each value x_i of X_i can be normalized as follows:

$$x_i^z = \frac{x_i - \min X_i}{\max X_i - \min X_i}, \quad \min X_i \leq x_i \leq \max X_i \quad (1)$$

From now on, we shall assume all attribute values are so normalized and shall not distinguish x_i from x_i^z , unless otherwise stated.

3.2 Range-Sum Queries

Let $X = (X_1, \dots, X_d)$ be the set of dimension attributes. For each $1 \leq i \leq d$, denoted by x_i is a value of the attribute X_i . Given $x = (x_1, \dots, x_d) \in [0, 1]^d$, and $y = (y_1, \dots, y_d) \in [0, 1]^d$, we say $x \leq y$ if $x_i \leq y_i$ for all $i=1, \dots, d$.

We assume all range constraints in a range-sum query have the form $a_i < X_i \leq b_i$. Other forms of range constraints, such as $a_i \leq X_i \leq b_i$, $a_i \leq X_i < b_i$, $a_i < X_i < b_i$ can all be converted to this form. For example, $a_i \leq X_i \leq b_i$ can be rewritten as $a_i^- < X_i \leq b_i$ where a_i^- is largest X_i value that is smaller than a_i . Let $\{X_{i1}, \dots, X_{ik}\}$ be the set of attributes on which range constraints, such as $a_i < X_i \leq b_i$, are posted in the queries. By denoting $a_i = 0$ and $b_i = 1$ for $i \notin \{i_1, \dots, i_k\}$, a range-sum query $Q(a, b)$, where $a = (a_1, \dots, a_d) \in [0, 1]^d$, $b = (b_1, \dots, b_d) \in [0, 1]^d$, and $a_i \leq b_i$ for all $i = 1, \dots, d$, is to compute the sum of measure attribute values for those cells whose dimension attribute values satisfy $a_i < X_i \leq b_i$ for all $1 \leq i \leq d$.

3.3 Empiric Distribution

Consider a random variable M that has a cumulative distribution function F . If F is known, the probability of M falling in the range $(a, b]$ is

$$P\{a < X \leq b\} = F(b) - F(a) \quad (2)$$

For simplicity, we shall use the term “distribution” for “cumulative distribution” from now on. The empiric (cumulative) distribution represents exactly the data distribution of the data cube without losing any information. Consider a data cube as a sample from the measure attribute domain R . Each cell of the data cube can be viewed as an observation. Given a set of n non-zero (measure attribute) observations $\{v_1, \dots, v_n\}$, whose coordinates in the data cube are $\{y_1, y_2, \dots, y_n\}$, the empiric distribution function $\hat{F}(x)$ is defined by

$$\hat{F}(x) = \sum_k v_k, \quad \text{whose } y_k \leq x, \quad x \in [0, 1]^d \quad (3)$$

Then given a sample of a random variable M , we can use the empirical distribution constructed from the sample to estimate the distribution of M .

Consider a one-dimensional example here. Given a sample of 6 measure attribute values $\{3, 1, 4, 1, 2, 2\}$ at coordinates $\{0.2, 0.3, 0.4, 0.5, 0.6, 0.8\}$, we can estimate the probability $P\{X \leq 0.4\}$ as the ratio of the sum of the values whose coordinates are less than or equal to 0.4 (i.e., 8) to the sum of all the values (i.e., 13), that is $8/13$.

In this paper, our goal is to find an approximation, denoted by $\hat{F}_m(X)$, to the empiric distribution $\hat{F}(X)$. The approximation shall use much less space and yet with little information loss.

4 Empirical Distribution Estimation Via Cosine Series

In this section, we discuss how to derive an estimator for the empiric distribution.

Let the cosine series be denoted as $\phi_i(x)$, $i \geq 0$,

$$\phi_i(x) = \begin{cases} 1, & i = 0; \\ \sqrt{2} \cos i\pi x, & i > 0; \end{cases}$$

That is, $\{1, \sqrt{2}\cos\pi x, \sqrt{2}\cos 2\pi x, \dots, \sqrt{2}\cos i\pi x, \dots\}$. Let $\Phi_i(x) = \int_x^1 \phi_i(u)du$, That is,

$$\Phi_i(x) = \begin{cases} 1 - x, & i = 0; \\ -\sqrt{2} \sin i\pi x / i\pi, & i > 0; \end{cases}$$

We assume for simplicity all dimension attributes have the same domain size D . Consider a d -dimensional data cube with n non-zero measure attribute values (or cell values) $v_1, v_2, \dots, v_n$, whose coordinates are $y_1, y_2, \dots, y_n$, respectively. Then, the empirical distribution $\hat{F}(x_1, \dots, x_d)$ of the cells' values can be represented by a cosine series with D^d coefficients, $\hat{\beta}_{i_1, \dots, i_d}$ $0 \leq i_1, \dots, i_d \leq D - 1$, as

$$\hat{F}(x_1, \dots, x_d) = \sum_{i_1=0}^{D-1} \dots \sum_{i_d=0}^{D-1} \hat{\beta}_{i_1, \dots, i_d} \prod_{j=1}^d \phi_{i_j}(x_j) \quad (4)$$

where $\hat{\beta}_{i_1, \dots, i_d}$'s are

$$\hat{\beta}_{i_1, \dots, i_d} = \sum_{k=1}^n v_k \left(\prod_{j=1}^d \Phi_{i_j}(y_{kj}) \right) \quad (5)$$

where y_{kj} is the j^{th} -dimension coordinate of $y_k, 1 \leq k \leq n$.

While the empirical function $\hat{F}(x_1, \dots, x_d)$ describes the exact distribution of tuples, the storage of such a function could be large, especially when the number of domain attributes d and the domain sizes thereof are large. To save storage space, we opt to approximate the function by a smaller number of cosine coefficients.

The cosine transform has a good energy compaction property; most energy is preserved in the first few low frequency terms. Thus, one can approximate, without much information loss, the empirical distribution $\hat{F}(x_1, \dots, x_d)$ with its first m^d (low frequency) terms, denoted $\hat{F}_m$, as

$$\hat{F}(x_1, \dots, x_d) \approx \hat{F}_m(x_1, \dots, x_d) = \sum_{i_1=0}^{m-1} \dots \sum_{i_d=0}^{m-1} \hat{\beta}_{i_1, \dots, i_d} \prod_{j=1}^d \phi_{i_j}(x_j) \quad (6)$$

In general, the larger the m value, the better the approximation. It is noted that only the m^d coefficients (real numbers) need to be stored for the approximate frequency function. In contrast, besides the coefficients, the wavelet needs to store the indexes of coefficients too (to indicate which terms are selected). Thus, the cosine method is more space-efficient than the wavelet method.

Example 4.1: Consider a one-dimensional data cube with 6 non-zero measure attribute values v 's: $\{2, 1, 5, 4, 2, 1\}$ and their respective coordinates y 's: $\{0.12,$

0.32, 0.33, 0.66, 0.80, 0.90}. The cosine transform of this distribution is derived as follows.

$$\hat{\beta}_0 = \sum_{j=1}^6 v_j \Phi_0(y_j) = \sum_{j=1}^6 v_j (1 - y_j) = 7.65$$

$$\hat{\beta}_1 = \sum_{j=1}^6 v_j \Phi_1(y_j) = \sum_{j=1}^6 v_j \left[-\frac{\sqrt{2}}{\pi} \sin \pi y_j \right] = -4.8951$$

The estimator of the empiric distribution with 2 coefficients is $\hat{F}_2(x) = 7.65 - 4.8951(\sqrt{2} \cos \pi x)$.

5 Estimation of Range-Sum Queries

In this section, we elaborate on the estimation of range-sum queries using the approximate empiric distributions derived in the previous section.

Let us illustrate our idea through a 1-dimensional case. Suppose X is a random variable such that $X \in [0, 1]$ with a cumulative distribution function $F(x)$. The

Range-sum($m, a, b, \beta[]$, T)

Input: an integer $m > 0$, two vectors $a = (a_1, \dots, a_d)$, $b = (b_1, \dots, b_d)$,
 $l \leq a_i \leq b_i \leq r$, and m^d coefficients $\{\beta[0, \dots, 0], \dots, \beta[m-1, \dots, m-1]\}$
of the orthogonal series estimators $\hat{F}_m(x)$.

Output: an estimation of $Q(a, b)$.

begin

 prob $\leftarrow 0$;

$k \leftarrow 1$;

for $i \leftarrow 0$ **to** $2^d - 1$ **do**

for $j \leftarrow 0$ **to** $d - 1$ **do**

if $i \bmod 2^j = 0$ **then**

$p[j] = a[j]$;

$k \leftarrow (-k)$;

end

else

$p[j] = b[j]$;

end

end

 prob $\leftarrow$ prob $+ k \times$ Empiric-Distribution($m, p, \beta[]$);

end

if prob > 0 **then**

return prob;

end

else

return 0;

end

end

Fig. 1. Estimate a Range-sum Query

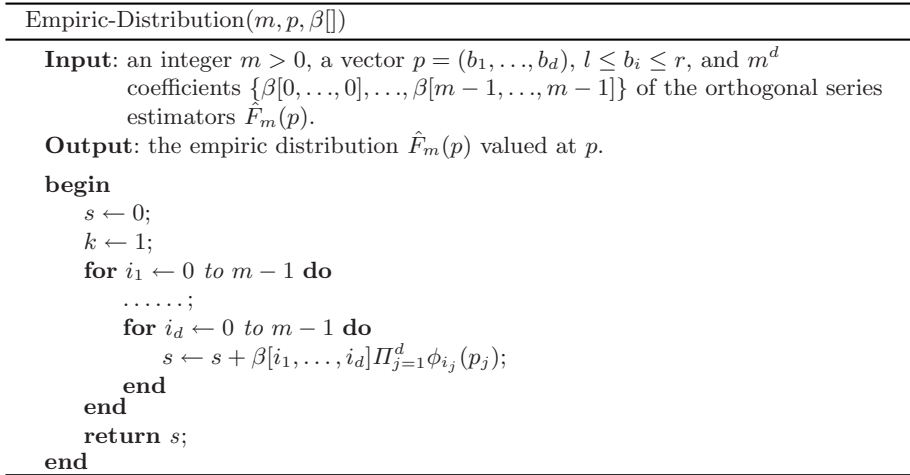


Fig. 2. Compute the Empiric Distribution $\hat{F}_m(p)$ at p

probability that $a < X \leq b$ is

$$P\{a < X \leq b\} = F(b) - F(a) \approx \hat{F}_m(b) - \hat{F}_m(a) \quad (7)$$

Let us extend X to a d -variate random vector. Let $a = (a_1, \dots, a_d) \in [0, 1]^d$, $b = (b_1, \dots, b_d) \in [0, \dots, 1]^d$, and $a < b$. A vertex of the hyperinterval $(a, b]$

$$(a, b] = \{x = (x_1, \dots, x_d) \in [0, 1]^d : a_1 < x_1 \leq b_1, \dots, a_d < x_d \leq b_d\} \quad (8)$$

is denoted as $u = (u_1, \dots, u_d) \in [0, 1]^d$ with $u_i \in \{a_i, b_i\}$ for $i = 1, \dots, d$. Let $\Delta_k(a, b)$ be the set of all vertices u with $u_i = a_i$ for exactly k coordinates and $u_j = b_j$ for the remaining coordinates. Then,

$$P\{a_i < X_i \leq b_i, \quad \forall 1 \leq i \leq d\} = \sum_{k=0}^d (-1)^k \sum_{u \in \Delta_k(a, b)} F(u) \quad (9)$$

Thus, the range-sum query $Q(a, b)$ is estimated as

$$Q(a, b) \approx \sum_{k=0}^d (-1)^k \sum_{u \in \Delta_k(a, b)} F(u). \quad (10)$$

Example 5.1. Suppose $d = 2$, $a = (a_1, a_2) \in [0, 1]^d$, $b = (b_1, b_2) \in [0, 1]^d$, $a_1 \leq b_1$, $a_2 \leq b_2$. Then, by Eq. (9)

$$P\{a_i < X_i \leq b_i, \quad \forall 1 \leq i \leq 2\} = F(b_1, b_2) + F(a_1, a_2) - F(a_1, b_2) - F(b_1, a_2).$$

Figure 1 summarizes the computation of a range-sum query $Q(a, b)$.

The above algorithm calls Empiric-Distribution, depicted in Figure 2, to compute the empirical distribution of the measure attribute values $\hat{F}_m(p)$ (i.e., Eq.(6) at a specific point p .

5.1 Storage of the Estimator

By utilizing the good energy compaction property, one can further filter out high frequency terms without much information loss. A technique, called Triangle Sampling [9] can be applied. It stores only those coefficients whose indexes satisfy $i_1 + \dots + i_d \leq m - 1$. Thus, the number of the coefficients finally stored is $C(m+d-1, d) \approx m^d/d!$, which is much smaller than m^d . Note that the indexes $(i_1, \dots, i_d)$ of the coefficients need not be stored because they are unique and can be derived on the fly.

For example, consider a 2-dimensional case ($d = 2$) and m has been set to 3. Then, there will be $m^d (=3^2)$ coefficients in the approximation function, denoted as $C_{i,j}$, $0 \leq i, j \leq m - 1$. By triangle sampling, only 6 ($=C(m+d-1, d)$) of them that satisfy the condition: $i_1 + i_2 \leq m - 1 = 2$, are kept. They are: $C_{0,0}$, $C_{0,1}$, $C_{0,2}$, $C_{1,0}$, $C_{2,0}$, $C_{1,1}$. We will incorporate this technique in our implementation.

6 Dynamic Maintenance of the Estimator

As observed from Eq. (5), each coefficient $\hat{\beta}_{i_1, \dots, i_d}$ of the transform is basically the sum of the product of the measure attribute values and the products of basis functions on the measure attribute value's coordinates. Therefore, for insertion or deletion of a measure attribute value, we can just compute the "contribution" of the value to the transform and then combine them with the old coefficients. That is, for insertion of a new value t at $y = (y_1, y_2, \dots, y_d)$, $\hat{\beta}_{i_1, \dots, i_d}$ is updated as

$$\hat{\beta}_{i_1, \dots, i_d} = \hat{\beta}_{i_1, \dots, i_{d-1}} + t \prod_{j=1}^d \Phi_{i_j}(y_j) \quad (11)$$

Similarly, for deletion of a value t at $y = (y_1, y_2, \dots, y_d)$, it is updated as

$$\hat{\beta}_{i_1, \dots, i_d} = \hat{\beta}_{i_1, \dots, i_{d-1}} - t \prod_{j=1}^d \Phi_{i_j}(y_j) \quad (12)$$

An update to the measure attribute value can be accomplished by a deletion followed by an insertion. Let m be the number of coefficients of the estimator. The complexities of an insertion, a deletion, and an update are all $O(m)$.

Coefficients can be updated easily and dynamically. This property makes the cosine transform well suited in data stream environments, where tuples continuously flow in. The updates of the coefficients can be performed on the fly as well as in batch. In addition, the computation workload can be easily distributed among processors as the "contributions" of tuples can be computed separately.

7 Experimental Results

In this section, we report experimental results of estimating range-sum queries using the cosine and wavelet methods.

7.1 Experiment Setup

We have implemented both methods in C++ and compiled them with GNU C/C++ Compiler V3.2.3. The test platform is Redhat Linux Enterprise 4 running on a Dell Precision 360 workstation with 3.3 GHZ CPU and 1GB RAM.

Experiments are run on both synthetic and real-life datasets. The purpose of using synthetic data is to study the methods in relation to different characteristics of data in a controlled environment. The synthetic relations are generated following the TPC-D benchmark [18] with attribute values distributed Zipfianly [17]. We generate distributions with two different z values, 0.5, and 1.0, which represent, roughly speaking, a slightly skewed and skewed distribution, respectively. The domain size of each dimension attribute is 1,024 and the sum of measure attribute values is 10^6 .

We have also used a real-life dataset from the Bureau of Census [19]. We select the data for a period of three-months, from January to March 2004. The dataset has around 140,000 tuples for each month. The dimension attributes are Age, Education and State, whose ranges are [1, 99], [1, 46] and [1, 99], respectively, and the measure attribute is count (or the number of tuples).

7.2 Estimation Accuracy

We ran 100 queries with randomly chosen ranges on dimension attributes. The accuracy of estimation is measured by average relative errors.

Performance on Synthetic Datasets. Figures 3 and 4 show the estimation results of range-sum queries over two-dimensional data cube with Zipf parameters, 0.5 and 1.0, respectively. As shown in the figures, in general, the greater the number of coefficients used, the better the results for both techniques.

As shown in Figure 3, the cosine method performed better than the wavelet using the same number of coefficients for the slightly skewed distribution. The wavelet generated average errors ranging from 3.93% for 100 coefficients to 1.07% for 2,000 coefficients, while ours from 1.02% to 0.33% for the same number of coefficients.

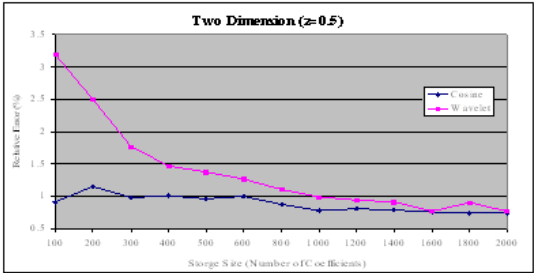


Fig. 3. Two-dimensional Queries, $z=0.5$

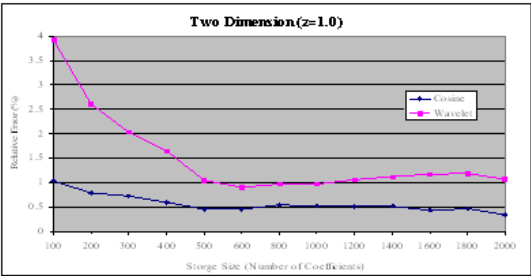


Fig. 4. Two-dimensional Queries, $z=1.0$

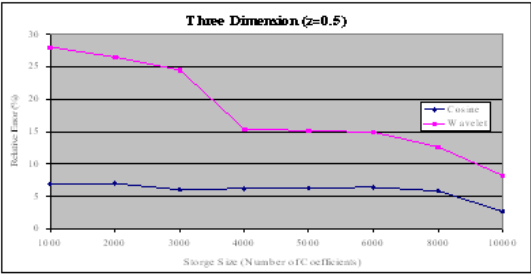


Fig. 5. Three-dimensional Queries, $z=0.5$

Notice that with only 100 coefficients, our estimates are already very accurate (around 1% error) in both cases. But the accuracy does not improve much when the number of coefficients increases further. This demonstrates the good energy compaction property of the cosine transform.

As mentioned earlier, the cosine transform stores only the coefficients, but the wavelet needs to store the indexes with the coefficients. Thus, for the same number of coefficients, the wavelet uses at least twice as much space as ours. This further demonstrates the efficiency of the cosine transform.

As the distributions become more skewed (i.e., $z = 1.0$), it becomes more difficult to capture the sharp changes in frequency and thus estimation accuracy degrades. Nevertheless, our method demonstrates an even larger performance edge over wavelet in the more skewed case ($z = 1.0$) than in the smoother case ($z = 0.5$).

Figures 5 and 6 show the results of range-sum queries with constraints on three-dimensional data cubes with Zipf parameters 0.5 and 1.0, respectively. In general, the higher the dimension, the greater the number of coefficients is needed to achieve a desired accuracy. This is mainly due to the increased number of frequency values to be approximated (or compressed) in higher dimensional spaces. Again, our method performed better than the wavelet method for the same number of coefficients. In Figure 5, wavelet generated average errors ranging from 28.04% for 1,000 coefficients to 8.25% for 10,000 coefficients, while

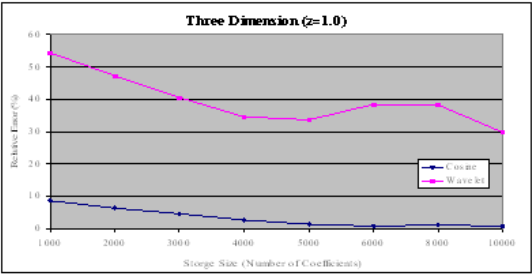


Fig. 6. Three-dimensional Queries, $z=1.0$

ours from 6.9% to 2.67% for the same number of coefficients. Wavelet’s errors are about 4 times larger than ours.

As distributions become more skewed, the errors become greater like in the 2-dimensional cases. For example, at $z = 1.0$, the wavelet generated average errors ranging from 54.45% for 1,000 coefficients to 29.88% for 10,000 coefficients while our approach generated from 8.68% to 0.81% for the same number of coefficients. The wavelet’s errors are about 6 to 37 times larger than ours. In Figures 7 and 8, we show the results of six-dimensional queries. With each dimension partitioned into 16 regions, it results in a $16^6 (= 16 \text{ million})$ -bucket histogram. The wavelet generated large errors (e.g., $> 100\%$) for small numbers of coefficients (e.g., 1,000, 2,000, etc). Even for the largest number of coefficients we tested, i.e., 8,000 coefficients, the errors are still very large, for example, 49.5% for $z = 0.5$ and 59.3% for $z = 1.0$. Due to the large errors of wavelet, we present only the results of cosine series in the following.

As a short summary, the cosine transform performs much better than the wavelet in accuracy. Nevertheless, all these two methods use much less space than a prefix-sum cube method, which requires at least as large space as the original data cube (1024^d cells).

A Real Dataset. Figures 9 and 10 show the results on the real dataset.

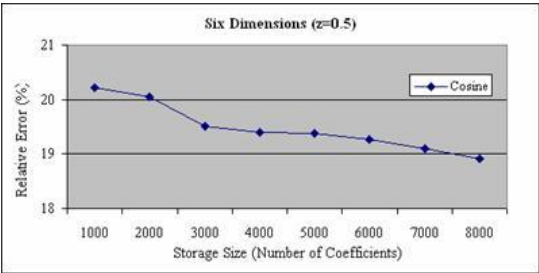


Fig. 7. Six-Dimensional Queries, $z=0.5$

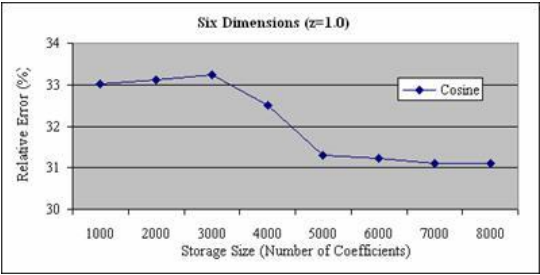


Fig. 8. Six-Dimensional Queries, $z=1.0$

As in the synthetic experiments, our method performs better than the wavelet. For example, with only 100 coefficients, as shown in Figure 9, the errors of the wavelet and cosine methods are 12.45% and 1.68%, respectively.

For three-dimensional queries, as shown in Figure 10, with 100 coefficients, our error is already below 10% while the wavelet still has an error as high as 54%.

7.3 Update and Estimation Speeds

As mentioned earlier, the higher the number of dimensions, the greater the number of coefficients is required for an estimator to achieve an acceptable accuracy. We assume that the estimators have 400, 3,000, 8,000 coefficients for 2-dimensional 3-dimensional, and 6-dimensional cases, respectively, which we believe are generally large enough to yield reasonable accuracy.

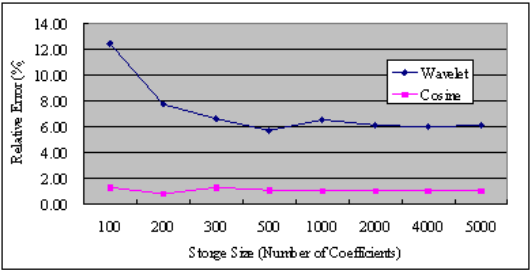


Fig. 9. Real Dataset: Two-dimension Queries

Table 1. Update Speed

Time (μs)	2-dimension	3-dimension	6-dimension
Wavelet	4.7	6,906	—
Cosine	132	1,250	3,002

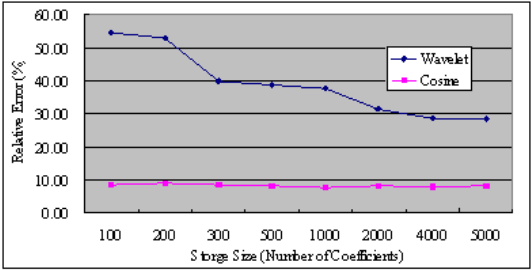


Fig. 10. Real Dataset: Three-dimension Queries

Table 2. Estimation Speed

Time (μs)	2-dimension	3-dimension	6-dimension
Wavelet	210	2,058	—
Cosine	72	389	1,321

Let m be the number of coefficients used in the estimators. It takes $O(m)$ time to update a cosine estimator, as shown in Eqs. (11) and (12). On the other hand, wavelet takes $O(\log H)$ time to update the coefficients, where H is the size of the underlying histogram (or the number of cells in the data cube), recalling that wavelet is a histogram-based method. Note that the size of histogram generally increases exponentially with the number of dimensions (d) of the histogram, i.e., $H = |D|^d$, where $|D|$ is the size of each dimension attribute domain (assumed to be the same for all dimension attributes). Consequently, Table 1 shows wavelet is much slower in high dimensional cases.

The cosine method has a complexity of $O(2^d m)$ for a range-sum query estimation, as demonstrated in Eq. (9). The wavelet has a complexity of $O(2^d H \log(H))$. Hence, the wavelet estimator can be very slow in high dimensions, as shown in Table 2.

8 Conclusions

In this paper, we develop a nonparametric statistical range-sum query estimation approach, which is based upon the empiric distribution estimation by the cosine series. First, we derive an estimator for the empiric distribution of measure attribute values in a data cube, and then use the empiric distribution estimator to compute the range-sum query estimates. The empiric distribution estimator can be stored easily and updated efficiently. The experimental results have shown that our approach produced much more accurate estimates than the wavelet method. The proposed method is well suited for on-line approximate aggregate query estimation over data cubes. It is simple, accurate, efficient, and adaptive.

References

1. Acharya, W., Gibbons, P., Poosala, V.: Aqua: A Fast Decision Support System Using Approximate Query Answers. In: Proc. 25th VLDB Conference (1999)
2. Barbara, D., Sullivan, M.: Quasi-cubes: Exploiting approximation in multi-dimensional databases. SIGMOD Record 26, 12–17 (1997)
3. Chan, C., Ioannidis, Y.: Hierarchical cubes for range-sum queries. In: Proc. VLDB, pp. 675–686 (1999)
4. Geffner, S., Agrawal, D., Abbadi, A., Smith, T.: Relative prefix sums: an efficient approach for querying dynamic OLAP Data Cubes. In: Proc. ICDE, pp. 328–335 (1999)
5. Geffner, S., Agrawal, D., Abbadi, A.: The Dynamic Data Cube. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 237–253. Springer, Heidelberg (2000)
6. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In: Proc. ICDE Conference (1996)
7. Ho, C., Agrawal, R., Megiddo, N., Srikant, R.: Range queries in OLAP data cubes. In: Proc. ACM SIGMOD Conference, pp. 73–88 (1997)
8. Lakshmanan, L., Pei, J., Han, J.: Quotient cube: How to summarize the semantics of a data cube. In: Proc. 28th VLDB Conference, pp. 528–539 (2002)
9. Lee, J., Kim, D., Chung, C.: Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. ACM SIGMOD, 205–214 (1999)
10. Li, S., Wang, S.: Semi-closed cube: An effective approach to trading off data cube size and query response time. Journal of Computer Science and Technology 20(3), 367–372
11. Matias, Y., Vitter, J., Wang, M.: Wavelet-based histograms for selectivity estimation. In: ACM SIGMOD Conference, pp. 448–459 (1998)
12. Matias, Y., Vitter, J., Wang, M.: Dynamic Maintenance of Wavelet-Based Histograms. In: Proc 26th VLDB Conference, pp. 101–110 (2000)
13. Nievergelt, Y.: Wavelets Made Easy. Birkhauser, Basel (1999)
14. Sismanis, Y., Roussopoulos, N., Deligiannakis, A., Kotidis, Y.: Dwarf: Shrinking the petacube. In: Proc. ACM SIGMOD Conference, pp. 464–475 (2002)
15. Vitter, J., Wang, M., Lyer, B.: Data cube approximation and histograms via wavelets. In: Proc. CIKM, pp. 96–104 (1998)
16. Wang, W., Feng, J.L.: Condensed cube: An effective approach to reducing data cube size. In: Proceedings of the 18th International Conference on Data Engineering (2002)
17. Zipf, G.: Human behavior and the principle of least effort. Addison-Wesley, Reading (1949)
18. TPC benchmark D, decision support (1995)
19. http://www.bls.census.gov/sipp/_ftp.html#sipp04

Querying Multigranular Spatio-temporal Objects

E. Camossi¹, M. Bertolotto¹, and E. Bertino²

¹ University College Dublin, Ireland*

Tel.: +353 (0)1 7162-913; Fax: +353 (0)1 2697-262

{elena.camossi,michela.bertolotto}@ucd.ie

² Purdue University, Indiana, USA**

Tel.: +1 765 496-2399; Fax: +1 765 494-0739

bertino@cs.purdue.edu

Abstract. The integrated management of both spatial and temporal components of information is crucial in order to extract significant knowledge from datasets concerning phenomena of interest to a large variety of applications. Moreover, multigranularity, i.e., the capability of representing information at different levels of detail, enhances the data modelling flexibility and improves the analysis of information, enabling to zoom-in and zoom-out spatio-temporal datasets. Relying on an existing multigranular spatio-temporal extension of the ODMG data model, in this paper we describe the design of a multigranular spatio-temporal query language. We extend OQL value comparison and object navigation in order to access spatio-temporal objects with attribute values defined at different levels of detail.

Keywords: Spatio-temporal query language, Spatial and temporal granularities.

1 Introduction

As the available datasets are becoming increasingly large and often unnecessarily detailed due to the development of sophisticated collection technologies, effective methods for presenting information to users are required. In such respect approaches able to present the data at different levels of detail (i.e., *granularities*) represent an effective solution to facilitate the analysis when additional details are only required for specific subsets of the data. For example, zoom-out operations can improve the efficiency of spatio-temporal data mining algorithms, which are time consuming [1]. On the other hand, zoom-in operations can help in refining the mining of specific data subsets. Multigranularity, multiresolution and multiple representation have been investigated first for temporal data [5,6], and more recently for both spatial [2,22] and spatio-temporal data [7,12,9,20].

* School of Computer Science and Informatics - University College Dublin, Belfield, Dublin 4, Ireland.

** CERIAS - Purdue University, 250 N. University Street West Lafayette, Indiana, USA 47907-2066.

In particular, the *ST*_ODMG data model [9] has been defined as extension of ODMG [10], the reference model for object-oriented databases. *ST*_ODMG models multigranular spatio-temporal values, relying on the definition of temporal granularity proposed by Bettini et al. [6], which is commonly adopted by the temporal databases and reasoning community. The model also relies on a notion of spatial granularity compliant with the formalization of stratified map spaces proposed by Stell and Worboys [26]. Moreover, unlike other multigranular models, it incorporates a framework for the conversion of spatio-temporal values at different spatial and temporal granularities.

Since the effectiveness of a model greatly depends on the associated query language, several spatio-temporal query languages have been developed [13,15,17,19]. Among these approaches, in [13,15] SQL:99 has been extended to spatio-temporal support. The extension of SQL proposed by Chen and Zaniolo [13] is based on user-defined spatio-temporal aggregates and functions in order to minimize changes to SQL. Erwig et al. [15] propose an extension of SQL to include spatio-temporal objects defined in terms of abstract data types relying on the abstract framework defined in [18] for moving objects.

By contrast, the approaches by Huang and Claramunt [19] and by Griffiths et al. [17] define how to query spatio-temporal values in an object oriented paradigm. Huang and Claramunt propose an OQL spatio-temporal extension that includes spatio-temporal operators for evaluating spatial queries and topological relationships; Griffiths et al. propose supporting queries against spatio-temporal objects at application level.

This paper focuses on the access of multigranular spatio-temporal data, which has not been discussed in the previous proposals. Specifically, we investigate the impact of multigranularity on the specification and execution of spatio-temporal queries. Spatio-temporal multigranularity may not be simply supported relying only on data types and operators already available in object oriented and relational query languages: a specific query language must be designed in order to support accesses to subsets of data that refer to spatio-temporal *granules* and sets of granules, both explicitly and implicitly represented through constraints against database values. Thus, simple expressions for representing temporal and spatial granules at different granularities must be provided. Furthermore, expressions in a multigranular query language must support multigranular spatio-temporal comparison of attribute values. Moreover, since an attribute can be accessed with granules at a different granularity levels, a suitable set of value conversions to convert spatio-temporal data at different granularities has to be integrated in the query language. Such conversions should support attribute values conversions according to the semantics of the attribute involved in the query. Therefore the types of conversion should vary according to the data types and the semantics of the represented information.

In this paper we propose a multigranular spatio-temporal query language which provides specific solutions to these requirements; its design relies on the multigranular model defined in [9]. The overall conceptual design of a multigranular spatio-temporal model and query language addresses some important issues

related to spatio-temporal multigranularity. Taking advantage of the extension capability given by the object oriented paradigm, this design can be applied to extend both object oriented and relational database models to support spatio-temporal multigranularity.

The language herein presented extends OQL, the set-oriented content-based query language associated with the ODMG data model, with features supporting queries against non-homogeneous multigranular spatio-temporal objects. The two mechanisms supported by SQL for querying data, namely value comparison and object navigation, have been extended with multigranular spatio-temporal capabilities: we have extended the conventional path expressions in order to query multigranular spatio-temporal objects, whose attributes can be accessed according to given spatial and temporal elements [16], expressed at multiple granularities.

Whenever attribute values at different granularities have to be accessed and compared in a multigranular path expression, multigranular conversions [9] are applied to obtain values expressed at a common granularity. Throughout the paper, the formalization we propose is described in details and its application is illustrated through significative examples. Moreover, we specify the consistency conditions for accessing multigranular spatio-temporal values.

The paper is organized as follows. In Section 2 we briefly describe the main characteristics of the *ST-ODMG* model. In Section 3 we specify how multigranular spatio-temporal values are accessed, while in Section 4 we give some illustrative examples of multigranular spatio-temporal queries. The complete syntax of *ST-ODMG* queries is reported in the Appendix. Finally, Section 5 concludes the paper outlining future research directions.

2 *ST-ODMG*: A Multigranular Spatio-temporal Model

Granularities in *ST-ODMG* are specified as mappings from an index set *IS* to the power set of the *TIME* and the *SPACE* domains, respectively. *TIME* is totally ordered. The supported *SPACE* domain is 2-dimensional (i.e., a proper subset of R^2). For instance, *days*, *weeks*, *years* are temporal granularities; *meters*, *kilometers*, *feet*, *yards*, *provinces* and *countries* are spatial granularities.

Each subset of the temporal and spatial domains corresponding to a single granularity mapping is referred to as a temporal or spatial *granule*, i.e., given

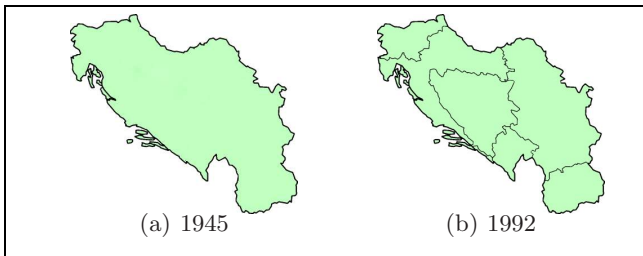


Fig. 1. Example of spatio-temporal geometric value

a granularity G and an index $i \in \mathcal{IS}$, $G(i)$ is a granule of G that identifies a subset of the corresponding domain. Through granules we can specify the validity spatio-temporal bounds of attribute values. For instance, we can say that a value reporting the measure of the daily temperature in Dublin is defined for the first and the second day of January 2000. The granules of interest for this example can be identified by three textual labels: ‘01/01/2000’, ‘02/01/2000’, and ‘*Dublin*’, that respectively identify two temporal and one spatial granules.

The interior of granules of the same granularity cannot overlap¹. Moreover, non-empty temporal granules must preserve the order given by the index set.

Spatial and temporal granularities are related by the *finer-than* relationship [5]. Such a relationship formalises the intuitive idea that different granularities correspond to different partitions of the domain, and that, given a granule of a granularity G , usually a granule of a coarser granularity H exists that properly includes it. For example, granularity *days* is finer-than *months*, and granularity *months* is finer-than *years*. Likewise, *municipalities* is finer-than *countries*. If a granularity G is finer-than H , we also say that H is coarser-than G .

Beyond the conventional database values, an *ST_ODMG* database schema can include spatial, temporal, and spatio-temporal values. 2-dimensional geometric vector features can be represented. Multigranular spatial and temporal data are uniformly defined as instances of two parametric types: *Spatial_{SG}(τ)* and *Temporal_{TG}(σ)*, which are specified according to a granularity (spatial and temporal, respectively) and an inner type. The inner type can be a type without spatio-temporal characteristics, or a geometric type. Moreover, multigranular spatio-temporal types are defined as compositions of *Spatial* and *Temporal*.

Figure 1 illustrates the changes of the political boundaries of Balkan nations in different historical periods. A legal *ST_ODMG* type specification for this value is *Temporal_{years}(Spatial_{countries}(set(Polygon)))*.

Granularity conversions are provided in order to represent data at the most appropriate level of detail for a specific task, i.e., to increase or reduce the level of detail used for data representation. In *ST_ODMG* different semantics can be applied when converting values. The conversion of multigranular geometrical features is obtained through the composition of model-oriented and cartographic map generalisation operators [23] that guarantee topological consistency [4,24], an essential property for data usability. Refinement operators perform the inverse functions. Such operators are classified according to the type of conversion applied [9]. For example, *merge* operators merge adjacent features of the same dimension into a single one, while *splitting* operators subdivide single features in adjacent features of the same dimension. Other supported operators perform *contraction* and *thinning* (whose inverse is *expansion*); *abstraction* and *simplification* (whose inverse is *addition*).

On the other hand, the model provides also operators for converting quantitative (i.e., not geometrical) attribute values, both temporal and spatial. These conversions are classified in families according to the semantics of the operation

¹ Temporal granules, according to the definition in [5], do not overlap, while spatial granules can overlap on the granule boundaries.

performed: *selection* (e.g., **projection**, **main**, **first**), and *aggregation* (e.g., **sum**, **average**) convert values to coarser representation; their inverse functions, *restriction* and *splitting*, convert attribute values to finer representations, according to downward hereditary property [25] or according to a probability distribution, respectively. The different semantics we provided for converting spatio-temporal values at finer granularities have been introduced to address indeterminacy and imprecision that affect such type of conversion.

Granularity conversions have been proven to return legal values of the type system defined [9]. Moreover, those that generalize geometric attribute values to coarser spatial granularities have been proven preserve the semantics of the spatio-temporal data represented [9].

3 Multigranular Spatio-temporal Queries

The query language we propose extends value comparison and object navigation paradigms of OQL [10] to support multigranular spatio-temporal values. *Multigranular spatio-temporal path expressions* are the key concept of the resulting language: indeed, they are used to specify multigranular spatio-temporal queries. The access to multigranular class attributes is performed according to spatial and temporal *elements* and *expressions*, which specify portions of *SPACE* and *TIME* domains at a given granularity, in explicit or implicit form, respectively. Whenever path expressions involve different granularities, during their evaluation granularity conversions described in the previous section are applied. In the remaining of the section, multigranular spatial and temporal elements, expressions, and path expressions are described.

3.1 Spatial and Temporal Elements and Expressions

Temporal elements have been formally introduced by Gadia [16], and then extended with respect to temporal granularities by Bertino et al. [3]. In a multigranular model, a temporal or spatial element is a set of granules expressed at the same granularity. For instance, $\{1999, 2000, 2001\}^{years}$ is a temporal element at granularity *years*, whereas $\{Roma, Berlin, Zurich\}^{municipalities}$ is a spatial element at granularity *municipalities*. Temporal and spatial elements can be converted to different granularities. Let Υ^H be a temporal (respectively, a spatial) element, with temporal (respectively, spatial) granularity H . Let G be a temporal (respectively, spatial) granularity, such that $G \preceq H$ or $H \preceq G$, then $G(\Upsilon^H)$ denotes the conversion of Υ^H to granularity G . For instance, $months(\{1999, 2000, 2001\})^{years}$ denotes the temporal element at granularity *months* $\{\text{January 1999, February 1999, } \dots, \text{December 2001}\}^{months}$.

Temporal and spatial elements at the same granularity can be combined by using the conventional set operators: complement ($\setminus$), intersection ($\cap$), and union ($\cup$). Moreover, operators *first* and *last*, relying on the order of temporal granules, can be applied to temporal elements. For instance, $first(\{1999, 2000, 2005\}^{years})$ returns the year 1999.

Temporal and spatial elements can be represented explicitly, as in the previous examples, or implicitly, by means of *temporal* and *spatial expressions*. Expressions represent conditions that are evaluated on database objects. Intuitively, the temporal and spatial elements resulting from temporal and spatial expressions specify *when* and *where* such conditions are satisfied. Conditions are specified through temporal and spatial variations of conventional comparison operators (i.e., $=, <, >, <=, >=$) and binary topological relationships as defined by Egenhofer and Franzosa [14] (i.e., *equal*, *disjoint*, *meet*, *overlap*, *contains*, *inside*, *cover*, *coveredby*). For instance, each temporal comparison operator (i.e., $=_T, <_T, >_T, <=_T, >=_T$) compares (spatio-)temporal and conventional values, and the resulting temporal expression represents the set of instants *when* the comparison is satisfied. Analogously, spatial expressions involving spatial comparison operators (i.e., $=_S, <_S, >_S, <=_S, >=_S$) specify *where* a comparison is satisfied. Consider for example the spatio-temporal value:

$$v = \{ \langle 2004, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{United Kingdom}, \text{'Blair'} \rangle \}^{\text{countries}} \rangle, \\ \langle 2007, \{ \langle \text{France}, \text{'Fillon'} \rangle, \langle \text{United Kingdom}, \text{'Brown'} \rangle \}^{\text{countries}} \rangle \}^{\text{years}}.$$

The temporal expression $v=_T$ 'Brown' returns the temporal element $\{2007\}^{\text{years}}$, whereas the spatial expression $v=_S$ 'Brown' returns the spatial element $\{\text{United Kingdom}\}^{\text{countries}}$.

Similarly, temporal and spatial variations of topological relationships are provided. For instance, equals_T , disjoint_T , meet_T , overlaps_T , contains_T , inside_T can be applied between spatio-temporal values: the resulting temporal expressions represent the set of instants when the values satisfy the topological relationships. By contrast, equals_S , disjoint_S , meets_S , overlaps_S , contains_S , inside_S are applied between spatial and spatiotemporal values, and return where the specified topological relationships are satisfied.

3.2 Spatio-temporal Access and Path Expressions

Spatio-temporal path expressions are extension of conventional path expressions as used in object-oriented languages and models: the access to object attribute values is specified also according to temporal and spatial elements as described in the previous section. The access to conventional object attribute values is obtained through the usual postfix dot notation. To specify the access to spatio-temporal attribute values the $\downarrow$ operator is provided. As in conventional path expressions, the nesting of attribute accesses is allowed, and internal nodes in a path expression must result in single object identifiers.

Example 1. We define an object type for describing geographical historical maps. For each map the information recorded includes the political boundaries, the capital and the name of the head of government of each country. The definition of class Map in ST-ODMG syntax is as follows:

```
class Map ((extent Maps, key ...) {
    attribute Temporalyears(Spatialcountries(set(Polygon))) boundaries {...};
    attribute Temporalyears(Spatialcountries(Point)) capitals {...};
    attribute Temporalyears(Spatialcountries(string)) head_of_government {...};
};
```

Consider object o of type Map and value v of attribute `head_of_government` such that:

$$v = \{ \langle 2004, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{United Kingdom}, \text{'Blair'} \rangle \} \}^{\text{countries}}, \\ \langle 2005, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{Germany}, \text{'Merkel'} \rangle, \\ \langle \text{United Kingdom}, \text{'Blair'} \rangle \} \}^{\text{countries}}, \\ \langle 2007, \{ \langle \text{France}, \text{'Fillon'} \rangle, \langle \text{Germany}, \text{'Merkel'} \rangle, \\ \langle \text{United Kingdom}, \text{'Brown'} \rangle \} \}^{\text{countries}} \}^{\text{years}}.$$

Then, the temporal path expression $o.\text{head\_of\_government} \downarrow \{2007\}^{\text{years}}$ returns the spatial value: $\{ \langle \text{France}, \text{'Fillon'} \rangle, \langle \text{Germany}, \text{'Merkel'} \rangle, \langle \text{United Kingdom}, \text{'Brown'} \rangle \}^{\text{countries}}$.

By contrast, the spatial path expression $o.\text{head\_of\_government} \downarrow \{ \text{France} \}^{\text{countries}}$ returns the temporal value: $\{ \langle 2004, \text{'Raffarin'} \rangle, \langle 2007, \text{'Fillon'} \rangle \}^{\text{years}}$. $\square$

Whenever the spatial or temporal element involved in a spatio-temporal path expression includes more than one granule, the access results in a subset of the specified attribute value that corresponds to the *restriction* of the attribute value to the given element, as illustrated in the following Example.

Example 2. Given object o and value v of Example 1, the temporal path expression $o.\text{head\_of\_government} \downarrow \{2004, 2005\}^{\text{years}}$ returns the spatio-temporal value:

$$v = \{ \langle 2004, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{United Kingdom}, \text{'Blair'} \rangle \} \}^{\text{countries}}, \\ \langle 2005, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{Germany}, \text{'Merkel'} \rangle, \\ \langle \text{United Kingdom}, \text{'Blair'} \rangle \} \}^{\text{countries}} \}^{\text{years}}.$$

Moreover, the spatial path expression $o.\text{head\_of\_government} \downarrow \{ \text{France}, \text{United Kingdom} \}^{\text{countries}}$ returns the spatio-temporal value:

$$v' = \{ \langle 2004, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{United Kingdom}, \text{'Blair'} \rangle \} \}^{\text{countries}}, \\ \langle 2005, \{ \langle \text{France}, \text{'Raffarin'} \rangle, \langle \text{United Kingdom}, \text{'Blair'} \rangle \} \}^{\text{countries}}, \\ \langle 2007, \{ \langle \text{France}, \text{'Fillon'} \rangle, \langle \text{United Kingdom}, \text{'Brown'} \rangle \} \}^{\text{countries}} \}^{\text{years}}. \quad \square$$

Given a spatio-temporal path expression, the following consistency property holds.

Property 1. (Spatio-temporal path expression consistency) Given object o , attribute a defined for o , and (temporal or spatial) element el , the access $o.a \downarrow el$ resulting in value v verifies the following consistency conditions:

If a is a *temporal (spatial) attribute* and

- el is a spatial (temporal) element, $o.a \downarrow el$ is undefined;
- el is a temporal (spatial) element including a single granule, v is a conventional i.e., non-temporal (non-spatial) value;
- el is a temporal (spatial) element including two or more granules, v is a temporal (spatial) value;

If a is a *spatio-temporal attribute* and

- el is a temporal element including a single granule, v is a spatial value;
- el is a spatial element including a single granule, v is a temporal value;
- el is a spatial element including two or more granules, v is a spatio-temporal value. ∇

Whenever the granularity in a spatio-temporal path expression differs from that of the value accessed, a granularity conversion is applied. To apply a granularity conversion, the starting and the target granularities must be related according to the finer-than relationship. If for the attribute being accessed a suitable granularity conversion has been defined in the database schema, such a conversion is applied. Otherwise, we can specify which conversion to apply in the path expression by using the access operator of the form $\downarrow^{gconv}$, where $gconv$ is a granularity conversion. To avoid conflicts, granularity conversions specified in path expressions take precedence over those specified in the schema. This enables to convert attribute values according to different semantics. Moreover, even if not required to perform the access, the application of an existing granularity conversion can be forced by specifying the access operator in the form $\downarrow^G$, where G is the target granularity of the conversion. For instance, when performing a temporal access, the value can be conveniently converted to a different spatial granularity, and vice versa, to format the access result.

Example 3. Given value v for attribute `head_of_government` of Example 1, the access `o.head_of_government` $\downarrow^{\text{last}_{years \rightarrow decades}}$ $\{2000-2009\}^{decades}$ returns the spatial value:

$\{\langle \text{France}, \text{'Fillon'} \rangle, \langle \text{Germany}, \text{'Merkel'} \rangle, \langle \text{United Kingdom}, \text{'Brown'} \rangle\}^{countries}$.

To perform such an access, value v has been first converted to granularity *decades* according to the granularity conversion $\text{last}_{years \rightarrow decades}$. Then, the resulting spatio-temporal value has been accessed according to the temporal element $\{2000-2009\}^{decades}$. $\square$

4 Querying Multigranular Spatio-temporal Objects

The spatio-temporal query language described in the previous section extends the OQL syntax [10] to multigranular spatio-temporal path expressions as described above. Queries have the usual OQL `select-from-where` form. According to the OQL specification, spatio-temporal path expressions can be used in the target list to specify the data to retrieve, and in the where clause to express the conditions against multigranular spatio-temporal objects.

The complete syntax for the specification of multigranular spatio-temporal queries is presented in the Appendix. In this section, we describe its application through some illustrative examples, emphasizing the use of granularity conversions when querying multigranular attribute values. In particular, we focus on the access to spatio-temporal values, performed converting both spatio-temporal elements and attribute values. The access to temporal and spatial values follows straightforwardly. Moreover, we demonstrate the expressiveness of the spatio-temporal extensions of comparison and topological operators, which can be used to restrict the constraints used in the search, as well as to characterize the query results.

In these examples, the database schema including the class **Map** of Example 1 is extended with class **Flight**, representing passenger aircrafts. For each flight performed by an aircraft, we record its flight number, the departure and arrival airports, and the name of the pilot who flew the plane. Moreover, we record the spatial location of the aircraft during the flight. According to this specification, the class **Flight** in *ST-ODMG* is defined as follows:

```
class Flight (extent Flights, key ..) {
  attribute Temporalminutes(string) flightNum;
  attribute Temporalminutes(string) departure;
  attribute Temporalminutes(string) arrival;
  attribute Temporalhours(string) pilot {
    mainhours→months, restrhours→minutes };
  attribute Temporalminutes(Spatialmeters(set(Region))) trips {
    r_mergemeters→provinces, r_contrprovinces→countries };
};
```

In particular, the spatio-temporal attribute **trips** reports the spatial location of the aircraft over time. For such attribute two spatial multigranular conversions have been specified, which support converting the aircraft location from the spatial granularity *meters* to the spatial granularity *provinces*, and from *provinces* to *countries*. The first conversion specified, i.e., **r_merge**, merges multiple regions in order to give the spatial representation of the aircraft with respect to a single region; by contrast, **r_contr** (i.e., region contraction), collapses regions in single points preserving topological consistency.

Other conversions have been specified for the temporal attribute **pilot**. Granularity conversion **main** converts the temporal value of this attribute to granularity *months* selecting the more frequent pilot name recorded for each month; by contrast, granularity conversion **restr** converts the same values to finer granularity *minutes* by applying the downward hereditary property [25], according to which if a multigranular value assumes the value *v* in a granule *g*, value *v* also refers to any *finer* granule *g'* included in *g*. For instance, given a value representing the address of a person, defined with temporal granularity *years*, each value referring to a year *Y* is the valid address of that person for every day of year *Y*.

Given the above specification for class **Flight**, the following query retrieves (the names of) the pilots flying the flight with number 'AZ555' during December 2007:

```
select distinct a.pilot ↓ hours({12/2007}months)
from Flight f
where f.flightNum ↓ minutes({12/2007}months) = 'AZ555'
```

We can further refine the query asking which pilot has more often flown such a flight, during the same period of time. The query has to be modified as follows:

```
select a.pilot ↓ {12/2007}months
from Flight f
where f.flightNum ↓ minutes({12/2007}months) = 'AZ555'
```

The granularity conversion **main** is automatically applied for converting the value of attribute **pilot** to granularity *months*. According to the semantics of this conversion, the value of **pilot** occurring more often in the selected month is the value for this month.

Beyond quantitative queries, multigranular spatio-temporal expressions that return geometric data can also be defined. For example, what follows is an example of a more complex query which retrieves the capitals and the corresponding countries that have been flown over by flight number ‘AZ555’ flown on Christmas 2007:

```
select distinct m.capitals,
  (f.trips↓countries((f.flightNum↓minutes({25/12/2007}days))=T‘AZ555’))
  <>S ⊥
from Map m, Flight f
where f.flightNum ↓minutes({25/12/2007}days) = ‘AZ555’ and
  f.trips ↓minutes({25/12/2007}days) overlaps
  m.capitals ↓restryears→days {25/12/2007}days.
```

Starting from the where clause, the topological test:

```
f.trips ↓minutes({25/12/2007}days) overlaps
m.capitals ↓restryears→days {25/12/2007}days
```

compares two spatial values at granularity *countries*. The first value represents the trajectory of aircrafts, given through the position of their centroid over time, which is returned executing the temporal access on attribute **trips**. The temporal element specified for the access is given at granularity *minutes* as expected for such attribute, but granularity conversions **r_merge** and **r_contr**, defined in class **Flight**, are applied to convert its value to granularity *countries*. The result is further refined thanks to the access to attribute **flightNum** specified in the where clause. This spatial value is compared in the topological test with the capitals of the countries represented in some map on Christmas 2007, which are returned accessing attribute **capitals**. Note that the spatio-temporal value of attribute **capitals** is converted from granularity *years* to granularity *days* before evaluating the expression by applying granularity conversion **restr**, which supports downward hereditary property [25]. The spatial element resulting from the topological comparison includes the granules at granularity *countries* where the trajectories of the aircraft overlapped some country capitals.

To answer the query, however, this value is not sufficient, and we can note no attribute is defined in the database schema to represent the countries given in the maps. However, this information is implicitly stored because of the multigranular support, and through multigranular spatio-temporal path expressions we can make it explicit. Indeed, the required value is obtained by the spatial expression specified in the target list:

```
((f.flightNum ↓minutes({25/12/2007}days))=T ‘AZ555’)) <>S ⊥
```

This path expression is a little more complex than the previous ones: we have two nested path expressions, and a temporal and a spatial expression. First, the

temporal path expression $\mathbf{f.flightNum} \downarrow \text{minutes}(\{25/12/2007\}^{days})$ returns the flight numbers of the flights flown during Christmas 2007. This value (let it be v) is temporally compared with the flight number ‘AZ555’: the temporal expression $v =_T \text{‘AZ555’}$ returns the temporal element at granularity *minutes* specifying the exact time of occurrence for the flight ‘AZ555’ during Christmas 2007. Then, this temporal element, namely te , is used to access the attribute **trips** in the temporal path expression $\mathbf{f.trips} \downarrow \text{countries } te$. This temporal access returns the trips performed by the flight already selected through the where clause during the period of time specified by temporal element te . Before performing the access, the value of **trips** is converted at spatial granularity *countries* by applying granularity conversions **r_merge** and **r_contr**. The spatial value at granularity *countries* resulting from the access (let it be v') is then involved in the spatial comparison $v' <_S \perp$. This straightforward comparison simply returns the spatial element including the granules where v' is defined, at granularity *countries*, which at the end represents the (granules of) countries whose capitals have been flown over by the aircrafts selected through the where clause.

5 Conclusions

In this paper we have defined a multigranular spatio-temporal object oriented query language. The language is based on the spatio-temporal model we previously defined in [9]. The query language extends OQL, the ODMG query language, to support multigranular spatio-temporal access. The access to spatio-temporal values is given in terms of spatial and temporal elements and expressions. Their use in multigranular spatio-temporal path expressions, combined with multigranular conversions, allows one to access and compare spatio-temporal values expressed at different granularities.

Unlike previous work on spatio-temporal query languages, in this paper we provide a formalization that addresses several important issues arising from the introduction of spatio-temporal multigranularity. Despite the importance of multigranularity and of the integrated management of spatio-temporal information, no currently available DBMS include suitable tools for dealing with spatio-temporal data at different levels of detail. The overall design proposed in [9] and in this paper is suitable for the development of both object-oriented and relational multigranular spatio-temporal models and query languages. Our design relies on ODMG and the related query language OQL: the features we introduce to support spatio-temporal multigranularity extend the model type systems with specific types and suitable operators for handling time and space, as well as spatial and temporal granularities. Accordingly, the model Data Definition Language, value comparison and object navigation have been extended to support multigranular spatio-temporal types and expressions.

The work presented in this paper can be refined and extended in different ways. First of all, we plan to develop a software prototype to assess the effectiveness and to test the performance of our approach. In this respect, a suitable

indexing system for spatio-temporal values (such as the one proposed in [8] for dealing with the expiration of historical values, and the one proposed in [27]), an efficient representation for those values to support value coalescing, and an efficient implementation for granularity conversions, are desirable. Moreover, in order to build a comprehensive spatio-temporal information system, conventional spatial operations, like intersections and overlay, should be integrated in the formalization provided. Finally, we plan to extend spatio-temporal comparison operators to support qualitative semantics both for time and space.

Acknowledgement

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support. The work of Elena Camossi is supported by the Irish Research Council for Science, Engineering and Technology.

References

1. Andrienko, G., Malerba, D., May, M., Teisseire, M.: Mining spatio-temporal data. *Journal of Intelligent Information Systems* 27(3), 187–190 (2006)
2. Balley, S., Parent, C., Spaccapietra, S.: Modelling Geographic Data with Multiple Representations. *International Journal of Geographical Information Science* 18(4), 327–352 (2004)
3. Bertino, E., Ferrari, E., Guerrini, G., Merlo, I.: T-ODMG: An ODMG Compliant Temporal Object Model Supporting Multiple Granularity Management. *Information Systems* 28(8), 885–927 (2003)
4. Bertolotto, M.: Geometric Modeling of Spatial Entities at Multiple Levels of Resolution. Ph.D. Thesis, Università degli Studi di Genova (1998)
5. Bettini, C., Dyreson, C., Evans, W., Snodgrass, R., Wang, X.: A Glossary of Time Granularity Concepts. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) *Dagstuhl Seminar 1997*. LNCS, vol. 1399, pp. 406–413. Springer, Heidelberg (1998)
6. Bettini, C., Jajodia, S., Wang, X.: *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, Heidelberg (2000)
7. Bittner, T.: Reasoning about qualitative spatio-temporal relations at multiple levels of granularity. In: van Harmelen, F. (ed.) *Proc. of the 15th European Conf. on Artificial Intelligence*. IOS Press, Amsterdam (2002)
8. Camossi, E., Bertino, E., Guerrini, G., Mesiti, M.: Handling Expiration of Multigranular Temporal Objects. *Journal of Logic and Computation* 14(1), 23–50 (2004)
9. Camossi, E., Bertolotto, M., Bertino, E.: A multigranular Object-oriented Framework Supporting Spatio-temporal Granularity Conversions. *International Journal of Geographical Information Science* 20(5), 511–534 (2006)
10. Cattel, R., Barry, D., Berler, M., Eastman, J., Jordan, D., Russel, C., Schadow, O., Stanienida, T., Velez, F.: *The Object Database Standard: ODMG 3.0*. Morgan-Kaufmann, San Francisco (2000)

11. Claramunt, C., Thériault, M.: Managing Time in GIS: an event oriented approach. In: Clifford, J., Tuzhilin, A. (eds.) *Proc. of the Int'l Workshop on Temporal Databases: Recent Advances in Temporal Databases*, pp. 23–42. Springer, Heidelberg (1995)
12. Claramunt, C., Jiang, B.: Hierarchical Reasoning in Time and Space. In: *Proc. of the 9th Int'l Symposium on Spatial Data Handling*, pp. 41–51 (2000)
13. Chen, C.X., Zaniolo, C.: SQLST: A Spatio-Temporal Data Model and Query Language. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) *ER 2000. LNCS*, vol. 1920, pp. 96–111. Springer, Heidelberg (2000)
14. Egenhofer, M.J., Franzosa, R.D.: Point-Set Topological Spatial Relations. *International Journal of Geographical Information Science* 5(2), 161–174 (1991)
15. Erwig, M., Schneider, M.: STQL – A Spatio-Temporal Query Language. In: Ladner, R., Shaw, K., Abdelguerfi, M. (eds.) *Mining Spatio-Temporal Information Systems*, ch. 6, pp. 105–126. Kluwer Academic Publishers, Dordrecht (2002)
16. Gadia, S.K.: A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems* 13(4), 418–448 (1988)
17. Griffiths, T., Fernandes, A.A.A., Paton, N.W., Barr, R.: The Tripod spatio-historical data model. *Data Knowledge and Engineering* 49(1), 23–65 (2004)
18. Güting, R.H., Bhölen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Shneider, M., Vazirgiannis, M.: A Foundation for Representing and Querying Moving Objects. *ACM Transaction On Database Systems* 25, 1–42 (2000)
19. Huang, B., Claramunt, C.: STOQL: An ODMG-based Spatio-Temporal Object Model and Query Language. In: *Proc. of the 10th Int'l Symposium on Spatial Data Handling* (2002)
20. Khatri, V., Ram, S., Snodgrass, R.T., O'Brien, G.: Supporting User Defined Granularities and Indeterminacy in a Spatio-temporal Conceptual Model. *Annals of Mathematics and Artificial Intelligence* 36(1), 195–232 (2002)
21. Kim, D.H., Ryu, K.H., Kim, H.S.: A spatio-temporal database model and query language. *Journal of Systems and Software* 55(2), 129–149 (2000)
22. Kulik, L., Duckham, M., Egenhofer, M.J.: Ontology driven Map Generalization. *Journal of Visual Language and Computing* 16(2), 245–267 (2005)
23. Muller, J.-C., Lagrange, J.P., Weibel, R. (eds.): *GIS and Generalization: methodology and practice*. Taylor and Francis, Abington (1995)
24. Saalfeld, A.: Topologically consistent line simplification with the Douglas-Peucker algorithm. *Cartography and Geographic Information Science* 26(1), 7–18 (1999)
25. Shoham, Y.: Temporal Logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence* 33(1), 89–104 (1987)
26. Stell, J.G., Worboys, M.: Stratified Map Spaces: A Fomal Basis for Multi-Resolution Spatial Databases. In: *Proc. of 8th Int'l Symposium on Spatial Data Handling* (1998)
27. Zhang, D., Gunopulos, D., Tsotras, V.J., Seeger, B.: Temporal Aggregation over Data Streams using Multiple Granularities. In: *Proc. of 8th Int'l Conf. on Extending Database Technology*, pp. 643–663 (2002)
28. ORACLETM, Oracle Corp., <http://www.oracle.com>
29. PostgreSQL, Inc., <http://www.postgresql.org>

Appendix: Syntax of Multigranular Spatio-temporal Queries

In the following, we report the syntax of *ST*_ODMG queries in Backus-Naur form.

```

<query> ::= select [distinct] <target_list> from <q_source> where
    <search_condition>;
<target_list> ::= [<target_list>, ] <value>
<q_source> ::= [<q_source>] class_name class_alias
<search_condition> ::= [not] <cond> |
    <search_condition> <bin_bool_op> <search_condition>
<value> ::= [<path>].value | <conv_value> | <t_value> | <s_value> | <st_value>
<path> ::= object_name.<internal_path>
<internal_path> ::= [<internal_path>.]attribute_name
<conv_value> ::= conv_attr_name | <t_value> ↓[<g_conv>] t_granule | [not] <conv_value> |
    <s_value> ↓[<g_conv>] s_granule | <conv_value> <bin_op> <conv_value> |
    <t_value> <comp_op> <conv_value> | <s_value> <comp_op> <conv_value>
<t_value> ::= temp_value | temp_attr_name [<t_access>] | [not] <t_value> |
    <t_value> <bin_op> <t_value> | <st_value> ↓[<g_conv>] s_granule
<s_value> ::= spat_value | spat_attr_name [<s_access>] | [not] <s_value> <s_value> |
    <bin_op> <s_value> | <st_value> ↓[<g_conv>] t_granule
<st_value> ::= spatio-temp_value | st_attr_name [<access>]
<access> ::= <s_access> | <t_access>
<t_access> ::= ↓[<g_conv>] <temp_elem>
<temp_elem> ::= explicit_temp_elem | <t_expr> | t_gran_name (<temp_elem>) |
    first(<temp_elem>) | last(<temp_elem>)
<t_expr> ::= <temp_value> <bin_op_T> <conv_value>
<s_access> ::= ↓[<g_conv>] <spat_elem>
<spat_elem> ::= explicit_spat_elem | <s_expr> | s_gran_name (<spat_elem>)
<s_expr> ::= <spat_value> <bin_ops> <conv_value>
<bin_op> ::= <comp_op> | <toprel> | <arithm_op> | <bool_op> | <set_op>
<arithm_op> ::= + | - | / | *
<bool_op> ::= and | or
<set_op> ::= \ | ∪ | ∩
<comp_op> ::= = | <> | < | > | <= | >=
<toprel> ::= equal | disjoint | meet | overlap | contains | inside | cover | coveredby
<bin_op_T> ::= <comp_op_T> | <toprel_T>
<comp_op_T> ::= =_T | <_T | <_T | >_T | <=_T | >=_T
<toprel_T> ::= disjoint_T | meet_T | overlap_T | equal_T | contains_T | inside_T |
    cover_T | coveredby_T
<bin_ops> ::= <comp_ops> | <toprels>
<comp_ops> ::= =_S | <_S | <_S | >_S | <=_S | >=_S
<toprels> ::= disjoint_S | meets_S | overlaps_S | equals_S | contains_S | inside_S |
    covers_S | coveredbys_S

```

Non terminal symbols *conv_attr_name*, *spatial_attr_name*, *temporal_attr_name*, *st_attr_name* are names for conventional, spatial, temporal and spatio-temporal attributes, while *attr_name* is a generic attribute name, used in a nested path. Nested paths must result in a single object identifier; *temp_value*, *spat_value*, *spatio-temp_value* are explicit temporal, spatial and spatiotemporal values, respectively; *class_name* is a class name, and *class_alias* is the class alias. *s_granule* and *t_granule* are a spatial and a temporal granule, respectively; *t_gran_name* and *s_gran_name* are a temporal and a spatial granularity name, respectively; *explicit_temp_element* and *explicit_spat_element* are a temporal and a spatial element, respectively, represented explicitly. Terminal elements <temp_elem> and <spat_elem>, when referred to in temporal and spatial access (<t_access> and <s_access>), include more than one granule. Finally, terminal element <g_conv> is a granularity conversion. Note that <g_conv>, when applied in attribute access, represents a granularity conversion that should be applicable to the attribute value.

Space-Partitioning-Based Bulk-Loading for the NSP-Tree in Non-ordered Discrete Data Spaces*

Gang Qian¹, Hyun-Jeong Seok², Qiang Zhu², and Sakti Pramanik³

¹ Department of Computer Science,
University of Central Oklahoma, Edmond, OK 73034, USA
gqian@ucok.edu

² Department of Computer and Information Science,
The University of Michigan, Dearborn, MI 48128, USA
{hseok, qzhu}@umich.edu

³ Department of Computer Science and Engineering,
Michigan State University, East Lansing, MI 48824, USA
pramanik@cse.msu.edu

Abstract. Properly-designed bulk-loading techniques are more efficient than the conventional tuple-loading method in constructing a multidimensional index tree for a large data set. Although a number of bulk-loading algorithms have been proposed in the literature, most of them were designed for continuous data spaces (CDS) and cannot be directly applied to non-ordered discrete data spaces (NDDS). In this paper, we present a new space-partitioning-based bulk-loading algorithm for the NSP-tree — a multidimensional index tree recently developed for NDDSs. The algorithm constructs the target NSP-tree by repeatedly partitioning the underlying NDDS for a given data set until input vectors in every subspace can fit into a leaf node. Strategies to increase the efficiency of the algorithm, such as multi-way splitting, memory buffering and balanced space partitioning, are employed. Histograms that characterize the data distribution in a subspace are used to decide space partitions. Our experiments show that the proposed bulk-loading algorithm is more efficient than the tuple-loading algorithm and a popular generic bulk-loading algorithm that could be utilized to build the NSP-tree.

1 Introduction

Applications that require multidimensional indexes often involve a large amount of data, where a bulk-loading (BL) approach can be much more efficient than the conventional tuple-loading (TL) method in building the index. In this paper, we propose an efficient bulk-loading algorithm for a recently-developed index tree, called the NSP-tree [14], in non-ordered discrete data spaces (NDDS).

A non-ordered discrete data space models vector data whose components are discrete with no inherent ordering. Such non-ordered discrete domains as the

* Research supported by US National Science Foundation (under grants # IIS-0414576 and # IIS-0414594), US National Institute of Health (under OK-INBRE Grant # P2PRR016478), The University of Michigan, and Michigan State University.

gender and profession are quite common in database applications. To support efficient similarity queries in NDDSs, the space-partitioning-based NSP-tree was proposed in [14]. The conventional TL algorithm of the NSP-tree inserts one vector at time into a leaf node of the tree. When a leaf overflows, its corresponding data space is split into two subspaces and its vectors are moved into one of the subspaces to which they belong. While the TL method is capable of constructing a high-quality NSP-tree, it may take too long to index a large data set, which is typical for many contemporary applications. For example, genome sequence databases, with non-ordered discrete nucleotides ‘a’, ‘g’, ‘t’ and ‘c’, have been growing rapidly in size in the past decade. The size of the GenBank, a popular collection of publicly available genome sequences, increased from 71 million residues (base pairs) and 55 thousand sequences in 1991, to more than 65 billion residues and 61 million sequences in 2006 [8]. Hence, an efficient bulk-loading technique is essential in building an NSP-tree for such applications. Unfortunately, there is no bulk-loading algorithm specially designed for the NSP-tree.

Bulk-loading has been an important research topic for multidimensional index structures. There are a number of bulk-loading algorithms proposed for multidimensional indexes in continuous data spaces (CDS), such as the R-tree [9]. One major category of such bulk-loading algorithms is based on sorting, which can be further divided into the bottom-up approach and the top-down approach [1]. In the bottom-up approach [6,11,12,16], vectors to be indexed are sorted according to certain global one-dimensional criteria and then placed in the leaves in the sorted order. The minimum bounding rectangles (MBR) of the leaves are sorted using the same criteria to build the first non-leaf level. The index is thus recursively constructed level by level until all MBRs can be fit into one node. In the top-down approach [2,7], all vectors are sorted using certain criteria and then divided into M subsets of roughly equal size, where M is the size of the root. The MBRs of the subsets are stored in the root. Subtrees corresponding to the subsets are recursively constructed in the same manner until vectors in the subsets can be fit into a leaf node. Unfortunately, these sorting-based algorithms cannot be directly applied in the NDDS, where no ordering exists.

Another category of bulk-loading algorithms is called the generic bulk-loading. Instead of sorting, these algorithms utilize some basic operations/interfaces (e.g., splitting an overflow node) from the corresponding TL algorithm of the target index tree. Therefore, generic bulk-loading algorithms can be applied to every target index tree having required operations although they may not be optimized for the index tree due to their generic nature. One such popular generic bulk-loading algorithm [3], denoted by GBLA in this paper, employs a buffered tree structure to reduce disk I/Os by accumulating inserted vectors in the buffer of a tree node and pushing the buffered vectors into child nodes when the buffer is full. The leaf nodes of the target tree are built first. The MBR of the leaves are then used to build their parent nodes. The target tree is, therefore, built level by level from bottom up. Another type of generic bulk-loading algorithms utilizes a sample-based approach [5,4]. A seed index is first built in memory based on sample vectors from the data set. The remaining vectors are then assigned to

individual leaves of the seed structure. The leaves are processed in the same way recursively until the whole target index is constructed.

Recently, a bulkloading algorithm named NDTBL [17] was proposed for the ND-tree [13,15], a data-partitioning-based indexing method for NDDSs. NDTBL first builds linked subtrees of the target ND-tree in memory. It then adjusts those subtrees to form a balanced target ND-tree. Some operations in the TL algorithm for the ND-tree are extended and utilized to choose and split data sets/nodes during the process.

In this paper, we propose a new bulk-loading algorithm for the NSP-tree, called NSPBL (the NSP-tree Bulk-Loading). It is a space-partitioning-based algorithm that employs a bottom-up process. Vectors to be loaded are first placed into a solo (typically oversized) leaf node of an intermediate tree structure. The leaf node(s) are repeatedly divided into a number of normal or oversized leaf nodes based on space partitions, while the tree structure grows upward as corresponding non-leaf nodes are created and split. Histograms that record the space distribution of the vectors in a leaf are used to find balanced splits of the subspace for the leaf. Memory buffers are adopted in the bulk-loading process to reduce unnecessary disk accesses. A multi-way splitting strategy that allows an oversized node to be directly split into more than two new nodes is employed to reduce splitting overhead. The final intermediate tree has the same structure as that of the target NSP-tree. Therefore, no post-processing is needed. Our experimental study demonstrates that NSPBL is more efficient than the conventional TL algorithm and the popular generic GBLA in constructing the NSP-tree. The quality of the built NSP-trees is comparable among these three methods.

The rest of this paper is organized as follows. Section 2 introduces the essential concepts and notation for the NDDS and the NSP-tree. Section 3 discusses the details of NSPBL. Section 4 presents the experimental results. Section 5 concludes the paper.

2 Preliminaries

The concepts of NDDSs were discussed in [13,15], while the NSP-tree was introduced in [14]. For completion, we briefly describe some relevant concepts here.

2.1 Concepts and Notation

A d -dimensional NDDS Ω_d is defined as the Cartesian product of d alphabets: $\Omega_d = A_1 \times A_2 \times \dots \times A_d$, where $A_i (1 \leq i \leq d)$ is the *alphabet* of the i -th dimension of Ω_d , consisting of a finite set of letters with no natural ordering. For simplicity, we assume A_i 's are the same in this paper. As shown in [15], the discussion can be readily extended to NDDSs with different alphabets. $\alpha = (a_1, a_2, \dots, a_d)$ is a vector in Ω_d , where $a_i \in A_i (1 \leq i \leq d)$. A *discrete rectangle* R in Ω_d is defined as $R = S_1 \times S_2 \times \dots \times S_d$, where $S_i \subseteq A_i (1 \leq i \leq d)$ is called the i -th *component set* of R . R is also called a *subspace* of Ω_d . For a given set SV of vectors, the *discrete minimum bounding rectangle (DMBR)* of SV is defined

as the discrete rectangle whose i -th component set ($1 \leq i \leq d$) consists of all letters appearing on the i -th dimension of the given vectors. The DMBR of SV is also called the *current data space* for the vectors in SV . The DMBR of a set of discrete rectangles can be defined similarly. For a given space (subspace) $\Omega'_d = A'_1 \times A'_2 \times \dots \times A'_d$, a *space split* of Ω'_d on the i -th dimension consists of two subspaces $\Omega_d'^1 = A'_1 \times A'_2 \times \dots \times A_i'^1 \times \dots \times A'_d$ and $\Omega_d'^2 = A'_1 \times A'_2 \times \dots \times A_i'^2 \times \dots \times A'_d$, where $A_i'^1 \cup A_i'^2 = A'_i$ and $A_i'^1 \cap A_i'^2 = \emptyset$. The i -th dimension is called the *split dimension*, and the pair $A_i'^1/A_i'^2$ is called the *dimension split (arrangement)* of the space split. A *partition* of a space (subspace) is a set of disjoint subspaces obtained from a sequence of space splits.

As discussed in [13,14], the Hamming distance is a suitable distance measure for NDDSs. The Hamming distance between two vectors gives the number of mismatching dimensions between them. A similarity (range) query is defined as follows: given a query vector α_q and a query range r_q of Hamming distance, find all the vectors whose Hamming distance to α_q is less than or equal to r_q .

2.2 NSP-Tree Structure

The NSP-tree [14] is designed based on the space-partitioning concepts. Let Ω_d be an NDDS. Each node in the NSP-tree represents a subspace from a partition of Ω_d , with the root node representing Ω_d . The subspace represented by a non-leaf node is divided into smaller subspaces for its child nodes via a sequence of (space) splits.

The NSP-tree has a disk-based balanced tree structure. A leaf node of the NSP-tree contains an array of entries of the form $(key, optr)$, where key is a vector in Ω_d and $optr$ is a pointer to the indexed object identified by key in the database. A non-leaf node contains the space-partitioning information, the pointers to its child nodes and their associated auxiliary bounding boxes (i.e., DMBRs). The space-partitioning information in a non-leaf node N is represented by an auxiliary tree called the Split History Tree (SHT). The SHT is an unbalanced binary tree. Each node of the SHT represents a split that has occurred in N . The order of all the splits that have occurred in N is represented by the hierarchy of the SHT, i.e., a parent node in the SHT represents a split that has occurred earlier than all the splits represented by its children. Each SHT tree node has four fields: i) *sp-dim*: the split dimension; ii) *sp-pos*: the dimension split arrangement; iii) *L-pntr* and *R-pntr*: pointers to an SHT child node (internal pointer) or a child node of N in the NSP-tree (external pointer). *L-pntr* points to the left side of the split, while *R-pntr* points to the right side. Using the SHT, the subspace for each child of N is determined. The pointers from N to all its children are, in fact, those external pointers of the SHT for N . Note that, from the definition, each SHT node SN also represents a subspace of the data space resulted from the splits represented by the SHT nodes from the root to SN (the root represents the subspace for N in the NSP-tree).

Figure 1 illustrates the structure of an NSP-tree. In the figure, a tree node is represented as a rectangle labeled with a number. Each non-leaf node contains an SHT. There are two DMBRs for each child. $DMBR_{ij}$ represents the j -th

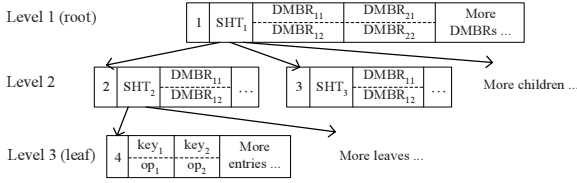


Fig. 1. Example of the NSP-tree

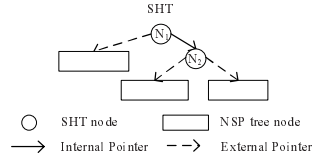


Fig. 2. Example of the SHT

($1 \leq j \leq 2$) DMBR for the i -th ($1 \leq i \leq M$) child at each node, where M is the maximum fan-out of the node, which is determined by the (disk) space capacity of the node. Figure 2 shows an example SHT. Each SHT node is represented as a circle. A solid pointer in the figure represents an internal pointer that points to an SHT child node, while a dotted pointer is an external pointer that points to a child of the relevant non-leaf node of the NSP-tree that contains the SHT.

3 Space-Partitioning-Based Bulk-Loading

3.1 Key Idea of the Algorithm

The key idea of our bulk-loading algorithm NSPBL is to first load all vectors into one (large) node and then keep splitting overflow nodes until an NSP-tree structure is eventually constructed in a bottom-up fashion. Instead of splitting a node by directly dividing the relevant data (vectors) set, NSPBL partitions the underlying space for the relevant data (vectors) set into subspaces and then places the relevant vectors into the subspaces (nodes) that they belong to.

To achieve a good target NSP-tree and reduce the I/O cost for bulk-loading, NSPBL adopts the following strategies: (i) *Partitioning current data space*. Instead of partitioning a whole NDDS, NSPBL partitions the current data space (see Section 2.1) for a given set of input vectors. This improves the target tree quality because partitioning subspaces that contain no input vectors is not only wasting but also making the target tree unnecessarily larger. (ii) *Utilizing histograms of letters appearing on each dimension for the input vectors*. One challenge for building an index tree using space partitioning is to make each partition as balanced as possible so that both space utilization and search performance of the resulting tree are high. NSPBL tackles this challenge by using the global data distribution information from the histograms to properly split a data space. (iii) *Seeking a balanced multi-way splitting*. The conventional two-way splitting is inefficient for bulk-loading. NSPBL adopts a multi-way splitting strategy to allow an overflow node (space) to be split into more than two new nodes (subspaces). In fact, a multi-way split is still determined by a series of two-way splits of the current data space for the overflow node. However, the splitting is propagated up to the parent node only once, rather than multiple times as required by the conventional two-way splitting. During the series of two-way splits, NSPBL always pick the subspace with the most vectors to split so that a balanced multi-way

split can be achieved in the end. (iv) *Adopting a memory buffer (page) for each new leaf node resulting from splitting.* When an overflow leaf node (space) is split, its vectors should be distributed into the new leaf nodes. Using a memory buffer for each new leaf node can significantly reduce the number of I/Os needed to write to the new leaf nodes.

NSPBL employs an intermediate disk-based tree structure (see Figure 3), called the Space-partitioning Bulk-Loading tree (SBL-tree). The structure of the SBL-tree is similar to that of the target NSP-tree (see Section 2.2): (1) the entry structures of a leaf node and a non-leaf node are the same as those in an NSP-tree, (2) the maximum number of entries for a non-leaf node is the same as that in an NSP-tree, i.e., M , and (3) each non-leaf node contains an SHT, which stores the space splitting history information for the node.

The SBL-tree differs from the NSP-tree by having two types of leaf nodes: normal leaves and buffered leaves. A normal leaf has at most M entries, like that in an NSP-tree. If a leaf node has more than M entries, it is a buffered (i.e., non-final) leaf. A buffered leaf overflows according to the target NSP-tree. Hence it needs to be split.

To facilitate the splitting, each buffered leaf node N maintains a histogram for each dimension D to record the frequencies (in percentage) of letters that appear on D in the indexed vectors in N . Once NSPBL turns all buffered leaves into normal leaves by space partitioning, the target NSP-tree can be obtained by outputting the final SBL-tree with corresponding DMBRs computed.

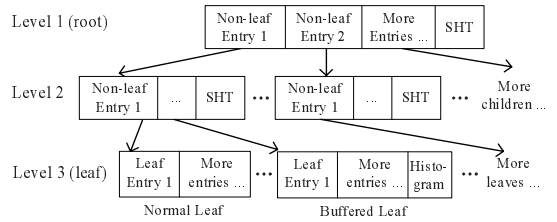


Fig. 3. Example of the SBL-tree

3.2 Main Procedure

The main procedure of algorithm NSPBL is given as follows. It invokes functions `SplitBufferedLeaf` and `SplitNonLeafNode` to repeatedly split overflow nodes until the target NSP-tree for the given set of input vectors is constructed.

ALGORITHM 3.1 : NSPBL

Input: a set SV of input vectors in a d -dimensional NDDSS.

Output: an NSP-tree $TgtTree$ for SV on disk.

Method:

1. create an SBL-tree BT with an empty leaf N as root;
2. load all vectors in SV into N and create relevant histograms for N ;
3. **if** size of $N \leq M$ **then** {
4. create target NSP-tree $TgtTree$ with single node N ; }
5. **else** {
6. let $Bleafset = \{N\}$;
7. **while** $Bleafset \neq \emptyset$ **do** {
8. fetch an overflow leaf $CN \in Bleafset$ and let $Bleafset = Bleafset - \{CN\}$;
9. $[rleafset, adoptSHT] = \text{SplitBufferedLeaf}(CN)$;
10. add leaves with size $> M$ in $rleafset$ into $Bleafset$;
11. calculate two DMBRs for every leaf with size $\leq M$ in $rleafset$;


```

12. if  $CN$  is not the root of  $BT$  then {
13.   replace entry for  $CN$  in its parent  $PN$  with entries for nodes in  $rleafset$ ;
14.   incorporate  $adoptSHT$  into  $PN$ ; }
15. else {
16.   create a non-leaf node  $PN$  as the new root;
17.   add entries for nodes in  $rleafset$  into  $PN$ ;
18.   add  $adoptSHT$  into  $PN$ ; }
19. if  $PN$  overflows then {
20.   [ $rnodeset$ ,  $adoptSHT$ ] = SplitNonleafNode( $PN$ );
21.   if  $PN$  is not the root of  $BT$  then {
22.     replace entry for  $PN$  in its parent  $P$  with entries for nodes from  $rnodeset$ ;
23.     incorporate  $adoptSHT$  into  $P$ ; }
24.   else {
25.     create a non-leaf node  $P$  as the new root;
26.     add entries for nodes in  $rnodeset$  into  $P$ ;
27.     add  $adoptSHT$  into  $P$ ; }
28.   propagate overflow/splitting up to the root of  $BT$  when needed; } }
29. create target NSP-tree  $TgtTree$  based on  $BT$ , and create DMBRs for non-leaf
   nodes in their corresponding entries in parents ; }
30. return  $TgtTree$ .

```

Algorithm NSPBL initially loads all input vectors into one leaf node (steps 1 - 2). Relevant histograms are computed during the loading (note that these histograms reflect the global data distribution of the whole data set). If this leaf is not oversized, the target NSP-tree has been obtained (steps 3 - 4). Otherwise, the oversized leaf node is put into a set that keeps track of buffered leaves during the bulk-loading process (step 6). For each buffered leaf, NSPBL invokes function SplitBufferedLeaf to split it into multiple new leaves using its histograms (steps 8 - 9). Some of the new leaves may be normal leaves whose two DMBRs are computed (step 11) and stored into their corresponding entries in their parents (steps 13 or 17). The others may still be buffered leaves that need further splitting (step 10). In fact, eventually every leaf will be a normal leaf node that can be directly copied into the target NSP-tree in step 29. NSPBL constructs the non-leaf nodes of the SBL-tree in a bottom-up fashion (steps 12 - 28). Unlike the TL algorithm for the NSP-tree, NSPBL does not require a top-down look-up phase (i.e., *ChooseSubtree*). It effectively uses the maintained histograms and the derived splitting history information (SHTs) to decide proper nodes (subspaces) for input vectors. In addition, it adopts a multi-way splitting rather than a conventional two-way splitting. Hence NSPBL is expected to be more efficient than the conventional TL approach. When an SHT $adoptSHT$ for a set of new nodes resulting from a split is incorporated into an existing non-leaf node PN or P (steps 14 and 23), NSPBL replaces the external pointer of the current SHT of existing PN or P that points to the original node before splitting (i.e., CN or PN) with an internal pointer that points to the root of $adoptSHT$. For a new root, $adoptSHT$ is simply added to it as its SHT (steps 18 and 27). The target NSP-tree is obtained when no buffered leaf exists (steps 4 and 29). The two DMBRs for leaf and non-leaf nodes are computed at different times (steps 11 and 29) to reduce the bulk-loading I/Os. The algorithms used for computing the DMBRS are the same as those for the NSP-tree[14].

3.3 Splitting a Buffered Leaf Node

Function `SplitBufferedLeaf` splits a buffered leaf node CN into multiple new leaf nodes. It does this by repeatedly invoking function `SplitSpace` to split the underlying data space for CN into subspaces. The splitting decisions are recorded in an SHT. At the end of the function, the vectors in CN are loaded into the resulting subspaces (leaf nodes) and the histograms for each node are updated.

To achieve a balanced space partition for the buffered leaf node N , in the process of multi-way splitting, `SplitBufferedLeaf` always pick the subspace with the most vectors to split. Since it is too expensive to actually count the vectors in subspaces, `SplitBufferedLeaf` uses the maintained histograms for N to estimate the number of vectors in a subspace. Such a heuristic assumes that the dimensions are mutually independent for the given data set.

As mentioned earlier, NSPBL associates a memory buffer (page) to each new leaf node (subspace). If there are B pages of memory space available, we cannot split the given data space into more than B subspaces. In addition, it is unnecessary to split a leaf node with $\leq M$ vectors. Hence, the number of subspaces resulted from multi-way splitting is bound by B and the number of subspaces/leaves with $\leq M$ vectors.

FUNCTION 3.1 : `SplitBufferedLeaf`

Input: (1) a buffered leaf node N containing a set of input vectors in a d -dimensional NDDS; (2) the number B of memory buffer pages.

Output: (1) a set NS of new leaf nodes; (2) the corresponding SHT T for the adopted split.

Method:

1. create the current data space Ω' based on the histograms for N ;
2. create an empty priority queue $PQueue$ of SHT node pointers;
3. create a pseudo SHT node pointer P that corresponds to Ω' ;
4. insert P into $PQueue$ with priority key $|N|$ (i.e., size of N);
5. set the number of used memory buffers $cur\_buf\_cnt = 0$;
6. **while** $PQueue$ is not empty **do** {
7. dequeue the next SHT pointer CP from $PQueue$ with priority key value vec_cnt ;
8. **if** $cur\_buf\_cnt \geq B$ **then** {
9. set CP as an external pointer; }
10. **else** {
11. **if** CP is the pseudo SHT node pointer P **then** {
12. $[dim, lset, rset, lratio, rratio] = \text{SplitSpace}(\Omega', \text{histograms for } N)$;
13. create a new SHT node SHT_N with split information $dim, lset$ and $rset$;
14. construct SHT T with SHT_N as its root; }
15. **else** {
16. construct subspace DS based on both Ω' and the space split information in the SHT T from its root to CP ;
17. $[dim, lset, rset, lratio, rratio] = \text{SplitSpace}(DS, \text{histograms for } N)$;
18. create a new SHT node SHT_N with split information $dim, lset$ and $rset$;
19. set CP as an internal pointer that points to SHT_N ; }
20. $lcnt, rcnt = vec\_cnt * lratio, rratio$;
21. **if** $lcnt, rcnt > M * \text{SPLIT\_RATIO}$ **then** {
22. enqueue $SHT\_N.l\_pntr, r\_pntr$ with priority key $lcnt, rcnt$; }
23. **else** { set $SHT\_N.l\_pntr, r\_pntr$ as an external pointer; }
24. cur_buf_cnt++ ; }

25. create a set NS of cur_buf_cnt new SBL-tree leaf nodes to which the external pointers of T point;
26. load each vector in N into one N' of the new leaf nodes in NS according to the subspaces given by T and update the histograms for N' accordingly;
27. **return** NS and $\hat{T}$.

Function `SplitBufferedLeaf` first uses the histograms for N to determine the current data space Ω' represented by N (step 1). The letters in the D -th component set of Ω' are those letters with frequencies > 0 in the histogram for dimension D . A priority queue $PQueue$ of SHT node pointers are maintained by `SplitBufferedLeaf` (step 2). The SHT node pointers in $PQueue$ are those pointers in the SHT T returned at the end of the function. Each pointer corresponds to a subspace of Ω' (step 16) and the estimated vector count in that subspace is used as the priority key for $PQueue$. This allows `SplitBufferedLeaf` to always pick the subspace with the largest estimated number of vectors to split (step 7) so that a balanced space partition can be achieved. The multi-way splitting process starts from Ω' , which is represented by a pseudo SHT node pointer P (steps 3 - 4). The process does not stop until all the pointers in $PQueue$ are exhausted (step 6). When there is no memory buffer page left (step 8), space-splitting stops. The remaining pointers in $PQueue$ are all set to be external pointers (step 9). Otherwise, function `SplitSpace` is invoked to split the subspace corresponding to the current pointer CP , and the SHT T is constructed and grown (steps 11 - 19). Steps 20 - 23 estimate the vector counts in the two new subspaces resulted from the space split. Depending on the estimated vector count in a subspace, the corresponding SHT node pointer is either enqueued or set to be an external pointer (no more split on that subspace). `SplitBufferedLeaf` terminates when all buffer pages are used up or there is no subspace to split. New leaf nodes for the subspaces are then created based on the SHT T , and loaded (steps 25 - 26).

`SplitBufferedLeaf` uses an additional adjustable parameter $SPLIT\_RATIO (\geq 1)$ to further control whether a subspace should be split or not (step 21). Obviously, the greater the value of $SPLIT\_RATIO$, the more bulk-loading I/Os are needed, since `SplitBufferedLeaf` will potentially produce less subspaces. The benefit of a greater $SPLIT\_RATIO$ value is that the space utilization of the target tree may be improved, since more vectors may be fit in a subspace. Our experimental results show that for uniform data sets, different $SPLIT\_RATIO$ values make no much difference in the space utilization of the target tree. Thus, a $SPLIT\_RATIO$ value of 1 can be used for uniform data. On the other hand, when the data set for bulk-loading is skewed, that is, the frequencies of different letters in the alphabet are quite different, a greater $SPLIT\_RATIO$ value may result in a target tree with a better space utilization.

Function `SplitSpace` splits a given subspace into two subspaces and returns the split information. Two heuristics are adopted when choosing the split dimension: **H1**: choose the dimension with a larger span, i.e., more distinct letters appearing in contained vectors; **H2**: choose the dimension that has a more balanced split. Histograms are used to support the two heuristics.

FUNCTION 3.2 : SplitSpace**Input:** (1) a d -dimensional subspace DS ; (2) histograms H for all the dimensions.**Output:** space split information: (1) dim ; (2) $lset$; (3) $rset$; (4) $lratio$; (5) $rratio$.**Method:**

1. $max\_span = \max\{spans \text{ of all dimensions in } DS\}$;
2. dim_set = the set of dimensions with max_span ;
3. $best\_balance = 0$;
4. **for** each dimension D in dim_set **do** {
5. sort into list L_0 in descending order the letters on the D -th dimension of DS based on their frequencies recorded in H for dimension D ;
6. set lists L_1, L_2 to empty, $weight_1 = weight_2 = 0$;
7. **for** each letter l in L_0 **do** {
8. **if** $weight_1 \leq weight_2$ **then** {
9. $weight_1 = weight_1 + \text{frequency of } l \text{ on } D\text{th dimension}$;
10. add l to the end of L_1 ; }
11. **else** {
12. $weight_2 = weight_2 + \text{frequency of } l \text{ on } D\text{th dimension}$;
13. add l to the beginning of L_2 ; }
14. concatenate L_1 and L_2 into L_3 ;
15. **for** $j = 2$ to $|L_3|$ **do** {
16. $set_1 = \{\text{letters in } L_3 \text{ whose position} < j\}$;
17. $set_2 = \{\text{letters in } L_3 \text{ whose position} \geq j\}$;
18. $f_i = \text{sum of letter frequencies in } set_i \text{ for } i = 1, 2$;
19. **if** $f_1 \leq f_2$ **then** { $current\_balance = f_1/f_2$; }
20. **else** { $current\_balance = f_2/f_1$; }
21. **if** $current\_balance > best\_balance$ **then** {
22. $best\_balance = current\_balance$;
23. $dim = D, lset = set_1, rset = set_2; lratio, rratio = f_1, f_2/(f_1 + f_2)$; }
24. **return** $dim, lset, rset, lratio, rratio$.

Function SplitSpace first picks those dimensions with the maximum span (steps 1 - 2). For each such dimension D , it first sorts the letters appearing on D based on their frequencies into a “U”-shaped list, i.e., letters with higher frequencies are placed at two ends (steps 5 - 14). It then finds the most balanced dimension split (steps 15 - 23).

3.4 Splitting a Non-leaf Node

Function SplitNonleafNode splits a non-leaf node N into several new non-leaf nodes. The main idea is to break the SHT of N into several subtrees so that the number of external pointers (to SBL-tree nodes) under each subtree is $\leq M$. The function then splits N into new non-leaf nodes according to the subtrees of the SHT. Let $ext\_set(SN)$ denote the set of external pointers in a subtree rooted at node SN from an SHT in the following description.

FUNCTION 3.3 : SplitNonLeafNode**Input:** an overflow non-leaf node N of an SBL-tree BT in a d -dimensional NDDSS.**Output:** a set SS of new normal non-leaf nodes and the corresponding SHT $N.SHT$ for the adopted split.**Method:**

1. let $S = \{ SN \mid SN \text{ is an SHT node in } N.SHT \text{ and } |ext\_set(SN)| \leq M \text{ and } |ext\_set(SN.parent)| > M \}$;
2. **for** each SN in S **do** {
3. create a new non-leaf node N' for given SBL-tree BT ;
4. move subtree ST' rooted at SN from $N.SHT$ into N' ;

5. change the internal pointer pointing to ST' in $SN.parent$ of $N.SHT$ to the external pointer pointing to N' ;
6. move those entries in N that correspond to external pointers in $ext\_set(ST')$ to N' and create a new entry for N' in N ;
7. add N' into the set SS of new non-leaf nodes for BT ; }
8. **return** SS and the updated $N.SHT$.

Function `SplitNonleafNode` essentially uses the split history information in the SHT of the given overflow non-leaf node N to find a set of subspaces that contain as many child nodes as possible without overflowing (step 1). It then creates a new non-leaf node for each subspace, links them to the SBL-tree, and adjusts the SHT and relevant entries in the original N (steps 2 - 7).

4 Experimental Results

To evaluate NSPBL, we conducted extensive experiments. Typical results from the experiments are reported in this section.

Our experiments were conducted on a PC with Pentium D 3.40GHz CPU, 2GB memory and 400 GB hard disk. Performance evaluation was based on the number of disk I/Os with the disk block size set at 4 kilobytes. The available memory sizes used in the experiments were simulated based on the program configurations rather than real physical RAM changes in hardware. The data sets used in the presented experimental results included both real genome sequence data and synthetic data. Genomic data (*geno*) was extracted from bacteria genome sequences of the GenBank [8], which were broken into q-grams/vectors of 25 characters long (i.e., 25 dimensions). Two synthetic data sets were generated using the Zipf distribution [18] with parameter values of 0 (*zipf0* – uniform) and 3 (*zipf3* – very skewed), both of which were 40 dimensional and had an alphabet size of 10 on all dimensions. For comparison purposes, we also implemented both the conventional TL algorithm (*TL*) of the NSP-tree [14] and the representative generic bulk-loading algorithm GBLA [3]. All programs were implemented in C++. According to [3], we set the size (disk block count) of the external buffer (pages on disk) of each index node of the buffer-tree in GBLA at half of the node fan-out, which was decided by the available memory size.

4.1 Effect of Adjustable Parameter

Function `SplitBufferedLeaf` uses an adjustable parameter *SPLIT_RATIO* to provide an additional control on whether a subspace should be split or not. The number of bulk-loading I/Os and the space utilization of the bulk-loaded NSP-trees by NSPBL for different *SPLIT_RATIO* values are presented in Table 1. From the table, we can see that greater *SPLIT_RATIO* values always result in more bulk-loading I/Os. For uniform data (*zipf0*), different *SPLIT_RATIO* values yield almost the same space utilization for the bulk-loaded NSP-trees. For very skewed data (*zipf3*), increasing the value of *SPLIT_RATIO* significantly improves of the space utilization. Genomic data, which is much less skewed than *zipf3*, has a behavior more similar to that of *zip0* than that of *zipf3*. Based on

Table 1. Effect of different *SPLIT_RATIO* values

<i>SPLIT_RATIO</i>	<i>zipf0</i>		<i>zipf3</i>		<i>geno</i>	
	<i>io</i>	<i>ut%</i>	<i>io</i>	<i>ut%</i>	<i>io</i>	<i>ut%</i>
1	418,620	65.7	612,189	44.2	279,694	75.7
2	769,079	65.8	765,348	60.1	431,425	80.5
4	976,624	65.7	969,844	65.8	529,228	82.8
6	1,112,688	65.7	1,113,127	64.2	533,578	82.7

the result, we can conclude that, for data with a distribution close to the uniform one, a *SPLIT_RATIO* value of 1 is acceptable; for skewed data sets, although a greater *SPLIT_RATIO* value causes more bulk-loading I/Os, the benefit of a high quality target tree is overwhelming. For the experimental results presented in the following subsections, we used a *SPLIT_RATIO* value of 1 for *geno* and *zipf0* data, and a *SPLIT_RATIO* value of 2 for *zipf3* data.

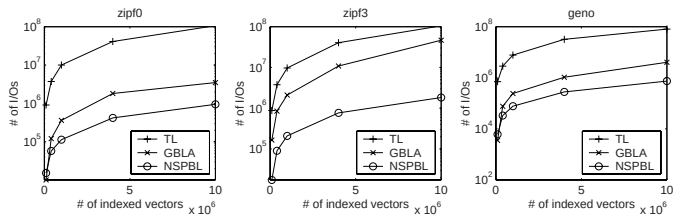
4.2 Efficiency Evaluation

Figure 4 (logarithmic scale in base 10 for Y-axis) shows the number of I/Os needed to construct NSP-trees using the TL algorithm, GBLA, and NSPBL for synthetic and genomic data sets of different sizes. The memory available for the algorithms was set to 4 megabytes. From the figure, we can see that NSPBL significantly outperformed the conventional TL algorithm. On average, NSPBL was about 80 times faster than the TL algorithm in our experiments. For uniform synthetic data and genomic data, when the database size was small, GBLA was more efficient than NSPBL. This is because GBLA employs an in memory buffer-tree and can almost build the entire NSP-tree in memory in such a case, while NSPBL only uses memory as I/O buffers. As the database size became much larger than the available memory size, NSPBL was much more efficient than GBLA. For example, on average, NSPBL was 5.4 times faster than GBLA when bulk-loading 10 million genomic vectors. NSPBL is particularly efficient in bulk-loading the very skewed data set *zipf3*. This shows that the strategies adopted by NSPBL, such as balanced multi-way splitting, were effective.

Experiments were

also conducted to

study the effect of different memory sizes on the performance of GBLA and NSPBL. Table 2 shows the number of I/Os needed by these two algorithms to construct the NSP-

**Fig. 4.** Bulk-loading performance comparison

trees for 4 million vectors of synthetic and genomic data under different sizes of available memory. From the table, we can see that NSPBL was always more efficient than GBLA. When the memory size was small comparing to the database size, the performance of NSPBL was significantly better than that of GBLA.

Table 2. Effect of memory size on bulk-loading performance

Memory	zipf0		zipf3		geno	
	¹⁰ GBLA	¹⁰ NSPBL	¹⁰ GBLA	¹⁰ NSPBL	¹⁰ GBLA	¹⁰ NSPBL
64KB	4,723,892	626,686	18,365,289	1,077,452	3,906,983	399,954
4MB	1,819,533	418,620	10,862,459	765,348	1,046,984	279,694
256MB	659,238	322,019	7,382,932	641,957	375,590	216,599

On the other hand, when the memory was very large so that almost the entire NSP-tree could be fit in it, the performance difference between the two algorithms became smaller. In real applications such as genome sequence searching, since the available memory size is usually small comparing to the huge database size, NSPBL has a significant performance benefit. In other words, for a fixed memory size, the larger the database size is, the more performance benefit the NSPBL can provide. This can also be observed in Figure 4.

4.3 Quality Evaluation

To evaluate the effectiveness of NSPBL, we compared the quality of the NSP-trees constructed by all algorithms. The quality of an NSP-tree was measured by its query performance and space utilization. Table 3 shows the query performance of the NSP-trees constructed by the TL algorithm, GBLA, and NSPBL for synthetic and genomic data. These trees are the same as those presented in Figure 4. Query performance was measured based on the average number of I/Os for executing 100 random range queries at Hamming distance 3. The results show that for uniform synthetic data and genomic data, the NSP-trees constructed by NSPBL have comparable performance as those constructed by the TL algorithm and GBLA. For very skewed data, the performance of the NSP-trees from NSPBL is much better. This shows the advantage of applying histograms in NSPBL, which are capable of capturing global data distribution information, resulting a better tree structure. On the other hand, both the TL algorithm and GBLA only partition the space/data based on vectors already indexed in their structures, which may not accurately reflect the actual global distribution of the whole data set.

Table 4 shows the space utilization of the same set of NSP-trees for synthetic and genomic data. From the table, we can see that the space utilization of those NSP-trees constructed by NSPBL varied more than that of the NSP-trees from the other algorithms. This is because, as a space-partitioning-based approach, NSPBL does not guarantee the minimum space utilization. However, the space utilization of those NSP-trees was reasonably good due to the heuristics employed by NSPBL to find a balanced split in the bulk-loading process. The result for *zipf3* was even better than those for TL and GBLA.

Besides the experiments reported above, we have also conducted experiments with data sets of various alphabet sizes and dimensionalities. The results were similar. Due to the space limitation, they are not included in this paper.

Table 3. Query performance comparison

<i>key#</i>	<i>zipf0</i>			<i>zipf3</i>			<i>geno</i>		
	<i>io</i> <i>TL</i>	<i>io</i> <i>GBLA</i>	<i>io</i> <i>NSPBL</i>	<i>io</i> <i>TL</i>	<i>io</i> <i>GBLA</i>	<i>io</i> <i>NSPBL</i>	<i>io</i> <i>TL</i>	<i>io</i> <i>GBLA</i>	<i>io</i> <i>NSPBL</i>
100000	136	144	148	605	596	433	192	192	207
400000	269	238	246	1,534	1,497	1,019	344	342	344
1000000	400	400	400	2,718	2,704	1,542	476	468	518
4000000	679	680	680	6,010	6,007	2,502	771	766	782
10000000	1,053	1,032	1,039	9,704	9,611	3,489	1,046	1,048	1,104

Table 4. Space utilization comparison

<i>key#</i>	<i>zipf0</i>			<i>zipf3</i>			<i>geno</i>		
	<i>ut%</i> <i>TL</i>	<i>ut%</i> <i>GBLA</i>	<i>ut%</i> <i>NSPBL</i>	<i>ut%</i> <i>TL</i>	<i>ut%</i> <i>GBLA</i>	<i>ut%</i> <i>NSPBL</i>	<i>ut%</i> <i>TL</i>	<i>ut%</i> <i>GBLA</i>	<i>ut%</i> <i>NSPBL</i>
100000	59.5	63.4	63.7	54.0	52.6	60.6	69.5	69.7	63.2
400000	60.4	58.0	56.1	53.6	52.4	62.4	69.5	70.2	69.3
1000000	65.7	65.7	65.7	54.2	53.9	60.9	78.8	79.6	66.6
4000000	65.7	65.7	65.7	54.4	54.0	60.1	76.1	76.5	75.7
10000000	81.8	81.1	80.0	54.6	54.1	60.7	65.8	65.9	55.8

5 Conclusions

There is an increasing demand for applications such as genome sequence searching that involve similarity queries on large data sets in NDDSSs. Index structures such as the NSP-tree [14] are crucial to achieving efficient evaluation of similarity queries in NDDSSs. Although many bulk-loading techniques have been proposed to construct index trees in CDSs in the literature, no bulk-loading technique has been developed specifically for the NSP-tree in NDDSSs.

In this paper, we present a space-partitioning-based algorithm NSPBL to bulk-load the NSP-tree for large data sets in NDDSSs. NSPBL constructs a target NSP-tree by repeatedly partitioning the underlying space of the data set rather than partitioning the data set directly. To avoid accessing individual input vectors, it partitions the space based on the histograms for component letters of the vectors. To achieve better efficiency and effectiveness of bulk-loading, NSPBL adopts several strategies including partitioning the current space rather than the whole space, splitting an overflow node into multiple nodes rather than always two nodes, applying effective heuristics to choose balanced space splits, and associating a buffer page to each leaf node.

Our experimental results demonstrate that NSPBL significantly outperforms the conventional TL method and the popular generic bulk-loading algorithm GBLA [3], especially when being used for large data sets, for skewed data sets, and with limited available memory. The target NSP-trees obtained from all the algorithms have comparable searching performance and space utilization.

References

1. Arge, L., Berg, M., Haverkort, H., Yi, K.: The Priority R-tree: a practically efficient and worst-case optimal R-tree. In: Proc. of SIGMOD, pp. 347–358 (2004)
2. Berchtold, S., Bohm, C., Kriegel, H.-P.: Improving the query performance of high-dimensional index structures by bulk-load operations. In: Proc. of EDBT, pp. 216–230 (1998)

3. Bercken, J., Seeger, B., Widmayer, P.: A generic approach to bulk loading multi-dimensional index structures. In: Proc. of VLDB, pp. 406–415 (1997)
4. Bercken, J., Seeger, B.: An evaluation of generic bulk loading techniques. In: Proc. of VLDB, pp. 461–470 (2001)
5. Ciaccia, P., Patella, M.: Bulk loading the M-tree. In: Proc. of the 9th Australian Database Conference, pp. 15–26 (1998)
6. DeWitt, D., Kabra, N., Luo, J., Patel, J., Yu, J.: Client-server paradise. In: Proc. of VLDB, pp. 558–569 (1994)
7. Garcia, Y., Lopez, M., Leutenegger, S.: A greedy algorithm for bulk loading R-trees. In: Proc. of ACM-GIS, pp. 2–7 (1998)
8. <http://www.ncbi.nlm.nih.gov/Genbank/>
9. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proc. of SIGMOD, pp. 47–57 (1984)
10. Jermaine, C., Datta, A., Omiecinski, E.: A novel index supporting high volume data warehouse insertion. In: Proc. of VLDB, pp. 235–246 (1999)
11. Kamel, I., Faloutsos, C.: On packing R-trees. In: Proc. of CIKM, pp. 490–499 (1993)
12. Leutenegger, S., Edgington, J., Lopez, M.: STR: A Simple and Efficient Algorithm for R-Tree Packing. In: Proc. of ICDE, pp. 497–506 (1997)
13. Qian, G., Zhu, Q., Xue, Q., Pramanik, S.: The ND-tree: a dynamic indexing technique for multidimensional non-ordered discrete data spaces. In: Proc. of VLDB, pp. 620–631 (2003)
14. Qian, G., Zhu, Q., Xue, Q., Pramanik, S.: A space-partitioning-based indexing method for multidimensional non-ordered discrete data spaces. ACM TOIS 23, 79–110 (2006)
15. Qian, G., Zhu, Q., Xue, Q., Pramanik, S.: Dynamic indexing for multidimensional non-ordered discrete data spaces using a data-partitioning approach. ACM TODS 31, 439–484 (2006)
16. Roussopoulos, N., Leifker, D.: Direct spatial search on pictorial databases using packed R-trees. In: Proc. of SIGMOD, pp. 17–31 (1985)
17. Seok, H.-J., Qian, G., Zhu, Q., Pramanik, S.: Bulk-loading the ND-tree in non-ordered discrete data spaces. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DAS-FAA 2008. LNCS, vol. 4947, pp. 156–171. Springer, Heidelberg (2008)
18. Zipf, G.K.: Human behavior and the principle of least effort. Addison-Wesley, Reading (1949)

Efficient Updates for Continuous Skyline Computations*

Yu-Ling Hsueh¹, Roger Zimmermann², and Wei-Shinn Ku³

¹ Dept. of Computer Science, University of Southern California,
Los Angeles, CA 90089

² Computer Science Department, National University of Singapore, Singapore 117543

³ Dept. of Computer Science and Software Engineering, Auburn University,
Auburn, AL 36849

hsueh@usc.edu, rogerz@comp.nus.edu.sg, weishinn@auburn.edu

Abstract. We address the problem of maintaining *continuous skyline queries* efficiently over dynamic objects with d dimensions. Skyline queries are an important new search capability for multi-dimensional databases. In contrast to most of the prior work, we focus on the unresolved issue of frequent data object updates. In this paper we propose the *ESC* algorithm, an Efficient update approach for Skyline Computations, which creates a pre-computed *second skyline* set that facilitates an efficient and incremental skyline update strategy and results in a quicker response time. With the knowledge of the *second skyline* set, *ESC* enables (1) to efficiently find the substitute skyline points from the *second skyline* set only when removing or updating a skyline point (which we call a first skyline point) and (2) to delegate the most time-consuming skyline update computation to another independent procedure, which is executed after the complete updated query result is reported. We leverage the basic idea of the traditional *BBS* skyline algorithm for our novel design of a two-threaded approach. The first skyline can be replenished quickly from a small set of second skylines - hence enabling a fast query response time - while de-coupling the computationally complex maintenance of the second skyline. Furthermore, we propose the *Approximate Exclusive Data Region* algorithm (*AEDR*) to reduce the computational complexity of determining a candidate set for second skyline updates. In this paper, we evaluate the *ESC* algorithm through rigorous simulations and compare it with existing techniques. We present experimental results to demonstrate the performance and utility of our novel approach.

1 Introduction

Skyline query computations are important for multi-criteria decision making applications and they have been studied intensively in the context of spatio-temporal

* This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0534761, NUS AcRF grant WBS R-252-050-280-101/133 and equipment gifts from the Intel Corporation, Hewlett-Packard, Sun Microsystems and Raptor Networks Technology. We also acknowledge the support of the NUS Interactive and Digital Media Institute (IDMI).

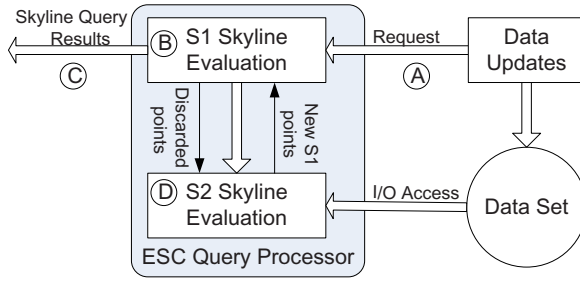


Fig. 1. ESC system framework

databases. Skyline queries have been defined as retrieving a set of points, which are not dominated by any other points. An object p dominates p' , if p has more favorable values than p' in all dimensions. Some of the prior work on skyline queries assumed that data objects are static [13,15]. Other approaches assumed that the skyline computation involved only a partial of dynamic dimensions [4]. In this paper, we address *Efficient Updates for Continuous Skyline Computations* over dynamic objects (*ESC* for short), where objects with d dynamic dimensions move in an unrestricted manner. Each dimension represents a spatial or non-spatial value. Towards an efficient continuous skyline computation the following challenges must be addressed: an effective incremental skyline query result update mechanism that is needed provides a fast response time of reporting the current query results, and an efficient strategy to reduce the search space dimensionality is required.

Existing work [6,14,19] generally computes a number of data point subsets, each of which is exclusively dominated by one skyline point. Therefore, when a skyline point moves or is deleted, only its exclusively dominated subset must be scanned. The determination of such an exclusive data set is very computationally complex in higher dimensions and it incurs a serious burden for the system in a highly dynamic environment. Therefore, these systems are often unable to provide up-to-date query results with a quick response time. We propose the *ESC* algorithm to efficiently manage the query results by delegating the time-consuming skyline update computations to another independent procedure, which is processed after the query processor reports the latest skyline query results. The key idea is to maintain a *second skyline* (or *S2*) set which is a skyline candidate set pre-computed when a traditional skyline (which we refer as the *first skyline*, *S1*) point requests an update. With the knowledge of the second skyline set, the skyline query result can be updated within a limited search space and the expensive computations (e.g., searching for new second skylines to substitute a promoted second skyline point) can be decoupled from the first skyline update computations.

Figure 1 shows the framework of the *ESC* system. The query processor initially computes the first and second skyline points. Any updates (A) performed on the data set are also submitted to the query processor. First, Task (B) examines whether the update request (e.g., inserting or removing a data point)

affects the first skyline set. If the request point becomes a new $S1$ point, Task **(B)** inserts the new $S1$ point into the current $S1$ set and removes the current skyline points that are dominated by the new $S1$ point. These discarded $S1$ points (new $S2$ points) are processed by Task **(D)** later to update the $S2$ set. In case that an update request stems from a removed or moving $S1$ point, some exclusive points are left un-dominated. The query processor searches for new substitute $S1$ points only from the $S2$ set. The query results **(C)** are immediately output as soon as Task **(B)** is completed. The processing time of the sequence of Tasks **(A)(B)(C)** is the system response time to a skyline query update. Task **(D)** maintains the $S2$ points when any $S2$ point is inserted or removed. To enhance Task **(D)**, which involves the expensive computation of determining exclusive data points where **(D)** searches for new or substitute $S2$ points from the rest of the data set, we also propose an *approximate exclusive data region* computation with lower amortized cost than existing techniques [14,19]. The remainder of this paper is organized as follows. Section 2 describes the related work. Section 3 presents and details our continuous skyline query processing design. We extensively verify the performance of our technique in Section 4 and finally conclude with Section 5.

2 Related Work

Borzsonyi et al. [1] proposed the straightforward non-progressive *Block-Nested-Loop* (BNL) and *Divide-and-Conquer* (DC) algorithms. The BNL approach recursively compares each data point with the current set of candidate skyline points, which might be dominated later. BNL does not require data indexing and sorting. The DC approach divides the search space and evaluates the skyline points from its sub-regions, respectively, followed by merge operations to evaluate the final skyline points. Both algorithms may incur many iterations and are inadequate for on-line processing. Tan et al. [17] presented two progressive processing algorithms: the *bitmap* approach and the *index* method. *Bitmap* encodes dimensional values of data points into bit strings to speed up the dominance comparisons. The index method classifies a set of d -dimensional points into d lists, which are sorted in increasing order of the minimum coordinate. The index scans the lists synchronously from the first entry to the last one. With the pruning strategies, the search space is reduced. The *nearest neighbor* (NN) method [5] indexes the data set with an R-tree. NN utilizes nearest neighbor queries to find the skyline results. The approach repeats the query-and-divide procedure and inserts the new partitions that are not dominated by some skyline point into a to-do list. The algorithm terminates when the to-do-list is empty. The *branch and bound skyline* (BBS) algorithm [13] traverses an R-tree to find the skyline points. Although BBS outperforms the NN approach, the performance can deteriorate due to many unnecessary dominance checks.

Finally, many of the recent techniques aim at continuous skyline support for moving objects and data streams. Lin et al. [8] present n -of- N skyline queries against the most recent n of N elements to support on-line computation against sliding windows over a rapid data stream. Morse et al. [11] propose a scalable *LookOut* algorithm for updating the continuous time-interval skyline efficiently. Sharifzadeh et al. [16] introduce the concept of Spatial Skyline Queries (SSQ). Given a set of data points P and a set of query points Q , SSQ retrieves those points of P which are not dominated by any other point in P considering their derived spatial attributes with respect to query points in Q . For moving query points, a continuous skyline query processing strategy is presented in [4] with a kinetic-based data structure. However, prompt query response is not considered in the design. A suite of novel skyline algorithms based on a Z-order curve [3] is proposed in [6]. Among the solutions, *ZUpdate* facilitates incremental skyline result maintenance by utilizing the properties of a Z-order curve. Other related techniques can be found in the literature [2,9,12,18,19]. However, all the aforementioned studies differ from the main goal of this research which is to support frequent skyline data object updates efficiently while providing a quick response.

3 ESC Algorithm

3.1 The Problem Definition of Continuous Skyline Queries

The formal definition of skyline points in d -dimensional space is a distinct object set P , where any two objects $p = (x_1, \dots, x_d)$ and $q = (y_1, \dots, y_d)$ in the set satisfy the condition that if for any k , $x_k < y_k$, there exists at least one dimension of $m \leq d$ that satisfies $x_m > y_m$. We say p dominates q ($p \vdash q$ for short), iff $x_k < y_k$, $\forall k$ ($1 \leq k \leq d$). The general setup of the problem consists of a set of dynamic query and data objects with d dimensions. Moving objects can freely maneuver

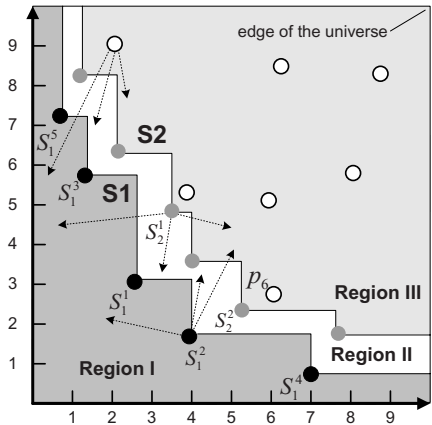


Fig. 2. S_1 and S_2 sets

in an unrestricted and unpredictable fashion, meaning that their parameters x_k may arbitrarily change their values. The major challenging issue of a continuous skyline query is to avoid unnecessary dominance checking on irrelevant data points for skyline query result updates. After observing the *BBS* algorithm [13], we deduced that when evaluating the skyline query result, a set of *second skyline* ($S2$) points can always be obtained with little extra work while retrieving the *first skyline* ($S1$) points. We refer to the traditional skyline query result as the *first skyline*, consisting of $S1 = \{s_1^1, \dots, s_1^m\}$. The *second skyline* $S2 = \{s_2^1, \dots, s_2^k\}$ is defined as follows:

Definition 1. A data point p is a second skyline point *iff* $p \in (P - S1)$ and $\nexists p' \in (P - S1 - p)$, $p' \vdash p$. Informally, all $S2$ points are dominated by $S1$ and the rest of the data points ($P - S1 - S2$) are dominated by both $S1$ and $S2$.

When a $S1$ point s_1^i is removed or at least one value of its dimensions changes, the $S2$ points are naturally considered as new $S1$ point candidates to “substitute” s_1^i . The features of a $S2$ set are as follows: (1) it is a pre-computed set that covers all the new $S1$ candidate points, and (2) $S2$ is a relatively small data set. Therefore, with the knowledge of $S2$, the query processor can efficiently update the query result and provide a quicker response time to the query requester. An example is shown in Figure 2. If the $S1$ point s_1^2 moves to Region *I*, the search space for *ESC* to update the query result only involves the $S1$ and the $S2$ sets. In this case, s_1^2 remains a $S1$ point, but it dominates s_1^1 . *ESC* needs to remove s_1^1 from the $S1$ set and s_1^1 becomes a new $S2$ point, since no existing $S2$ point can dominate it. Due to the movement of s_1^2 , *ESC* searches for new $S1$ points from the $S2$ set. Since s_2^2 (an exclusive data point) is left un-dominated, s_2^2 becomes a new $S1$ point and is removed from the $S2$ set. The *ESC* algorithm delegates the necessary $S2$ maintenance (an independent procedure from $S1$ updates) to the query processor after $S1$ updates are completed. For example, new $S2$ points must be retrieved to substitute s_2^2 . To avoid scanning through the entire data points in Region *III* for new $S2$ points, we propose an *approximate exclusive data region* (*AEDR*) computation in contrast to a traditional *exclusive data region* (*EDR*) computation. Based on our observation and analysis, we provide the lemmas for incrementally updating the skyline query results in the following sections. Table 1 summarizes the symbols and functions we use throughout the following sections.

Table 1. Symbols and functions

Symbols	Descriptions
P	Number of data objects
d	Number of dimension
$S1$	First skyline point set (traditional skyline query result set)
$S2$	Second skyline point set
$DataRtree$	Disk-based Rtree for indexing P
$S1Rtree$	Main-memory Rtree for indexing $S1$ points
$S2Rtree$	Main-memory Rtree for indexing $S2$ points
$EDR(p)$	A set of data points in the exclusive data region
$AEDR(p)$	A set of data points in the approximate exclusive data region
$W(p)$	A set of skyline points in the dominance area of p
$p.DomArea$	The dominance area of p

3.2 Second Skyline Computation

The existing work [14,19] performs time-consuming exclusive data point computations for the skyline query result updates. In Figure 3, the gray areas represent the traditional *EDRs* that contain exclusive data points. An *EDR* is not usually pre-computed because of the complexity of the calculation. In contrast, since the *S2* points (new *S1* candidates) can be easily computed before any *S1* point issues an update, the query processor is able to satisfy a query request with the latest query result and with a quicker response time. To further reduce the search space of visiting *S2* points to update the skyline query result, we introduce and define a *dominance set* for each *S1* point s_1^i . A *dominance set* contains a group of *S2* points which are dominated by s_1^i (denoted by $D(s_1^i)$) to substitute a removed or moving s_1^i point when the dominance relationship has changed. For example in Figure 3 the dominance set of s_1^2 includes s_2^2 . If s_1^2 is removed, *ESC* only checks the *S2* points in $D(s_1^2)$, instead of the entire *S2* points. In this example, s_2^2 becomes a new *S1* point, so it is removed from *S2*. We formally define a *dominance set* and establish Lemma 1 which states that a dominance set must contain all the necessary *S1* candidate points as follows:

Definition 2. (*Dominance Set: $D(s_1^i)$*)

A *dominance set* of a skyline point s_1^i (denoted by $D(s_1^i) = \{s_2^r, \dots, s_2^v\}$) is a *S2* subset where $\forall s_2^w \in D(s_1^i), s_1^i \vdash s_2^w$, and $0 \leq (s_2^w.mindist - s_1^i.mindist) \leq (s_2^w.mindist - s_1^t.mindist), \forall s_1^t \in (S1 - s_1^i)$. Each $D(s_1^i)$ is exclusive from any other dominance set; therefore, $S2 = D(S1)$, where $D(S1) = D(s_1^1) + \dots + D(s_1^m)$ and m is the size of *S1*.

Lemma 1. Given a dominance set $D(s_1^i)$. Let A be the skyline points extracted from $EDR(s_1^i)$. $D(s_1^i)$ must contain A (A is a subset of $D(s_1^i)$).

Proof. (By contradiction) Let $p \in A$ be a point not included in $D(s_1^i)$. This is a contradiction, since p is only dominated by s_1^i . Therefore, it must be in $D(s_1^i)$. It follows that $D(s_1^i)$ must contain all points in A . ■

In Figure 3, $D(s_1^2) = \{s_2^1, s_2^2\}$ contains two *S2* points in the set which is a superset of $A = \{s_2^2\}$. One can observe that some non-exclusive *S2* points (e.g., s_2^1 and s_2^4) can be assigned to different dominance sets. Intuitively, the *S1* point with the minimal *mindist* to the query point (which has the largest dominance area) may dominate the most *S2* points. Thus, it might produce a load imbalance problem because the query processor needs to perform many dominance checks when a skyline point with a short *mindist* moves. To ensure that each dominance set contains evenly distributed *S2* points, the *ESC* algorithm inserts a non-exclusive *S2* point s_2^w into $D(s_1^j)$, where s_1^j has the minimal value of $(s_2^w.minsit - s_1^j.mindist)$ among all other *S1* points. In our algorithm, we utilize the *BBS* approach to initially compute the skyline query results. Along with the query evaluation, *S2* points and the dominance set of each *S1* point are computed during the execution of the modified *BBS* dominance-checking

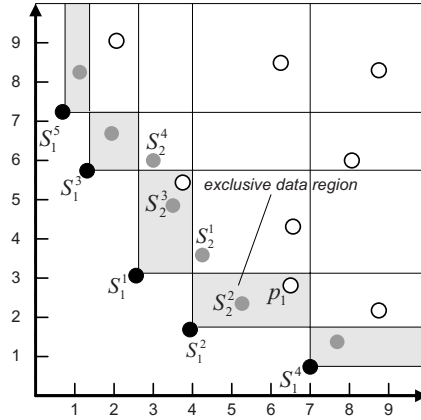


Fig. 3. Dominance set v.s. EDR set

procedure which runs a window query to determine a set of candidate skyline points. Let e be the next discarded entry during the process of the dominance-checking procedure (e is dominated by some S_1 point). Therefore, the algorithm proceeds to insert e into a dominance set and examines whether e is a S_2 point. Given is a heap $H = \{s_1^i \dots s_1^k\}$ that represents the set of the existing skyline points whose entries intersect with e . Since BBS always visits entries in the ascending order of their *mindist*, we have $\forall s \in H, s.mindist < e.mindist$. With the sorting of H by the *mindist* in descending order, $\exists s_1^j \in H, s_1^j \vdash e$ and the value of $(e.minsit - s_1^j.mindist) > 0$ is minimal among all other S_1 points. Next, Lemma 2 is provided to prove the correctness of the S_2 extraction.

Lemma 2. Given a point p which is dominated by $S_1' = \{s_1^i \dots s_1^j\}$, where $S_1' \subset S_1$. If $\forall s_2^t \in D(S_1'), s_2^t \not\vdash p$, p must be a S_2 point.

Proof. Since p is not dominated by $(S_1 - S_1')$, p can never be dominated by any S_2 point in $D(S_1 - S_1')$ either, by transitivity. Therefore, if p is not dominated by any S_2 point in $D(S_1')$, p is guaranteed to be a final S_2 point. ■

The pseudo code is shown in Algorithm 1, where the additional conditions (Lines 10-16 and 19-27) are inserted into the dominance-checking code for retrieving S_2 points and determining the dominance sets. Line 4 sorts the heap in descending order of the *mindist* such that the skyline points with larger *mindist* are examined first. Line 12 obtains the dominating skyline point e_r for p which is inserted into $D(e_r)$ later. Based on Lemma 2, Lines 13–15 check whether p is a S_2 point. Lines 20–23 ensure that each S_2 is a data point. If e is an intermediate node, BBS is performed to retrieve local skyline points from the entry. Lines 23 and 25 insert the final S_2 points O' into S_2 and update the S_2 set by deleting those S_2 points that are dominated by O' . To find such a set, the algorithm performs $S2Rtree.W(O')$, which is a window query that finds the S_2 points in the dominance areas of O' .

Algorithm 1. ESC dominance-check(p)

```

1: insert all entries of the root R in the heap
2: isDominated = false,  $e_r = \phi$ 
3: while heap not empty do
4:   remove top heap entry  $e$  //the heap is sorted in descending order of mindist.
5:   if ( $e$  is an intermediate entry) then
6:     for (each child  $e_i$  of  $e$ ) do
7:       if ( $e_i$  intersects with  $p$ ) then insert  $e_i$  into heap
8:     end for
9:   else
10:    if ( $e \vdash p$ ) then
11:      isDominated = true;
12:      let  $e_r = e$ , if  $e_r$  is not empty //  $e_r$ : the first  $S1$  point dominating  $p$ 
13:      for (each  $S2$  skyline point  $s_2^i \in D(e)$ ) do
14:        if ( $s_2^i \vdash p$ ) then set  $p$  as a regular data point and return isDominated
15:      end for
16:    end if
17:  end if
18: end while
19: if (isDominated) then
20:   if ( $p$  is an intermediate entry) then
21:     perform DataRtree.BBS( $p$ ) that returns a skyline point set  $O$ 
22:     let  $O' \in O$  be the data set that is not dominated by  $S2$ .
23:      $S2 = S2 + O' - S2Rtree.W(O')$  and insert  $O'$  into  $D(e_r)$ 
24:   else
25:      $S2 = S2 + p - S2Rtree.W(p)$  and insert  $p$  into  $D(e_r)$ 
26:   end if
27: end if
28: return isDominated

```

3.3 Description of the ESC Algorithm

The main procedures of the *ESC* algorithm include *S1Evaluation* for the $S1$ updates and *S2Evaluation* for the $S2$ set maintenance. *ESC* delegates most of expensive computations that are irrelevant to $S1$ query results to *S2Evaluation*. To improve the performance of *S2Evaluation*, we introduce the concept of an *approximate exclusive data region (AEDR)* that helps to reduce the amortized cost of the $S2$ updates. When $d = 2$, the traditional *EDR* is a regular rectangle. However, an *EDR* has an irregular shape in higher dimensions. For example, in Figure 4(a), s_2^i is a skyline point to delete. The *EDR* is an irregular rectangle after deleting the overlapping area with the dominance area of s_2^k and s_2^v . Based on this observation, we can obtain a regular shaped *EDR* only when we consider the skyline points which have a value x^i larger than that of s_2^i in only one dimension. Because these points are completely “outside” of the *EDR*, they can trim the entire areas that represent the upper dimensional value x^i .

Definition 3. (*AEDR*)

Let $s_2^i = (x^1, x^2, \dots, x^d)$, and $s_2^j = (y^1, y^2, \dots, y^d)$. $AEDR(s_2^i) = s_2^i.DomArea - (s_2^i.DomArea \cap s_2^j.DomArea)$, $\forall s_2^j \in (S2 - s_2^i)$, there exists exactly one $x^k < y^k$, $1 \leq k \leq d$.

For example, in Figure 4(b), s_2^i is the skyline to delete and the solid rectangle box is an *AEDR*, which is a regular shape resulting from trimming the overlapping dominance areas of s_2^i and s_2^j . *ESC* utilizes the *AEDR* to search for the new $S2$ points by traversing the R-tree. Each *MBR* e extracted from the heap is

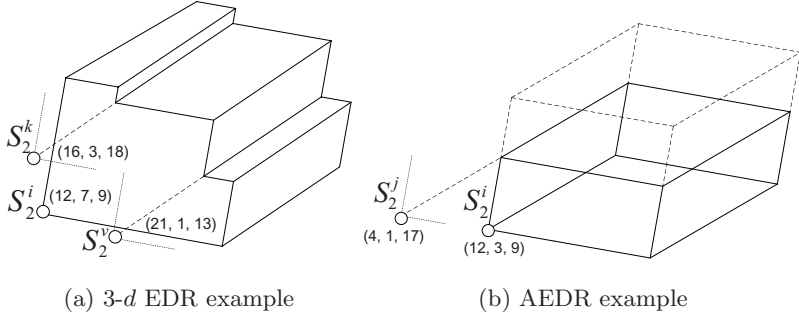


Fig. 4. Traditional *EDR* v.s. *AEDR*

checked whether it intersects with the *AEDR*. If true, *ESC* checks whether e is dominated by the existing $S2$ points.

When a $S1$ point p is newly inserted into the system or when it moves, *ESC* needs to re-group a new dominance set for p . A simple solution is to check every $S2$ point which currently belongs to a dominance set of some $S1$ point and migrate the $S2$ point to the dominance set of p if necessary. Instead, we provide *FindDomSet*, (the pseudo code is presented in Algorithm 2) applying the following Lemma that presents a heuristic to avoid checking the entire $S2$ set.

Lemma 3. Given a new $S1$ point s_1^k , re-group the points in $D(s_1^i)$, only where $\forall s_1^i \in (S1 - s_1^k)$, $s_1^i.mindist \leq s_1^k.mindist$.

Proof. Proof by definition. Let s_1^w be a $S1$ point that has the value of $(s_1^w.mindist > s_1^k.mindist)$. $\forall p \in D(s_1^w)$, the value of $(p.mindist - s_1^w.mindist)$ must be smaller than the value of $(p.mindist - s_1^k.mindist)$. Therefore, p must remain in the same dominance set of s_1^w . Hence, it is not necessary to re-group these points in $D(s_1^w)$. ■

Algorithm 2. *FindDomSet*(s_1^k)

```

1: for (each  $p \in D(s_1^i)$ , where  $s_1^i \in (S1 - s_1^k)$  and  $s_1^i.mindist < s_1^k.mindist$ ) do
2:   if ( $s_1^i \vdash p$ ) then
3:      $D(s_1^i).remove(p)$ 
4:      $D(s_1^k).insert(p)$ 
5:   end if
6: end for

```

The *ESC* algorithm is implemented in an event-driven fashion to handle the skyline query updates. The main procedures include *S1Evaluation* (Algorithm 3) and *S2Evaluation* (Algorithm 4). When the query processor receives a request (from a point in $S1$, $S2$, or a regular data point), it first performs *S1Evaluation*

to examine whether the request affects the $S1$ set (the query result) and outputs the updated $S1$ points if the set has been modified. Then $S2Evaluation$ processes the rest of non- $S1$ -related computations. In the $S1Evaluation$ procedure, Line 6 performs the $S1Rtree.dominance-descending$ function where the dominance checks access the $S1Rtree$ in the descending order of the $mindist$ of the entries. We use the same principle of the ESC dominance-check algorithm (discussed in Section 3.2) to find the dominating $S1$ point s_1^k (Line 7) for a request point p . If p becomes a new $S2$ point evaluated by $S2Evaluation$, p is inserted into $D(s_1^k)$. Lines 9–10 update the $S1$ set if p is a new $S1$ point and delete the I set, which is an existing $S1$ set dominated by p . I is obtained by executing a window query $S1Rtree.W(p)$, using the dominance area of p as the range on the $S1Rtree$. Line 11 inserts the new $S1$ point p into $\widetilde{S1}$ and $S1Evaluation$ will later pass this set to $S2Evaluation$ where $FindDomSet(\widetilde{S1})$ is performed to find a $S2$ set for $D(p)$. Since all the points in I become new $S2$ points (inserted into $\widetilde{S2}$ in Line 12), the $S2$ set is updated later in $S2Evaluation$ by adding the $\widetilde{S2}$ set. Lines 15–24 basically check all the $S2$ points $\in D(p')$ whether they are still dominated by p after p moves or is removed from the system. In Line 18, since o (a new $S1$ point after p moves) can never dominate any $S1$ point, o is added to the $S1$ set directly. This is because o is an exclusive data point, and therefore o must not dominate any existing $S1$ points.

Algorithm 3. $S1Evaluation(p)$

```

1: let  $\widetilde{S1} = \phi$  be a new  $S1$  point set
2: let  $\widetilde{S2} = \phi$  be a new  $S2$  point set
3: let  $\overline{S2} = \phi$  be the existing  $S2$  points to remove
4:  $p'$  be the last-updated point of  $p$ 
5:  $S1 = S1 - p'$ , if  $p$  was a  $S1$  point
6:  $isDomByS1 = S1Rtree.dominance-descending(p)$ 
7: let  $s_1^k$  be the  $S1$  point with the minimal  $(p.minsit - s_1^k.mindist)$  value among all other  $S1$  points

8: if ( $isDomByS1 == \text{false}$ ) then
9:    $I = S1Rtree.W(p)$ 
10:   $S1 = S1 + p - I$ 
11:   $\widetilde{S1}.insert(p)$ 
12:   $\widetilde{S2}.insert(I)$ 
13:   $D(p).insert(i), \forall i \in I$ 
14: end if
15: if ( $p$  was a  $S1$  point) then
16:   for (each  $o \in D(p')$ ) do
17:     if ( $S1Rtree.dominance-descending(o) == \text{false}$ ) then
18:        $S1 = S1 + o$ 
19:        $D(p).remove(o)$ 
20:        $\widetilde{S1}.insert(o)$ 
21:        $\widetilde{S2}.insert(o)$ 
22:     end if
23:   end for
24: end if
25: output the updated  $S1$  set and continue  $S2Evaluation(p, isDomByS1, s_1^k, \widetilde{S1}, \widetilde{S2}, \overline{S2})$  procedure

```

$S2Evaluation$ is a more expensive procedure than $S1Evaluation$, because it involves $AEDR$ computations to find a set of new $S2$ points to substitute a

moving or removed $S2$ point. Lines 6–7 are processed if p is a new $S2$ point. The insertion of p may dominate some existing $S2$ points; therefore, Line 6 finds the dominated $S2$ points ($S2Rtree.W(p)$) and removes them from the $S2$ set. Similarly, in Line 10, since each point in $\widetilde{S2}$ was originally a $S1$ point, the $D(\widetilde{S2})$ set is directly removed from the $S2$ set without performing a window query to look for the dominated points. The deletion of the $S2$ point set $\widetilde{S2}$ is executed in Lines 11–12 and A contains the substitute $S2$ points, after $\widetilde{S2}$ is removed from the $S2$ set. Finally, $FindDomSet$ is performed to find a group of $S2$ points for each point in $\widetilde{S1}$.

Algorithm 4. $S2Evaluation(p, isDomByS1, s_1^k, \widetilde{S1}, \widetilde{S2}, \overline{S2})$

```

1: Let  $p'$  be the last-updated point of  $p$ 
2:  $\overline{S2}.insert(p')$ , if  $p$  was a  $S2$  point
3: if ( $isDomByS1 == \text{true}$ ) then
4:    $isDomByS2 = S2Rtree.dominance(p)$ 
5:   if ( $isDomByS2 == \text{false}$ ) then
6:      $S2 = S2 + p - S2Rtree.W(p)$ 
7:      $D(s_1^k).insert(p)$  and  $D(s_1^{k'}).remove(p)$ , where  $s_1^{k'} (\neq s_1^k)$  was the dominating point of  $p$ 
8:   end if
9: end if
10:  $S2 = S2 + \widetilde{S2} - D(\widetilde{S2})$ 
11:  $A = DataRtree-AEDR(\overline{S2})$ , where  $A$  is a regular data set and is not dominated by  $S2$  points.
12:  $S2 = S2 - \widetilde{S2} + A$  //  $A$  substitutes  $\widetilde{S2}$ 
13:  $FindDomSet(\widetilde{S1})$ 

```

4 Experimental Evaluation

We evaluated the performance of the *ESC* algorithm by comparing it with the well-known *BBS* approach [14] and the *DeltaSky* algorithm [19]. For the *EDR* computations in *BBS*, we adopt the *ABBS* (Adaptive Branch-and-Bound Search) [19] to avoid complex irregular-shaped *EDR* computations. *ABBS* basically traverses the R-tree and determines whether an intermediate *MBR* e_i intersects with the dominance area of a skyline to delete. If this is true, it further checks whether any existing skyline dominates e_i . All of these algorithms utilize R-trees as the underlying structure for indexing the data and skyline points. We use the Spatial Index Library [7] for the R-tree index. A page size of 4Kbytes is deployed, resulting in node capacities between 94 ($d = 5$) and 204 ($d = 2$). $S1$ and $S2$ sets are indexed by a main-memory R-tree to improve the performance of the dominance checks. Our data sets are generated on a terrain service space of $[0, 1000]^2$ with the random walk mobility model [10]. Each object moves with a constant velocity until an expiration time. The velocity is then replaced by a new velocity with a new expiration time. We generated from 100,000 to 1,000,000 normal distributed data points with dimensions in the range of 2 to 5. The object update ratio is set in a range from 1% to 10%. Experiments are conducted with a Pentium 3.20 GHz CPU and 1 GByte of memory. The query results are evaluated in an event-driven approach. Therefore, the query processor

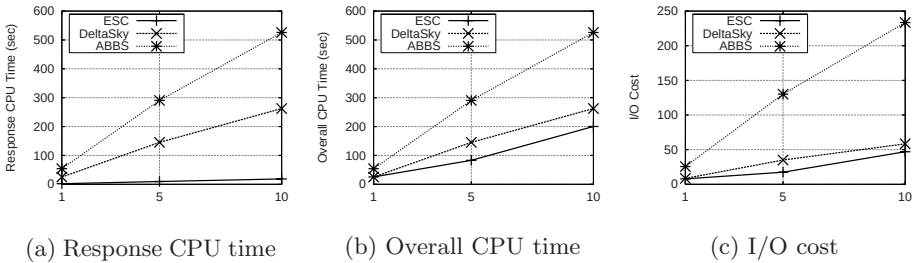
Table 2. Simulation parameters

Parameter	Default	Range
P	100,000	100,000, 500,000, 1,000,000
d	5	2, 3, 4, 5
f_{update}	10%	1%, 5%, 10%

calls different procedures based on each specific event type. The main measurement in the following simulations is the response CPU time (from receiving a data update request to the $S1$ update completion time or the evaluation time of $S1Evaluation$) and the overall CPU time (the evaluation time of $S1Evaluation$ plus $S2Evaluation$). For *ABBS* and *DeltaSky* the overall CPU time also represents the response time. Our experiments use several metrics to compare these algorithms. Table 2 summarizes the default parameter settings in the following simulations.

4.1 Update Ratio

First, we evaluated the impact of the update ratio. Figures 5(a) and (b) show the response time and overall CPU time as a function of update ratio, respectively, and Figure 5(c) illustrates the I/O cost for the three methods. We fix the data cardinality at 100,000 and dimensionality at 5. The *ESC* approach achieves a better performance than *ABBS* and *DeltaSky* for all update rates. The degradation of *DeltaSky* is caused by the expensive Maximum Coverage computations scanning over the projection lists and the increase of skyline point size which incurs bigger projection lists. *ESC* also outperforms both methods in terms of the overall CPU time, since the amortized cost of the *AEDR* computations and exclusive data evaluation is lower than the other two methods.

**Fig. 5.** Performance v.s. Update Ratio ($P = 100k$, $d = 5$)

4.2 Dimensionality

Next we report on the impact of the dimensionality on the performance of all three methods. Figures 6(a), (b) and (c) show the CPU overheads and I/O cost

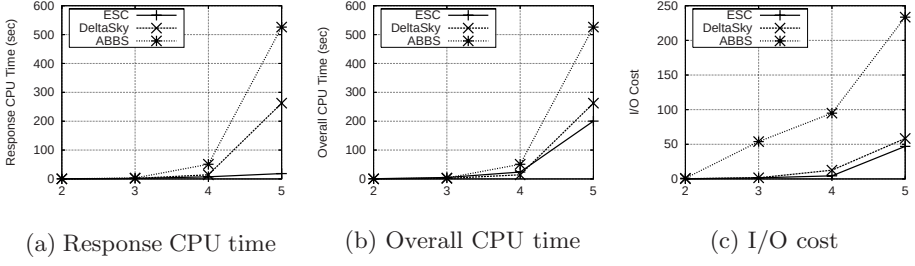


Fig. 6. Performance v.s. Dimensionality ($P = 100k$, $f_{update} = 10\%$)

v.s. the dimensionality ranging from $d = 2$ to 5, respectively. When d increases, the performance of all methods is degraded because the exclusive data point computations are complex and R-trees fail to filter out irrelevant data entries in higher dimensions. From all the figures we can see that *ESC* outperforms *ABBS* and *DeltaSky* in terms of the CPU time and I/O cost.

4.3 Cardinality

Figures 7(a) and (b) show the response and overall CPU time as a function of the number of data points, respectively, and Figure 7 (c) illustrates the corresponding I/O cost. Overall, the CPU overheads increase as a function of the number of data points. *ESC* achieves a significant reduction in terms of the response CPU time compared to *ABBS* and *DeltaSky*. *ESC* takes advantage of the pre-computed *S2* points retrieved by the latest *S2Evaluation* procedure and quickly locates relevant new *S1* candidates for substituting a removed or moving *S1* point. As we can see from the experimental results, the adoption of *AEDR* helps *ESC* to achieve better overall CPU performance and competitive I/O cost with *DeltaSky*.

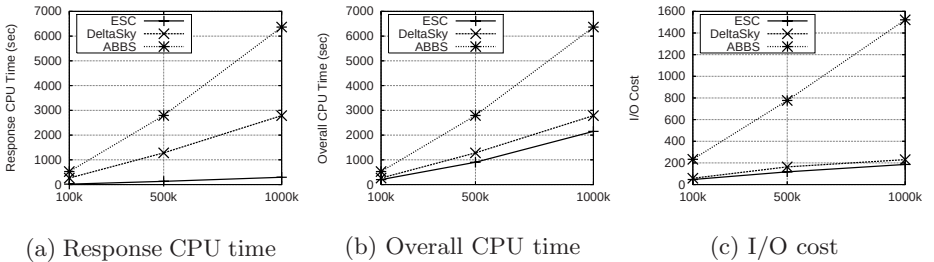


Fig. 7. Performance v.s. Cardinality ($d = 5$, $f_{update} = 10\%$)

5 Conclusions

In this paper, we propose an incremental skyline update approach. Our *ESC* algorithm achieves a faster response time and better overall CPU performance.

With the adoption of the pre-computed *S2* sets, *ESC* can efficiently update the skyline query results and delegate the most complex computations to a separate procedure that executes after the updates of the query results are completed. An approximate exclusive data region (*AEDR*) is proposed and our experiments confirm the feasibility of *AEDR* which has a low amortized cost of the exclusive data evaluation in high dimensional and dynamic data environments. The *S1Evaluation* procedure first examines all the incoming data requests and updates the *S1* result if necessary and the *S2Evaluation* procedure integrates our lemmas and heuristics to achieve a low CPU overhead and reduced I/O cost.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany, pp. 421–430 (2001)
2. Chan, C.Y., Eng, P.-K., Tan, K.-L.: Stratified computation of skylines with partially-ordered domains. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, pp. 203–214 (2005)
3. Gaede, V., Günther, O.: Multidimensional Access Methods. *ACM Comput. Surv.* 30(2), 170–231 (1998)
4. Huang, Z., Lu, H., Ooi, B.C., Tung, A.K.H.: Continuous Skyline Queries for Moving Objects. *IEEE Trans. Knowl. Data Eng.* 18(12), 1645–1658 (2006)
5. Kossmann, D., Ramsak, F., Rost, S.: Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In: Proceedings of 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China, pp. 275–286 (2002)
6. Lee, K.C.K., Zheng, B., Li, H., Lee, W.-C.: Approaching the Skyline in Z Order. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), pp. 279–290. University of Vienna, Austria (2007)
7. S.I. Library, <http://www.research.att.com/~mariah/spatialindex/index.html>
8. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the Sky: Efficient Skyline Computation over Sliding Windows. In: Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan, pp. 502–513 (2005)
9. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting Stars: The k Most Representative Skyline Operator. In: Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey, pp. 86–95 (2007)
10. McDonald, A.B.: A mobility-based framework for adaptive dynamic cluster-based hybrid routing in wireless ad-hoc networks. Ph.D. Dissertation proposal, University of Pittsburgh (1999)
11. Morse, M.D., Patel, J.M., Grosky, W.I.: Efficient Continuous Skyline Computation. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, GA, USA, p. 108 (2006)
12. Morse, M.D., Patel, J.M., Jagadish, H.V.: Efficient Skyline Computation over Low-Cardinality Domains. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), pp. 267–278. University of Vienna, Austria (2007)
13. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An Optimal and Progressive Algorithm for Skyline Queries. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, New York, NY, USA, pp. 467–478 (2003)
14. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive Skyline Computation in Database Systems. *ACM Trans. Database Syst.* 30(1), 41–82 (2005)

15. Pei, J., Jin, W., Ester, M., Tao, Y.: Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, pp. 253–264 (2005)
16. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea, pp. 751–762 (2006)
17. Tan, K.-L., Eng, P.-K., Ooi, B.C.: Efficient Progressive Skyline Computation. In: Proceedings of the 27th International Conference on Very Large Data Bases (VLDB), pp. 301–310. Morgan Kaufmann Publishers, San Francisco (2001)
18. Tian, L., Wang, L., Zou, P., Jia, Y., Li, A.: Continuous Monitoring of Skyline Query over Highly Dynamic Moving Objects. In: Sixth ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), Beijing, China, pp. 59–66 (2007)
19. Wu, P., Agrawal, D., Egecioglu, Ö., Abbadi, A.E.: Deltasky: Optimal Maintenance of Skyline Deletions without Exclusive Dominance Region Generation. In: Proceedings of the 23rd International Conference on Data Engineering (ICDE), The Marmara Hotel, Istanbul, Turkey, pp. 486–495 (2007)

Semantic Decision Tables: Self-organizing and Reorganizable Decision Tables

Yan Tang¹, Robert Meersman¹, and Jan Vanthienen²

¹ Semantic Technology and Application Research Laboratory (STARLab),
Department of Computer Science,
Vrije Universiteit Brussel, Pleinlaan 2 B-1050 Brussels, Belgium
{yan.tang, robert.meersman}@vub.ac.be

² Katholieke Universiteit Leuven, Faculty of Business and Economics
Department of Decision Sciences and Information Management
Naamsestraat 69, 3000 LEUVEN Belgium
jan.vanthienen@econ.kuleuven.be

Abstract. A Semantic Decision Table (SDT) provides a means to *capture* and *examine* decision makers' concepts, as well as a tool for *refining* their decision knowledge and facilitating *knowledge sharing* in a *scalable* manner. One challenge SDT faces is to organize decision resources represented in a tabular format based on the user's needs at different levels. It is important to make it *self organized* and automatically *reorganized* when the requirements are updated. This paper describes the ongoing research on SDT and its tool that supports the self organizations and automatic reorganization of decision tables. We argue that *simplicity*, *precision*, and *flexibility* are the key issues to respond to the paper challenge. We propose a novel combination of the principles of Decision Support and Database Modeling, together with the modern technologies in Ontology Engineering, in the adaptive self-organization and automatic reorganization procedures (SOAR).

1 Introduction

Sharing decision resources efficiently is mandatory for group decision making. The problems of *ambiguity*, *inconsistency* and *scalability*, which occur while drawing a decision table amongst a decision group, are tackled by Semantic Decision Table (SDT, [20]). SDT provides a means to *capture* and *examine* decision makers' concepts, as well as a tool for *refining* their decision knowledge and facilitating *knowledge sharing* in a *scalable* manner. An SDT is the result of annotating a set of decision tables (or any well structured decision resources) with ontologies. It contains *richer* decision rules than a mere decision table, as it specifies the hidden decision rules and meta-decision rules of a decision table. We guide a decision group to construct an SDT using an efficient stepwise methodology described in [19]. Note that the term "decision table" used in this paper is a table that contains decision rules, which can be an incomplete rule set.

In our current research projects, such as the EC Prolix project¹, SDT is used as a tool embedded in decision processes of a system in order to improve its flexibility and effectiveness, such as in [21]. An important feasibility provided by SDT is to visualize the results in the form of decision tables when the decisions are taken in every micro process. Recently, we get increasing requirements of managing the decision tables at a high level; enabling them *self organized* and automatically *reorganized* when the user queries are updated. We consider this kind of decision tables as an extension to SDT. It needs to automatically check the dependencies of different knowledge blocks, quickly adapt to dynamic inputs, and accurately generate the decision tables. These requirements become the challenges of this paper.

A traditional decision table takes the form of a ‘flat’ reasoning structure with three basic constituents [3]. One constituent is the *condition stubs* and *action stubs*; the second one holds the *condition entries* and the *action entries*; the third one includes the *decision rules*, each of which corresponds to a combination of the elements in the above two constituents. The only constituent type used for reasoning is the third one, which is physically represented as the table columns.

SDT, in general, also contains these three constituent parts. In addition, SDT provides three types of sub-elements for reasoning: 1) the one that corresponds to the dependencies between the conditions (or between the actions); 2) the hidden decision rules, constraints or operational dependencies between the conditions and the actions; and 3) the (possible) meta-rules of a set of decision tables.

The three extra elements of SDT are the key approaches to the paper challenges. In this paper, we propose a novel combination of the principles of Decision Support and Database Modeling, together with the modern technologies in Ontology Engineering, in the procedures called SOAR. SOAR is the abbreviation of the collection of the adaptive Self-Organization and Automatic Reorganization procedures for SDT. In this paper, we propose to use the principles of data dependencies in Database Modeling to constrain the output of SOAR, and thus improve its *precision*.

In our early paper [19], we are careful to stress that SDT, as a sort of group decision support system, has a natural connection with ontology engineering. Ontologies [6, 7], in modern computer science realm, are used to model a domain so far as a universe of discourse. An ontology, by definition, is supposed to be *consistent*. Seeing the reasoning feasibility provided by modern ontology engineering, we store the decision rules at different levels, including the meta-decision rules and other constraints, as a set of *axioms* in an ontology. In this paper, SOAR contains the checksum of the ontological constraints before generating the outputs. By doing so, we can ascertain that its outputs are consistent. We argue that *simplicity*, *precision*, and *flexibility* are the key issues to respond to the algorithm.

The approach of considering SDTs as self-organizing and reorganizable decision tables is based on the characteristics of semantics stored in SDTs. SDT is defined as a decision table with appropriate semantics, containing the constraints at different level, as well as a system that supports data learning. The remainder of this paper is structured as follows. In section 2, we present a grounded understanding of Semantic

¹ The objective of PROLIX is to align learning with business processes in order to enable organizations to faster improve the competencies of their employees according to continuous changes of business requirements. URL: <http://www.prolixproject.org/>

Decision Tables (SDTs, section 2.1) and the paper motivation (section 2.2). We design the procedures in SOAR in section 0. SOAR checks whether all the constraints represented by an SDT are satisfied before the outputs are generated. It also provides the outputs at different levels. Section 3.1 details the main constraints used for SDT. An SDT tool called “SDT SOAR Plug-in” that supports SOAR is demonstrated in section 3. We present our experimental analysis in section 4. We compare our work with the existing technologies, and discuss both the advantages and disadvantages of our work in section 5. Section 6 contains the paper conclusion and the future work.

2 Background

Based on the de-facto standard [3], there are three basic elements in a decision table: the *conditions*, the *actions* (or decisions), and the *rules* that describe which actions might be taken based on the combination of the conditions. A condition is described by a *condition stub* and a *condition entry*. A condition stub contains a statement of a condition. Each condition entry indicates the relationship between the various conditions in the condition stub. An action (or decision) contains an *action stub* and an *action entry*. Each action stub has a statement of what action to be taken. The action entries specify whether (or in what order) the action is to be performed for the combination of the conditions that are actually met in the rule column.

Table 1. A simple example of a traditional decision table², which is used to decide whether we hire a driver or not

	1	2	3	...
Condition				
Has driver’s license	Yes	Yes	Yes	...
Previous job	Bus driver	N/A	N/A	...
Language	French, Dutch	French	English	...
Action				
Hire	*			...
Hire and train		*		...

Table 1 presents a part of a simple decision table with three conditions: “Driver’s license type”, “Previous job” and “Language”; and two actions: “Hire” and “Hire and train”. The condition “Has driver’s license” has two condition entries - “Yes (the person has a driver’s license)” and “No (the person doesn’t have a driver’s license)”. The rule column with ID ‘1’ expresses a decision rule as “If one person has a driver’s license, his previous job is a bus driver and he speaks French and Dutch, then hire him”.

² A traditional decision table is often used as a complete set of decision rules in computer science, e.g. decision tables as a programming tool [2]. Strictly speaking, if Table 1 only contains three decision columns, it is not called a traditional decision table. Rather, it is a table consists of three decision rules.

2.1 SDT: Semantic Decision Table

The notion of Semantic Decision Table (SDT, [20]) was initially introduced to tackle the following problems in a traditional decision table: 1) *ambiguity* in the information representation of the condition stubs or action stubs, 2) *conceptual duplication* amongst the conditions, 3) *uncertainty* in the condition entries, and 4) difficulties in managing *large* tables (also known as the *scalability* problem). What makes an SDT different from a traditional decision table is its *semantics*. Unlike traditional decision tables, the concepts, variables and decision rules are explicitly defined.

An SDT is modeled in three-layer format: 1) the layer of the decision *binary fact types* called SDT *lexons*, 2) the SDT commitment layer that contains the constraints and axioms of these fact types; and 3) the layer of decision tasks or applications. The three-layer format is designed based on the principles of Developing Ontology-Grounded Methods and Applications approach to ontology engineering (DOGMA, [17]), which has been the main research topic at the VUB STARLab over ten years.

An SDT *lexon* is a quintuple $\langle \gamma, t_1, r_1, r_2, t_2 \rangle$, where γ is a context identifier. γ is assumed to point to a resource, and serves to disambiguate the terms t_1, t_2 into the intended concepts. r_1, r_2 , which are “meaningful” in this specific context γ , are the roles referring to the relationships that the concepts share with respect to one another. For example, a lexon $\langle \gamma, \text{driver}, \text{has}, \text{is issued to}, \text{driver's license} \rangle^3$ explains a fact that “a driver has a driver’s license”, and “a driver’s license is issued to a driver”. The linguistic nature of a lexon represents that a fundamental DOGMA characteristic is its grounding in the linguistic representation of knowledge. The community of decision makers chooses (or has to agree on) a given (natural) language, e.g. English, to store and present lexon terms and roles.

An SDT *commitment* corresponds to an explicit instance of an intentional interpretation by a decision task. It contains a set of rules in a given syntax, and describes a particular application view of reality, such as the use by the application of the (meta-) lexons in the lexon base. The commitments need to be expressed in a *commitment language* that can be easily interpreted. Suppose that the above lexon - $\langle \text{driver}, \text{has}, \text{is issued to}, \text{driver's license} \rangle$ - has the constraint as “EACH driver should have AT LEAST ONE driver’s license”. We apply the *mandatory* constraint on the lexon written as below:

P1 = [driver, has, is issued to, driver’s license]: (1)
MAND (p1).⁴

The decision rules in a decision table can be equivalently mapped into a set of SDT commitments. For example, the following commitment is the decision rule in column 1 of Table 1.

In this use case, SDT is the result of annotating a decision table with ontologies. The goal of using SDT is to tackle the problems, such as the ambiguity problem

³ In this paper, we do not focus on the discussion of the context identifier γ , which is omitted in other lexons. E.g. $\langle \gamma, \text{driver}, \text{has}, \text{is issued to}, \text{driver's license} \rangle$ is thus written as $\langle \text{driver}, \text{has}, \text{is issued to}, \text{driver's license} \rangle$.

⁴ The syntax can be found at: <http://www.starlab.vub.ac.be/website/SDT.commitment.example>

```

(P2 = [Has driver's license, has, is of, value],
P3 = [Previous job, has, is of, value],
P4 = [Language, has, is of, value],
P5 = [action, is about, is a, Hire])
: IMP5 (AND (P2 (value) = 'Yes', P3 (value) = 'Bus
driver', P4 (value) = 'French, Dutch'), P5).

```

(2)

and the conceptual duplication problem, early discussed in this section. During the anotation process, the decision makers need to specify all the hidden rules, such as “*EACH driver should have AT LEAST ONE driver's license*” shown above. Thus, an SDT contains *richer* decision rules than a mere decision table.

There are many other interesting use cases of SDT. One of them is to embed SDT in a process, separate decision rules from the process in order to improve the system flexibility [21]. An important feasibility provided by SDT is to visualize the process results in the form of decision tables when the decisions are taken in every micro process. A detailed explanation is given in the next subsection.

2.2 A Use Case of SDT and Motivation

Suppose we have a training process in the domain of human resource management. We want to train the employees from different companies, e.g. MIVB⁶. Firstly, we collect the data of the employees from the company. The data can be personal information or professional background, e.g. the name, the address and the driving skills. Then, we decide which courses he should take. The decisions are drawn based on many decision rules, which can be modeled and embedded in various approaches, such as business process models⁷ (BPM, [16]).

This use case is a simplified one we encounter in the EC Prolix project. We are motivated to use SDT because of its advantages. An SDT is a subtype of a decision table. It has all the advantages of a decision table. E.g. a decision table is extremely convenient and user-friendly for non-technical people. It can be easily imported to their workbench, such as Excel⁸. An SDT is a decision table enhanced by semantics, which makes it better than a mere decision table. As early discussed at the beginning of section 2.1, SDT has many advantages over a decision table. For example, an SDT doesn't contain any ambiguities in the decision items, as they are properly annotated with ontologies. We refer to [19, 20, 21] for more details.

In addition, we're motivated to use SDT because of the feasibility of building SDTs. We build an SDT with the method in [20], which requires domain ontologies.

⁵ IMP is the implication operator. AND is the conjunction operator. This SDT commitment is verbalized as: IF the value of 'Has driver's license' is 'Yes', AND the value of 'Previous job' is 'Bus driver', AND the value of 'Language' is 'French, Dutch', THEN the action is about (to) 'Hire'.

⁶ It is a public transport company in Belgium. <http://www.mivb.be>

⁷ Nevertheless, the decision rules are separated from the processes.

⁸ Excel is part of the Microsoft® Work Suit, which are widely used by many enterprises. An Excel file contains a spreadsheet, which is used to design informal decisions. <http://office.microsoft.com/en-us/excel/>

In the Prolix project, one training center, such as GENO⁹ in the project, is responsible for training the employees from many companies. Different companies can have different database systems and applications; therefore, the domain ontologies are constructed to improve the *interoperability*. We can use the available ontologies to build an SDT.

Recently, we get detailed requirements as follows:

- The first (and probably the most important) requirement is to automatically check the constraints, quickly adapt to dynamic inputs, and accurately (re-)generate the SDT. As soon as a user adds a new constraint, the SDTs earlier generated should be rechecked.
- The second requirement is to present SDT at different levels when needed. A user may have a question on a specific decision in an SDT. For example, he wants the explanation of the first column in Table 1. He sees the formal SDT commitments bundled with the SDT. Unfortunately, he is not familiar with the syntax of the commitments. In a worse case, he even didn't contribute to the SDT commitment writing in the decision group. A simple solution is to provide the verbalization of the SDT commitments in a natural language. For example, "*EACH driver should have AT LEAST ONE driver's license*" is the verbalization of the commitment $P1 = [driver, has, is\ issued\ to, driver's\ license]: MAND(p1)$. The user is happy when there are only a few sentences. He gets nervous when he sees a big bunch of text. Therefore, we need a better solution to categorize the information. In practice, we observe that SDTs are often layered. One decision rule presented in a SDT can be propagated in another SDT. It gives us a hint to present SDT at different levels. Whenever a user wants to know a specific detail level, the system needs to automatically generate another SDT at required level.

The above requirements are the paper challenge and the main motivation: SDT needs to be *self organized* and automatically *reorganized* when the user queries are updated. We have been working on a tool called SDT Plug-in¹⁰ for more than two years. The plug-in implements many user scenarios of SDT, such as the SDT annotation scenario demonstrated in [20]. In this paper, we focus on the above requirements, design a collection of the adaptive *self organized* and automatically *reorganized* procedures (SOAR), which is introduced in section 3 and implemented in section 3.2.

3 SOAR

SOAR is a collection of the adaptive **S**elf-**O**rganization and **A**utomatic **R**eorganization procedures used for SDT. Fig. 1 shows the pseudo code for three main procedures.

⁹ <http://www.geno-stuttgart.de/>

¹⁰ It is a Java plug-in used in the DOGMA Studio Workbench, which is an ontology engineering tool developed by VUB STARLab. It collects the implementations of all the researching efforts at the lab, e.g. the implementation of the ontology creation methodologies, domain ontology modeling and visualization. A detailed explanation of DOGMA Studio Workbench and the plug-ins can be found at: <http://www.starlab.vub.ac.be/website/tools>.

<pre> Generate_1(condition stub[], action stub[], SQL query){ load data from database based on SQL query; generate key rows in decision table DT; complete DT; load ontology constraints set ONT[]; load SDT commitments SDTC[]; while (consistent(DT, SDTC[], ONT[])) is false){ user edits SDTC[]; generate_1(condition stub[], action stub[]); } generate SDT; } </pre> (a) Pseudo code for generating SDT from the database and user defined table layout <pre> Generate_2 (action, column ID){ load relevant ontological constraints set ONT[]; load relevant SDT commitment set SDTC[] ; while (SDT of next level exists){ visualize SDT of next level; visualize ONT[]; generate the verbalization; } } </pre> (c) Pseudo code for generating SDT of all levels	<pre> Boolean consistent(DT, SDTC[], ONT[][]){ for all constraints in ONT[] { if (DT satisfies ONT[]){ for all constraints in SDTC[] { DT satisfies SDTC[]; return true; } } } return false; } </pre> (b) Pseudo code for the SDT consistency checking <pre> reorganize(SDT, commitment[]){ load ontology constraints set ONT[]; for all constraints in commitment[] { if (database is not consistent with new constraint){ propose to delete this con- straint; user deletes the constraint; } } while (consistent(SDT, commitment[], ONT[][])) is false){ delete inconsistent SDT column; } generate SDT; } </pre> (d) Pseudo code for reorganizing an existing SDT when new commitments are added

Fig. 1. Pseudo code for SOAR

We explain Fig. 1 as follows:

- The procedure generate_1 () is executed to generate SDT from the data stored in the database. First, users need to provide condition stubs (e.g. “Name” and “Has driver’s license” in Table 2) and action stubs (e.g. “Driving course type” and “Language course type” in Table 2) for the table layout, which are the input of the procedure. Second, users need to provide at least one key condition stub of the table. For example, “Personnel ID” is the key condition stub in Table 2. Other data is automatically filled in Table 2 by looking up in the database system. This process is based on the unique key and the foreign keys in DB models. Third, users need to provide an SQL query to select a few records from the database. In a big company, the database can be rather big; therefore, we need the select query to ensure the size of the generated SDT under control. For example, we select our data in department X for Table 2. Fourth, when a temporary generated SDT is inconsistent with the ontologies, users need to edit¹¹ the SDT commitments, which are often predefined and stored as a set of business rules.

¹¹ Based on the requirement in practice, users are not allowed to change the ontologies, but they can override the ontological constraints in the SDT commitments.

Table 2. A decision table that decides which training courses are suitable for an employee in department X

	1	2	3	4	5	6	7	8
Condition								
Name	Tom	John	Emily	White	Lee	Rose	Kate	Smith
Personnel ID	4240	4561	4310	1003	4290	4296	1300	3390
Has driver's license	Yes	No	Yes	Yes	No	No	Yes	Yes
Driver's license type	D	D	A	A	/	/	D	D
Previous relevant job	Bus driver	/	Taxi driver	/	/	/	Bus driver	/
Age	28	30	28	63	20	18	50	35
Language	Dutch, French	Dutch	French, English	Dutch	French	English	Dutch	French
Experience (years)	2	/	5	40	/	/	20	10
...	...	...	...	...	...	...	...	...
Decision								
Driving course type	D	A	B		A	A	C	C
Language course type	FR C, EN A	FR A	FR C		DU A	FR A	FR A	DU A
...	...	...	...	...	...	...	...	...

	1	2	3	4	5
Condition					
Driving skill	≤ 2	$\leq 5, > 2$	$\leq 8, > 5$	$\leq 12, > 8$	> 12
Decision					
Driving course type	A	B	C	D	

An SDT commitment for the SDT on the left side:

($P1 = [Driving\ skill, decides, is\ decided\ by, Driving\ course\ type]$, $P2 = [Driving\ skill, is, is\ a, Read\ road\ sign]$, $P3 = [Driving\ skill, is, is\ a, Basic\ control]$, $P4 = [Driving\ skill, is, is\ a, Manage\ Vehicle\ distance]$): $P1(Driving\ skill) = P2(Driving\ skill) + P3(Driving\ skill) + P4(Driving\ Skill)$, $IMP(AND(P1(Driving\ skill) \leq 12, P1(Driving\ skill) > 8), P1(Driving\ course\ type) = 'D')$.

Fig. 2. Two SDTs that shows a decision rule at two different levels

- The procedure `reorganize()` is executed when a user adds new commitments to an existing SDT. The system first checks the consistency of the existing database with the commitment set. It proposes to the user to delete this constraint when there is a conflict. For example, if the user wants to add a commitment as “each user has at least one previous relevant job”. The existing database may not satisfy his mandatory constraint (see the “Language” data that is automatically filled in the condition entries in Table 2). The solution proposed in this procedure is to delete¹² this constraint in the commitment set. Then, the system checks the consistency of the commitment set with the existing ontology. Users need to edit the commitment set when the conflicts happen.
- The procedures `generate_2()` is executed when a user wants to visualize a decision rule of all levels. First, a user provides an action/decision stub in an existing SDT, e.g. “Driving course type” in Table 2, and a column number of an SDT, e.g. column “1” in Table 2. Then, the system loads all the relevant SDT commitments and ontological constraints. An SDT commitment or an

¹² In practice, it costs too much if users change the database system in a company just for one SDT. Therefore, they are required to delete the constraint in SDT commitments when the conflicts happen.

ontological constraint is relevant when it contains this action. Then, the system finds a set of condition stubs needed by this action. It generates another SDT by filling the actual data, which are retrieved from the database system, in the conditions. This process is repeated until no more SDT can be generated (see two SDTs in Fig. 2). The SDT on the left side explains the decision rule with column ID “1” in Table 2 (see the case of “Tom”). It shows all the relevant conditions for “Driving course type D”, such as “(the ability to) Read road sign”. The SDT on the right side (Fig. 2) shows a more general decision rule about “Driving course type”. Relevant SDT commitments are listed. In the meanwhile, necessary verbalizations of the SDT commitments and ontological constraints are generated. For example, the SDT commitment in Fig. 2 is verbalized as: the value of “Driving skill” is the total number of “Read road sign (skill level)”, “Basic control (skill level)” and “Manage vehicle distance (skill level)”; if the value of “Driving skill” is less than or equal to 12, and it is larger than 8, then the value of “Driving course type” is “D”.

All the procedures in SOAR contain the consistency checking. We have defined 22 SDT constraint types in 6 categories. In the next subsection, we explain how to check the consistency of a few SDT constraints, which are mostly used in SOAR.

3.1 Constraints in Semantic Decision Tables

A traditional decision table takes the form of a ‘flat’ reasoning structure represented by three basic constituents. The first one contains the *stubs* of conditions and actions/decisions (e.g. “Has driver’s license” is a condition stub in Table 2. “Driving course type” is a decision stub); the second one holds the *entries* of the conditions and actions/decisions (e.g. “Yes” is a condition entry for the condition “Has driver’s license” in Table 2. “D” is a decision entry for the decision “Driving course type”); the third one includes the *decision rules*, each of which corresponds to a combination of the elements in the above two constituents. The only constituent used for reasoning is the third one, which is physically represented as the table columns (e.g. column 1 in Table 2).

SDT, in general, also contains these three constituent types. In addition, SDT provides three sub-element types for reasoning¹³:

- 1) The one that corresponds to the dependencies between the conditions (or between the actions). For example, the condition of “Experience (years)” partly depends on the condition “Previous relevant job” in Table 2.
- 2) The one that represents the hidden decision rules, constraints or operational dependencies between the conditions and the actions. For example, the rule “if a person is about to be retired, then he doesn’t need to be trained” can be specified for Table 2.
- 3) The (possible) meta-rules of a set of decision tables. For instance, we can specify a meta-rule for Table 2 as “if a column doesn’t contain any decisions, then the table should not contain this column¹⁴”.

¹³ Note that all the constraints used for reasoning are stored as SDT commitments.

The three extra elements of SDT contain the main constraints in SOAR. The SDT constraints mostly used in SOAR are the constraints of *dependencies*, such as subset dependencies, and *logical operators*, such as implication.

As discussed in [13], dependencies in the most general sense are constrained relations in database modeling. Among all kinds of dependencies, *multivalued dependencies*, *subset dependencies*, and *mutual dependencies* are the mostly used. Based on the work in [8, 13], we carefully bring the database modeling principles into the ontology engineering and decision engineering. We mainly use multivalued dependencies, equality, subset, exclusion, mandatory, uniqueness and value constraints in [8].

The types of constraints depend on the requirements in practice. According to Halpin, the total number of constraint types, in theory, is infinite [8]. Including ORM, the various constraints among data have been extensively studied in the literature [1, 5, 8]. The specification illustrated in this section can be further translated into first-order-logic. The translation is useful for reasoning.

With regard to SOAR, it takes the knowledge of data in the database into account. Every record has its meaning. In other words, *data* has its *semantics*. It recalls the debates on whether separate data from knowledge or not, which has been carried on for a long time, e.g. in [14]. We argue that every data has its semantics. It is comparable to the fact that the content in a webpage has its meaning in the context of Semantic Web. By doing so, our approach can benefit from the modern technologies of semantics and ontologies. A drawback can be the difficulties at the implementation level.

By now, we have designed the self-organizing and reorganizing procedures and explained main constraints used in SOAR. In the next subsection, a tool that supports SOAR will be demonstrated.

3.2 SDT SOAR: A Tool to Support Self-organizing and Reorganizable Decision Tables

SOAR (Fig. 1) is developed as SDT SOAR Plug-in in DOGMA Studio Workbench 1.0¹⁵. The Workbench is constructed according to the plug-in architecture in Eclipse¹⁶. There, plug-ins, being loosely coupled ontology viewing, querying or editing modules support the different ontology engineering activities and new plug-ins continuously emerge. MySQL Server¹⁷ is used as the database management system to store the employee information.

There are five main views in the SDT SOAR Plug-in as indicated in Fig. 3. The top view is the SDT tabular view. The bottom view in the left corner represents a tree

¹⁴ It is not necessary to delete such columns in many cases. Otherwise, the debate on the completeness of decision table may arise. However, we put this meta-rule here, because our intention is to use it as an example to demonstrate the meta-rules of a decision table.

¹⁵ DOGMA Studio is a tool suite, which contains both a Workbench and a Server, to support DOGMA ontology engineering approaches. <http://www.starlab.vub.ac.be/website/dogmastudio>

¹⁶ Eclipse is an open development platform, which supports Java language (<http://java.sun.com/>). It is mainly used for enterprise development, embedded device development, rich client platform, application frameworks and language IDE. <http://www.eclipse.org/>

¹⁷ MySQL is a multithreaded, multi-user SQL database management system (DBMS). <http://www.mysql.com/>

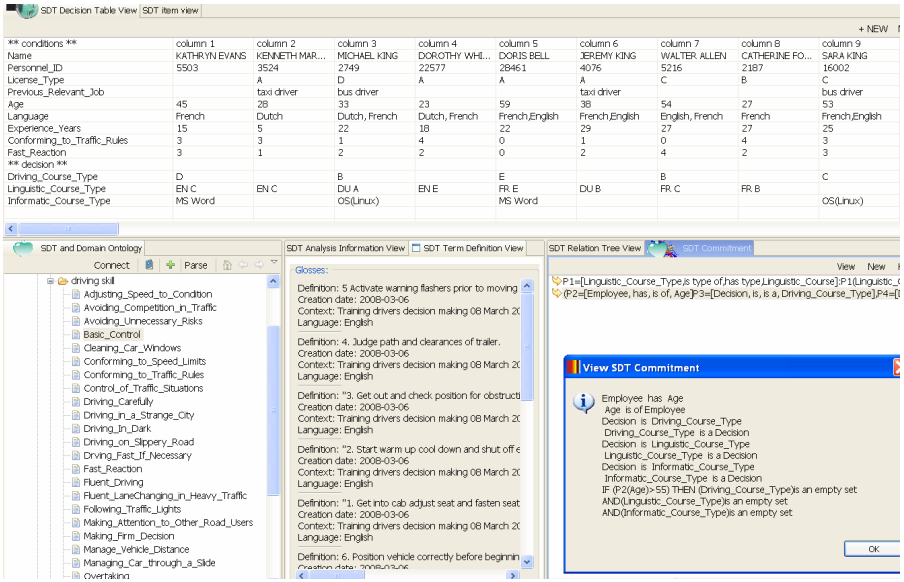


Fig. 3. SDT SOAR Plug-in screenshot

view of the domain ontologies, e.g. the ontology of HRM of drivers, with which the SDT is built. The bottom view in middle is the concept definitions categorized in glosses. The concept definition view gives the definitions when a concept in the ontology tree is selected (see ‘Basic_Control’ in Fig. 3). The bottom views in the right corner are the views of formal SDT commitments and SDT commitments in pseudo natural language. Users can add a new commitment and visualize its verbalization. For example, the window with the title “View SDT Commitment” in Fig. 3 shows a new rule “if the age of an employee is more than 55, then he doesn’t need to take any courses”. SDT-SOAR will automatically check and regenerate the SDTs at different levels, such as shown in Fig. 2, when the new rule is added.

In this section, we focus on how the procedures in SOAR are designed and implemented. In the next section, we present experimental analysis of SOAR.

4 Experimental Analysis

We have conducted several experimentations to evaluate SOAR.

The experimental setup is as follows: We use Intel(R) Pentium(R) processor 1500MHZ with 2 GB memory running Microsoft Windows XP professional version 2002 with Service Pack 2. We implement SDT-SOAR using JRE 1.6.0_02. The employee information is stored in MySQL Sever of version 5.2.

Fig. 4 shows the cost in milliseconds for generating SDTs from the local database. We increase the SDT size by adding its decision columns. In our problem settings, every decision column in an SDT corresponds to an employee. The more employees are selected from the local database, the bigger the resulting SDT becomes. The

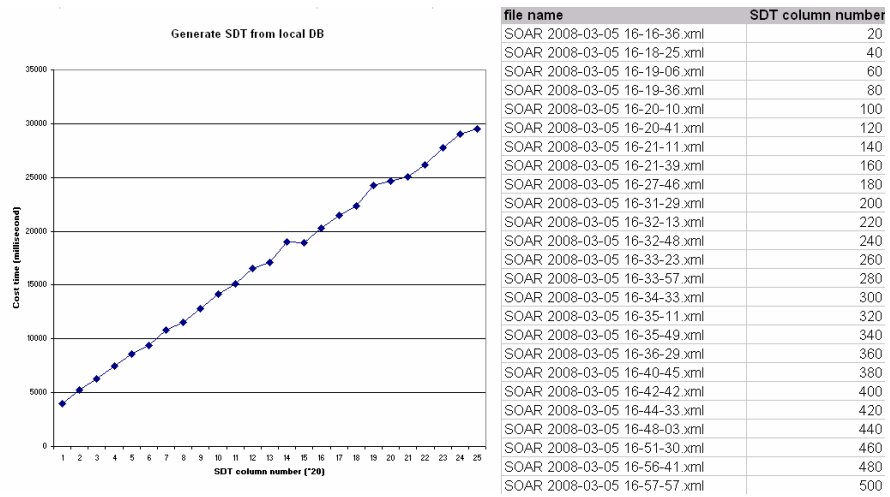


Fig. 4. Cost of generating SDTs from local DB

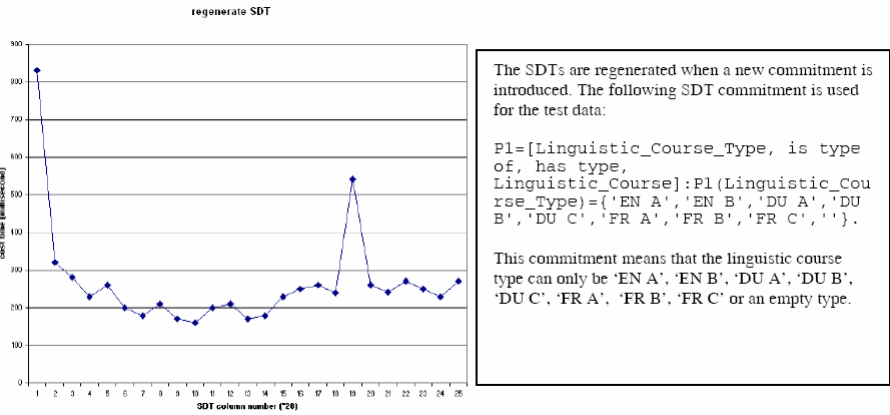


Fig. 5. Cost of regenerating and reorganizing SDTs

generated SDTs are stored as XML files (see the table on the right hand in Fig. 4). The minimum size of the XML file is 18.6 KB (SOAR 2008-03-05 16-16-36.xml), which is generated in 4005 milliseconds. The maximum XML file size is 180 KB (SOAR 2008-03-05 16-57-57.xml) generated in 29533 milliseconds. It increases linearly when the SDT sizes up gradually.

Fig. 5 illustrates the cost in milliseconds while regenerating and reorganizing SDTs. Once a user introduces a new commitment, such as shown in the figure, SDT SOAR checks the consistency in an SDT. The inconsistent decision columns are removed. The cost shown in Fig. 5 has is an irregular line, which means that the cost of regenerating and reorganizing an SDT does not depend on the size of the original SDT.

5 Related Work and Discussion

In the past, decision tables mainly used for computer programming can be found in many literatures, such as [4, 12, 18]. The application area of decision tables have been gradually moved from computer programming to many other domains during the last 50 years. A renewed research interest of decision tables focuses on the construction of the table itself [22]. As Vanthienen indicated, the application field of decision tables is enlarged into knowledge engineering, especially in the contexts of verification and validation of knowledge based systems, efficient execution of knowledge based systems, knowledge base maintenance, knowledge acquisition and knowledge discovery.

The approach of this paper is in the application area of knowledge validation and knowledge discovery. We focus on the discussion of the self organization and automatic reorganization of Semantic Decision Tables (SDTs). A similar solution is SORCER introduced in [10]. SORCER is a learning system that induces *second-order* decision tables from a given data set. Each entry (a condition entry or a decision entry) of a *first-order* decision table corresponds to a single value; while each entry of a *second-order* decision table is a value set. The authors in [10] tend to enhance comprehensibility of a decision tables by transforming a first-order decision table into a second-order decision table. By doing so, they can also reduce the table size without losing the decision rules. Our approach goes further than their work. We call both a first-order decision table and a second-order decision table as ‘traditional’ decision tables. For example in Fig. 2, the table on the left hand is a first-order decision table and the table on the right hand is a second-order table. The work in [10] is restricted to the transformation of these two kinds of tables. We provide a more generic transformation algorithm described in the SOAR procedures. Moreover, the work in [10] only uses the ‘if-then-else’ deduction rules for the transformation. We use various constraints, such as the mandatory constraint, the subset constraint and the exclusion constraint¹⁸, for the transformation. Similar debates can be applied to the approaches that are similar to SORCER, such as [9, 11].

Another interesting approach similar to ours is using decision tables in a decision table based development framework of decision support system [22, 23]. Decision tables are automatically created data patterns. We share the same comprehension of that fact that the decision logics behind a decision table are the key issues in the automatic decision table generation. The methods in [22, 23] use various classification techniques while generating the decision tables. For instance, classical neural networks, machine learning and classification tree algorithm. The generating rules of an applied domain are keyword (or label) based. Therefore, the resulting decision tables are not always accurate. In this paper, the semantics of SDT are from both the decision logics and the constraints in the domain ontologies. An ontology, by definition, deals with the concepts and their relations in a domain instead of the keywords. It has been proven that an ontology-based system can dramatically increase the accuracy of a process result, e.g. key words searching versus ontology-based searching [15]. Therefore, we argue that SOAR procedures in this paper, which are ontology based, can increase the accuracy of the generated decision tables.

¹⁸ Note that those constraints are not used at the level of database but at the level of the decision table.

Comparing to all the research efforts of the related work listed above, SDT has many basic yet important characteristics provided by modern ontology engineering. An SDT is the result of annotating (a set of) decision table(s) with a domain ontology. It can be stored in computers (e.g. the SDT xml files shown in Fig. 4) and is *explicit*, *sharable*, *formal* and *conceptual*. Comparing to a traditional decision table, an SDT contains richer decision rules. All the verification and validation rules of a decision table are specified in the form of ontological commitments of SDTs. Based on the constraints in the SDT commitments, SOAR ensures the *precision* of the resulting re-organized SDTs.

In this paper, we use SDTs to learn rules from data and match new rules with existing data. Another simple yet important understanding of SDTs is as follows. An SDT can also be considered as a decision table with appropriate semantics in order to define the decision logic in a modeling setting. In this case, we don't need to involve the database or actual case as we do in this paper.

A disadvantage of SDT might be its dependency on the availability of the domain ontology. According to our experience, to create an ontology costs a lot of time. For example, we used to spend six man months to create a HRM (Human Resource Management) ontology based on O*NET¹⁹ in PoCehrMOM Project²⁰. Therefore, SDT is feasible when one of the following conditions is satisfied: 1) there exist domain ontologies; 2) there exists formal knowledge documentations, which can be easily converted into an ontology; 3) the domain is rather small.

6 Conclusion and Future Work

In this paper, we focus on the discussion of Semantic Decision Tables (SDTs) as self-organizing and reorganizable decision tables. SOAR is developed as a collection of the adaptive **S**elf-**O**rganization and **A**utomatic **R**eorganization procedures used for SDT. SOAR is *precise*, *simple* and *flexible*. While reorganizing an SDT, SOAR contains the consistency checking based on the constraints in the SDT commitments. We introduce 7 constraints and 4 logical operators mainly used in formal SDT commitments. SOAR ensures the *precision* of the process of self-organization and reorganization by always satisfying these constraints. In our current projects (e.g. the EC Prolix project), we observe that it's rather easy to implement SOAR because the algorithm used in the SOAR procedures is rather *simple* (see the pseudo code in Fig. 1). The reasoning logics of SDTs are often layered. For example, the SDT on the right hand in Fig. 2 explains the SDT on the left hand in Fig. 2. The latter SDT represents part of the reasoning logics of Table 2. By using SOAR, end users can visualize SDTs at different levels. We call it *visualization flexibility*.

SOAR is implemented as a tool called *SDT-SOAR*. We have conducted several experiments to evaluate SDT-SOAR. The cost of generating an SDT from local database

¹⁹ O*NET provides a full-access, online version of the occupational network database. <http://online.onetcenter.org/>

²⁰ PoCehrMOM Project (Project omtrent Competenties en functies in e-HRM voor technologische toepassingen op het Semantisch Web door Ontologie en Meertalige terminologie). The project is to use ontologies to enhance human resource management. <http://cvc.ehb.be/PoCeHRMOM/Frameset.htm>

server increases linearly when the size of the SDT grows. The cost of regenerating and reorganizing an SDT does not depend on the size of the original SDT.

Currently, the tool SDT-SOAR only supports a few constraints, such as the value constraint. In the future, we will implement all constraints discussed in the paper. One of our recent ongoing researches focuses on using RuleML²¹ to store and interchange the SDT commitments. Later on, we will add a new SDT-SOAR function, which reads RuleML as the input and generates SDTs as the output.

Acknowledgments. The research is partly supported by the EC Prolix project. It is authors' pleasure to thank all the STARLab members for the paper discussion.

References

1. Camps Paré, R.: From Ternary Relationship to Relational Tables: A Case against. Common Beliefs, SIGMOD Record 31(20) (2002)
2. Cavouras, J.C.: On the Conversion of Programs to Decision Tables: Method and Objectives. Commun. ACM 17(8), 456–462 (1974)
3. CSA, Z243.1-1970 for Decision Tables, Canadian Standards Association (1970)
4. Geesink, L.H., van Dijk, J.E.M.: The construction of decision tables in PROLOG. *Ange wandte Informatik archive* 30(7), 294–301 (1988)
5. Goelman, D., Song, I.-Y.: Entity-Relationship Modeling Re-revisited. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 43–54. Springer, Heidelberg (2004)
6. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: Workshop on Formal Ontology, Padua, Italy; In book Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers (1993)
7. Guarino, N., Poli, R.: Formal Ontology in Conceptual Analysis and Knowledge Representation. Special issue of the International Journal of Human and Computer Studies 43(5/6) (1995)
8. Halpin, T.: Information Modeling and Relational Database: from Conceptual Analysis to Logical Design. Morgan-Kaufmann, San Francisco (2001)
9. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: Proc. of the 21st international conference on very large databases (VLDB 1995), Zurich, Switzerland, pp. 420–431. Morgan Kaufman, San Francisco (1995)
10. Hewett, R., Leuchner, J.H.: The Power of Second-Order Decision Tables. In: Proc. of the Second SIAM International Conference on Data Mining, Arlington, VA, USA. SDM 2002, April 11-13, 2002. SIAM, Philadelphia (2002)
11. Kohavi, R.: The Power of Decision Tables. In: Lavrač, N., Wrobel, S. (eds.) ECML 1995. LNCS(LNAI), vol. 912, pp. 174–189. Springer, Heidelberg (1995)
12. Langenwarter, D.F.: Decision tables - an effective programming tool. In: Proc. of the first SIGMINI symposium on Small systems, pp. 77–85. ACM, New York (1978)

²¹ The Rule Markup Language (RuleML) is a markup language developed to store rules in XML. The Rule Markup Initiative has taken steps towards defining a shared Rule Markup Language (RuleML), permitting both forward (bottom-up) and backward (top-down) rules in for deduction, rewriting, and further inferential-transformational tasks. <http://www.ruleml.org/>

13. Sadri, F., Ullman, J.D.: Template dependencies: a large class of dependencies in Relational Databases and its complete approximatization. *Journal of the ACM (JACM)* 29(2), 363–372 (1982)
14. Sheth, A.: Data Semantics: What, Where and How? Database Applications Semantics. In: *Proc. of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)*, Stone Mountain, Atlanta, Georgia, USA, Chapman & Hall, Boca Raton (1996)
15. Sheth, A.P., Ramakrishnan, C.: Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis. *IEEE Data Engineering Bulletin*, *IEEE Data Engineering* 26(4), 40–48 (2003)
16. Smith, H., Fingar, P.: *Business Process Management: The Third Wave*, 1st edn. Meghan-Kiffer, USA (2002)
17. Spyns, P., Meersman, R., Jarrar, M.: Data Modeling versus Ontology Engineering. *SIGMOD Record: Special Issue on Semantic Web and Data Management* 31(4), 12–17 (2002)
18. Sterbenz, R.F.: Tabsol decision table preprocessor. *ACM SIGPLAN Notices archive* 6(8) (September 1971); special issue on decision tables, pp. 33 – 40, B.F. Goodrich Chemical Company, Cleveland, Ohio. ACM, New York (ISSN:0362-1340)
19. Tang, Y.: On Conducting a Decision Group to Construct Semantic Decision Tables. In: Meersman, R., Tari, Z., Herrero, P. (eds.) *OTM-WS 2007, Part I. LNCS*, vol. 4805, pp. 534–543. Springer, Heidelberg (2007)
20. Tang, Y., Meersman, R.: On constructing semantic decision tables. In: Wagner, R., Revell, N., Pernul, G. (eds.) *DEXA 2007. LNCS*, vol. 4653, pp. 34–44. Springer, Heidelberg (2007)
21. Tang, Y., Meersman, R.: Organizing Meaning Evolution Supporting Systems Using Semantic Decision Tables. In: Meersman, R., Tari, Z. (eds.) *OTM 2007, Part I. LNCS*, vol. 4803, pp. 272–284. Springer, Heidelberg (2007)
22. Vanthienen, J.: Ruling the business: about Business Rules, Decision Tables and Intelligent Agents. In: Vandenbulcke, J., Snoeck, M. (eds.) *New directions in Software Engineering*, pp. 103–120, 160. Leuven University Press, Leuven (2001)
23. Wets, G., Vanthienen, J., Mues, C., Timmermans, H.: Extracting complete and consistent knowledge patterns from data. In: van Harmelen, F. (ed.) *Proc. of Sixth International Conference on Principles of Knowledge Representation and Reasoning: V&V Workshop*, Trento, Italy (1998) ISSN 1613-0073

Translating SQL Applications to the Semantic Web

Syed Hamid Tirmizi, Juan Sequeda, and Daniel Miranker

Department of Computer Sciences, The University of Texas at Austin, USA
{hamid, jsequeda, miranker}@cs.utexas.edu

Abstract. The content of most Web pages is dynamically derived from an underlying relational database. Thus, the success of the Semantic Web hinges on enabling access to relational databases and their content by semantic methods. We define a system for automatic transformation of SQL DDL schemas into OWL DL ontologies. This system goes further than earlier efforts in that the entire system is expressed in first-order logic. We leverage the formal approach to show the system is complete with respect to a space of the possible relations that can be formed among relational tables as a consequence of primary and foreign key combinations. The full set of transformation rules is stratified, thus the system can be executed directly by a Datalog interpreter.

1 Introduction

It has been estimated that Internet accessible databases contain up to 500 times more data compared to the static Web and that three-quarters of these databases are managed by relational database management systems [HeP07]. Thus, enabling the integration of relational databases and their content with the Semantic Web is critical to the Semantic Web's success.

The Semantic Web provides an ontology-based framework for integration, search and sharing of data drawn from diverse sources. Broadly stated, there are two architectural approaches to integrating databases with the Semantic Web. The more commonly researched approach is the development of wrapper systems that map a relational database schema to an existing domain ontology [AnB05, Bar04, Che06, Lab05, Lab06, Rod06]. To date there has been little work automating the creation of such wrappers. Thus, wrapper systems appear to be a labor-intensive solution.

The second approach, which is the subject of the work in this paper, concerns the automatic transformation of database content and/or schema to a Semantic Web representation, i.e. RDF and OWL [Biz03, LiD05, Ast07]. In this approach it is assumed that the data model entails a logical model of the application domain, and by syntactically analyzing the model's physical encoding in SQL Data Description Language (DDL) the logical model may be recovered. While many legacy databases were defined using strict relational syntax and semantics, and thus may encode modest application domain semantics, the current SQL standard coupled with modern software design methodology enables rich expression of domain semantics; albeit not in a form readily accessible to automated inference mechanism [Seq07]. In addition to foreign key constraints, SQL DDL supports a variety of constraints on the range of values allowed in a table. Building on related work we define a system for automatic transformation of relational

databases into OWL-DL ontologies. Two critical elements distinguish our transformation system from past efforts. *First*, the entire system is defined in first order logic (FOL) eliminating syntactic and semantic ambiguities in our rules. Much of the related work is expository in nature, at times influenced by domain specific examples and/or specifying the resulting rules in English prose. Often the influence of examples from a particular domain can result in incorrect rules. *Second*, we present a notion of completeness of our system in terms of a space of all possible relations describable by SQL DDL considering the interactions of primary and foreign keys in relations. We have partitioned the space of relations and have covered the transformation of each partition with sets of rules applicable to that partition.

Further, we observe that the FOL expression of our transformation system is stratified. Thus, in addition to implementation in Prolog environments, the system may integrate with databases supporting Datalog interpreters.

2 Related Work

A number of researchers have made inroads on this problem and serve as a foundation for our work [Sto02, LiD05, Ast07].

Stojanovic et al. [Sto02] provide rules for translation of relational schemas to Frame Logic and RDF Schema. This work formally defines rules for identification of classes and properties in relational schemas. It does not have the capability of capturing richer semantics that cannot be expressed in RDF Schema.

Li et al. [LiD05] propose a set of rules for automatically learning an OWL ontology from a relational schema. They define the rules using a combination of some formal notation and English language. Our analysis shows that some of their rules miss some semantics offered by the relational schema and some rules produce specific results for inheritance and object properties that may not accurately depict concepts across domains or database modeling choices. We believe these shortcomings are due to lack of a formal system and thorough examination of examples capturing a variety of modeling choices in various domains.

Astrova et al. [Ast07] provide expository rules and examples to describe a system for automatic transformation of a relational schema to OWL Full. When it was published this work was the most comprehensive. Since the rules were not formally defined, a number of transformations are ambiguous.

In addition to the lack of correctness due to informal specification of rules, these systems do not provide any notion of completeness of their rules. By completeness we mean consideration of all possible database key structures that may encode an ontological relationship. We present the results of a construction that enumerates such key structures and document that our transformation system is complete and unambiguous. In the rest of the paper, first we present the disparities between relational databases and ontologies. Then we systematically present how relational database schemas can be transformed into OWL ontologies. First, with the help of an example, we show how a domain expert can translate a relational schema in SQL DDL into an OWL ontology. Then, we present our assumptions and transformation rules, and

explain them using the same example. We also provide a comparison of human and automatically generated ontologies and relate the differences using our discussion on disparities as a basis.

3 Extracting Knowledge from a Relational Schema

Consider a relational database for a university and its definition (see Table 1).

The *Person* table contains data about all the people, some of them may be students and present in *Student* table, and some may be professors and present in *Professor* table. The *Dept* table lists the departments in the university where each department has a unique name, and the *Course* table lists the courses for every department. The *Semester* table contains a list of semesters which have a year and one of the three seasons, Spring, Summer or Fall, associated with them. A course could be offered in a particular semester with a particular professor, and recorded in *Offer* table. Two offered courses could be co-offered, and recorded as a self-relation in the *Offer* table. A student could study an offered course, which is recorded in *Study* table. Also, a student could be registered in a semester with or without taking a course, and this information is recorded in the *Reg* table.

Table 1. Schema of a University Database

University Database Schema
<pre>create table PERSON { ID integer primary key, NAME varchar not null } create table STUDENT { ROLLNO integer primary key, DEGREE varchar, ID integer unique not null foreign key references PERSON(ID) } create table PROFESSOR { ID integer primary key, TITLE varchar, constraint PERSON_FK foreign key (ID) references PERSON(ID) } create table DEPT { CODE varchar primary key, NAME varchar unique not null } create table SEMESTER { SNO integer primary key, YEAR date not null, SESSION varchar check in ('SPRING', 'SUMMER', 'FALL') } create table COURSE { CNO integer primary key, TITLE varchar, CODE varchar not null foreign key references DEPT(CODE) } create table OFFER { ONO integer primary key, CNO integer foreign key references COURSE(CNO), SNO integer foreign key references SEMESTER(SNO), PID integer foreign key references PROFESSOR(ID), CONO integer foreign key references OFFER(ONO) } create table STUDY { ONO integer foreign key references OFFER(ONO), RNO integer foreign key references STUDENT(ROLLNO), GRADE varchar, constraint STUDY_PK primary key (ONO, RNO) } create table REG { SID integer foreign key references STUDENT(ID), SNO integer foreign key references SEMESTER(SNO), constraint REG_PK primary key (SID, SNO) }</pre>

For a domain expert, it is easy to recognize the concepts in this database structure, and to identify the semantics of their properties and different kinds of relationships that exist between these concepts. Table 2 shows an ontology corresponding to the given schema, developed by a domain expert.

Table 2. Parts of an ontology corresponding to the schema in Table 1, developed by a domain expert. The ontology is presented in OWL Abstract Syntax. The highlighted sections in the table are later compared with an automated output.

Domain Expert's Ontology
<pre> Ontology(<urn:sql2owl> ObjectProperty(<REG> domain(<STUDENT>) range(<SEMESTER>)) ObjectProperty(<REG_I> inverseOf(<REG>)) ObjectProperty(<OFFER.CONO> Transitive Symmetric domain(<OFFER>) range(<OFFER>))... DatatypeProperty(<COURSE.CNO> Functional domain(<COURSE>) range(xsd:integer)) DatatypeProperty(<SEMESTER.YEAR> Functional domain(<SEMESTER>) range(xsd:date)) DatatypeProperty(<SEMESTER.SESSION> Functional domain(<SEMESTER>) range(oneOf("SPRING" "SUMMER" "FALL")) range(xsd:string))... Class(<PERSON> partial ...) Class(<PROFESSOR> partial <PERSON> ...) Class(<STUDENT> partial <PERSON> restriction(<STUDY.RNO_I> minCardinality(0)) ...) Class(<COURSE> partial restriction(<COURSE.DEPTCODE> cardinality(1)) restriction(<COURSE.CNO> cardinality(1)) ...) ... </pre>

4 Disparities between Relational Databases and Ontologies

While relational databases are capable of efficiently managing large amounts of structured data, ontologies are very useful for knowledge representation. Since these two data models are aimed towards different requirements specified by their domains, it is reasonable to expect some disparities among them in terms of capabilities.

To define a relational database to ontology transformation system, it is important to understand the mismatches between the two data models, and to make educated choices when confronted with such problems. Here we discuss some key issues – inheritance modeling, property characteristics and open/closed world assumptions – that affect a transformation system.

4.1 Inheritance Modeling

Relational databases do not express inheritance. However, inheritance hierarchies can be modeled in a variety of ways in relational schemas. Also, some modeling choices are harder to identify automatically. Given a foreign key definition between two entities, we would like to know whether a subclass relationship exists between the entities involved. If such patterns exist, we can map them to subclass relationships in the ontology.

The following list presents possible foreign key patterns to model inheritance:

- Foreign key is also the primary key: The *Professor-Person* relationship in our university schema is an example. This pattern uniquely identifies inheritance. An exception to this would be vertical partitioning of tables, but since we assume 3NF databases for our system, this case can be transformed to inheritance.
- Foreign key and primary key are disjoint: The *Student-Person* relationship in our university schema is an example. However, the *Course-Dept* relationship modeled

in the same schema is a counterexample. In fact, this pattern is the most common one used for expressing one-to-many relationships.

- Foreign key is a subset of the primary key: This is also not a good candidate for automatic translation to an inheritance hierarchy in the ontology. Many counterexamples for this pattern can be presented, particularly the ones for modeling ‘part-of’ relationships between entities.

4.2 Characteristics of Relationships

While relational schemas can capture some cardinality constraints on relationships between entities by defining constraints on foreign keys, they lack the expressive power to define relationships with interesting logical characteristics, like symmetry and transitivity etc. On the other hand, expressing such characteristics of relationships is natural to ontology languages like OWL, which are based on some form of logic.

For example, the self-relation on the *Offer* entity, that represents co-location of an offered course with another offered course, is symmetric and transitive. While these characteristics are obvious to a domain expert, the relationship is expressed like any other self-relationship in the relational schema, which may not have the same characteristics. Consider the example: `Employee(ID, Name, MgrID)`, where `ID` is the primary key, and `MgrID` is a foreign key to the `Employee` table itself referencing manager’s `ID`. This relationship is clearly not symmetric, and may or may not be transitive.

The example clearly shows that it is hard to identify logical characteristics of relationships in a relational schema without using the domain knowledge. Therefore, our rules do not capture these characteristics automatically.

4.3 The Effect of Open/Closed World Assumptions

Relational databases usually operate under the closed world (CW) assumption. This means that whatever is not in the database is considered false. On the other hand, knowledge bases operate in open world (OW) where whatever is not in the knowledge base is considered unknown. This assumption is natural for knowledge bases that often contain incomplete knowledge, and grow incrementally.

Therefore, the concept of a constraint has very different meanings in the two worlds [Mot07]. In a database setting, a constraint is mainly used for validation. In contrast, in an ontology, a constraint expresses some characteristics of classes or relationships but does not prevent assertion of any facts. In addition, some assertions may even result in unintuitive inferences.

When developing an ontology based on a relational schema, it is very important to keep these differences in mind. The question whether the open world should be closed or not depends upon the domain and application requirements. In our system, we produce an ontology with open world assumption. If needed, one way to close the world will be to assert that all inferred classes are pair-wise disjoint.

5 Translating SQL to Semantic Web

In this section, we explain the transformation of a relational schema to an ontology. First we present our assumptions and explain the rationale behind them. Then, we list the predicates and functions we have defined to express transformation rules in first order logic. In the next section, we explain the transformations for data types, classes, properties and inheritance, and provide mapping tables or first order logic rules to formally define the transformations.

5.1 Assumptions

In order to translate a relational schema into an ontology, we make the following assumptions:

- *The relational schema, in its most accurate form, is available in SQL DDL.* Databases evolve due to changing application requirements. Such modifications are often reflected solely in the physical model, usually expressed in SQL DDL, making it the most accurate source for the structure of the database.
- *The relational schema is normalized, at least up to third normal form.* While all databases might not be well normalized, it is possible to automate the process of finding functional dependencies within data and to algorithmically transform a relational schema to third normal form [DuW99, Wan00].

5.2 Predicates and Functions

We have defined a number of predicates and functions to aid the process of defining transformation rules in first order logic.

There are two sets of predicates in our system. *RDB predicates* test whether an argument (or a set of arguments) matches a construct in the domain of relational databases. Such predicates are listed below:

$Rel(r)$	r is a relation; e.g. $Rel(PERSON)$ holds, $Rel(ID)$ does not
$Attr(x,r)$	x is an attribute in relation r ; e.g. $Attr(ID,PERSON)$ holds
$NN(x,r)$	x is an attribute (or a set of attributes) in relation r with NOT NULL constraint(s); e.g. $NN(NAME,PERSON)$ holds
$Unq(x,r)$	x is an attribute (or a set of attributes) in relation r with UNIQUE constraint; e.g. $Unq(\{NAME\},DEPT)$ holds
$Chk(x,r)$	x is an attribute in relation r with enumerated list (CHECK IN) constraint; e.g. $Chk(SESSION,SEMESTER)$ holds
$PK(x,r)$	x is the (single or composite) primary key of relation r ; e.g. $PK(\{ONO,RNO\},STUDY)$ holds; Also: $PK(x,r) \rightarrow Unq(x,r) \wedge NN(x,r)$
$FK(x,r,y,s)$	x is a (single or composite) foreign key in relation r and references y in relation s ; e.g. $FK(\{ID\},STUDENT,\{ID\},PERSON)$ holds
$NonFK(x,r)$	x is an attribute in relation r that does not participate in any foreign key; e.g. $NonFK(NAME,DEPT)$ holds

On the other hand, *ontology predicates* test whether an argument (or a set of arguments) matches a construct that can be represented in an OWL ontology. These predicates are:

$Class(m)$	m is a class
$ObjP(p,d,r)$	p is an object property with domain d and range r
$DTP(p,d,r)$	p is an data type property with domain d and range r
$Inv(p,q)$	when p and q are object properties, p is an inverse of q
$FP(p)$	p is a functional property
$IFP(p)$	p is an inverse functional property
$Crd(p,m,v)$	the (max and min) cardinality of property p for class m is v
$MinC(p,m,v)$	the min cardinality of property p for class m is v
$MaxC(p,m,v)$	the max cardinality of property p for class m is v
$Subclass(m,n)$	m is a subclass of class n

The constructs represented by ontology predicates are described as they appear in the rules mentioned in the upcoming sections of this paper.

We have also defined the following functions:

$fkey(x,r,s)$	takes a set of attributes x , relations r and s , and returns the foreign key defined on attributes x in r referencing s
$type(x)$	maps an attribute x to its suitable OWL recommended data type (we discuss data types in more detail in a later section)
$list(x)$	maps an attribute x to a list of allowed values; applicable only to attributes with a CHECK IN constraint, i.e. $Chk(x)$ is true

In addition to the predicates and functions listed above, we describe the concept of a *binary relation*, written *BinRel*, as a relation that only contains two (single or composite) foreign keys that reference other relations. Such tables are used to resolve many-to-many relationships between entities. Using RDB predicates, we formally define *BinRel* as follows:

Rule Set 1:

$$BinRel(r,s,t) \leftarrow Rel(r) \wedge FK(q,r,_,t) \wedge FK(p,r,_,s) \wedge p \neq q \wedge Attr(y,r) \wedge \neg NonFK(y,r) \wedge FK(z,r,_,u) \wedge fkey(z,r,u) \in \{fkey(p,r,s), fkey(q,r,t)\}$$

This rule states that a binary relation r between two relations s and t exists if r is a relation that has foreign keys to s and t , and r has no other foreign keys or attributes (each attribute in the relation belongs to one of the two foreign keys). Note that there is no condition that requires s and t to be different, allowing binary relations that have their domain equal to their range.

5.3 Transformation Rules and Examples

In this section we present rules and examples for transformation of a relational database to OWL ontology.

Producing Unique Identifiers (URIs) and Labels

Before we discuss the transformation rules, it is important to understand how we can produce identifiers and names for classes and properties that form the ontology.

The concept of globally unique identifiers is fundamental to OWL ontologies. Each class or property in the ontology must have a unique identifier, or URI. While it is possible to use the names from the relational schema to label the concepts in the ontology, it is necessary to resolve any duplications, either by producing URIs based on

fully qualified names of schema elements, or by producing them randomly. In addition, for human readability, RDFS labels should be produced for each ontology element containing names of corresponding relational schema elements. Due to lack of space, we have not used fully qualified names in our examples. When needed, we append a name with an integer to make it unique, e.g. ID1, ID2 etc.

Transformation of Data Types

Transformations from relational schemas to ontologies require preserving data type information along with the other semantic information. OWL (and RDF) specifications recommend the use of a subset of XML Schema types [XMLSch] in Semantic Web ontologies [OWLRef, RDFSem].

In Table 3 we present a list of commonly used SQL data types along with their corresponding XML Schema types. During transformation of data type properties, the SQL data types are transformed into the corresponding XML Schema types.

Table 3. Common SQL types and corresponding XML Schema types recommended for OWL

SQL Data Type	XML Schema Type	SQL Data Type	XML Schema Type
INTEGER	xsd:integer	VARCHAR	xsd:string
FLOAT	xsd:float	DATE	xsd:date
BOOLEAN	xsd:boolean	TIMESTAMP	xsd:dateTime

Identifying Classes

According to OWL Language Guide [OWLGde], “the most basic concepts in a domain should correspond to classes ...”. Therefore we would expect basic entities in the data model to translate into OWL classes.

Given the definition of a binary relation, it is quite straightforward to identify OWL classes from a relational schema. Any relation that is not a binary relation can be mapped to a class in an OWL ontology, as stated in the rule below.

Rule Set 2:

$$Class(r) \leftarrow Rel(r) \wedge \neg BinRel(r, \_, \_)$$

Remember that a binary relation has exactly two foreign keys and no other attributes (see Rule Set 1). Keeping that in mind, we can see that this very simple rule covers a number of cases for identifying classes:

- All tables that do not have foreign keys should be transformed to classes. Therefore, we conclude *Class(PERSON)*, i.e. *Person* should be mapped to a class since it has no foreign key. The same reasoning holds for the *Dept* and *Semester* tables.
- All tables with one foreign key can be mapped to classes since they cannot be binary relations. Hence *Student*, *Professor* and *Course* should be mapped to classes.
- Tables with more than two foreign keys should be transformed to classes as well. Such tables may represent an entity or an N-ary relationship between entities. Fortunately, in OWL, both the cases can be modeled the same way, i.e. by translating the entity or the N-ary relationship into a class [Noy06]. From our example, *Offer* represents an N-ary relationship, and modeled as a class using the given rule.

- For tables containing exactly two foreign keys, presence of independent attributes qualifies them to be translated to classes. The table *Study*, with an independent attribute *Grade*, is an example, and is translated to an OWL class.

Thus Rule Set 2 identifies the OWL classes from the database schema. For example:

Class(PERSON), Class(STUDENT), Class(DEPT), Class(STUDY), Class(OFFER)

Identifying Object Properties

An object property is a relation between instances of two classes in a particular direction. In practice, it is often useful to define object properties in both directions, creating a pair of object properties that are inverses of each other. OWL provides us the means to mark properties as inverses of each other. In our work, when we translate something to an object property, say *ObjP(r,s,t)*, it implicitly means we have created an inverse of that property, which we write as *ObjP(r',t,s)*.

There are two ways of extracting OWL object properties from a relational schema. One of the ways is through identification of binary relations, which represent many-to-many relationships. The following rule identifies an object property using a binary relation.

Rule Set 3:

$$ObjP(r,s,t) \leftarrow BinRel(r,s,t) \wedge Rel(s) \wedge Rel(t) \wedge \neg BinRel(s,_,_) \wedge \neg BinRel(t,_,_)$$

This rule states that a binary relation *r* between two relations *s* and *t*, neither being a binary relation, can be translated into an OWL object property with domain *s* and range *t*. Notice that the rule implies *Class(s)* and *Class(t)* hold true, so the domain and range of the object property can be expressed in terms of corresponding OWL classes.

From our university database schema, the *Reg* table fits the condition. *Reg* is a binary relation between *Student* and *Semester* entities, which are not binary relations. Therefore, *ObjP(REG,STUDENT,SEMESTER)* holds, and since we can create inverses, *ObjP(REG',SEMESTER,STUDENT)* and *Inv(REG,REG')* also hold true.

Foreign key references between tables that are not binary relations represent one-to-one and one-to-many relationships between entities. A pair of object properties that are inverses of each other and have a maximum cardinality of 1 can represent one-to-one relationships. Also, one-to-many relationships can be mapped to an object property with maximum cardinality of 1, and an inverse of that object property with no maximum cardinality restrictions.

In OWL, a property with min cardinality of 0 and max cardinality of 1 is called *functional* which we represent by the functor *FP*. If an object property is functional, then its inverse is *inverse functional*, represented by the functor *IFP*. In addition to specifying cardinality restrictions on properties in general, we can also specify such restrictions when a property is applied over a particular domain. In our rules, we use ontology predicates *Crd*, *MinC* and *MaxC* to specify these restrictions. The examples following the rules explain the use of these predicates.

The following rule set identifies object properties and their characteristics using foreign key references (not involving binary relations, covered in Rule Set 3) with various combinations of uniqueness and null restrictions. To simplify the rules, we first define a predicate *NonBinFK* that represents foreign keys not in or referencing binary relations and then express the rules in terms of this predicate.

Rule Set 4:

$$NonBinFK(x,s,y,t) \equiv FK(x,s,y,t) \wedge Rel(s) \wedge Rel(t) \wedge \neg BinRel(s, \_, \_) \wedge \neg BinRel(t, \_, \_)$$

- a.
$$\begin{array}{l} ObjP(x,s,t), FP(x), \\ MinC(x',t,0) \end{array} \leftarrow NonBinFK(x,s,y,t) \wedge \neg NN(x) \wedge \neg Unq(x)$$
- b.
$$\begin{array}{l} ObP(x,s,t), FP(x), \\ Crd(x,s,1), MinC(x',t,0) \end{array} \leftarrow NonBinFK(x,s,y,t) \wedge NN(x) \wedge \neg Unq(x)$$
- c.
$$ObjP(x,s,t), FP(x), FP(x') \leftarrow NonBinFK(x,s,y,t) \wedge \neg NN(x) \wedge Unq(x)$$
- d.
$$\begin{array}{l} ObjP(x,s,t), FP(x), \\ Crd(x,s,1), FP(x') \end{array} \leftarrow NonBinFK(x,s,t) \wedge NN(x) \wedge Unq(x) \wedge \neg PK(x,s)$$

Each rule in Rule Set 4 states that a foreign key represents an object property from the entity containing the foreign key (domain) to the referenced entity (range). Since a foreign key references at most one record (instance) of the range, the object property is functional. This entails that the inverse of that object property is inverse functional. An example is the foreign key from *Study* to *Student* which gives us: $ObjP(RNO, STUDY, STUDENT), FP(RNO), Inv(RNO', RNO), IFP(RNO')$.

Rules 4a and 4b represent variations of one-to-many relationships.

- We can apply a stronger restriction on cardinality of the object property if the foreign key is constrained as NOT NULL. Without this constraint (rule 4a), the minimum cardinality is 0, which is covered by functional property predicate. With this constraint (rule 4b), we can set the maximum and minimum cardinality to 1.
- According to these rules, we can infer only the minimum cardinality restriction of 0 on the inverse property. Since an instance in the range could be referenced by any number of instances in the domain, we cannot apply a maximum cardinality restriction on the inverse property.

The other two rules, 4c and 4d, represent one-to-one relationships, modeled by applying a uniqueness constraint on the foreign key. It means that an instance in the range can relate to at most one object in the domain, making the inverse property functional too. This also means that the original object property is inverse functional as well.

The difference between rules 4c and 4d is that of a NOT NULL constraint that, like one-to-many relationships mentioned above, if present, gives us a stronger cardinality restriction on the object property represented by the foreign key.

Notice that none of the rules allow the foreign key to be the same as the primary key of the domain relation. Rule 4d restricts this by providing an extra condition, whereas the negation of uniqueness or NOT NULL constraints in rules 4a-c, by definition, implies this condition.

Examples of object properties and their characteristics obtained from the relational schema by applying Rule Sets 3 and 4 are:

$$\begin{array}{l} ObjP(REG, STUDENT, SEMESTER), ObjP(REG', SEMESTER, STUDENT), Inv(REG, REG') \\ ObjP(RNO, STUDY, STUDENT), FP(RNO), IFP(RNO'), MinC(RNO', STUDENT, 0) \\ ObjP(ID1, STUDENT, PERSON), FP(ID1), FP(ID1'), Crd(ID1, STUDENT, 1) \end{array}$$

Identifying Data Type Properties

Data type properties are relations between instances of classes with RDF literals and XML Schema data types. Like object properties, data type properties can also be

functional, and can be specified with cardinality restrictions. However, unlike object properties, OWL DL does not allow them or their inverses to be inverse functional.

Attributes of relations in a database schema can be mapped to data type properties in the corresponding OWL ontology. Rule Set 5 identifies data type properties.

Rule Set 5:

- a. $DTP(x, r, type(x)), FP(x) \leftarrow NonFK(x, r)$
- b. $DTP(x, r, type(x)), FP(x), Crd(x, r, 1) \leftarrow NonFK(x, r) \wedge NN(x, r)$
- c. $DTP(x, r, type(x) \cap list(x)), FP(x) \leftarrow NonFK(x, r) \wedge Chk(x, r)$

Rule Set 5 says that attributes that do not contribute towards foreign keys can be mapped to data type properties with range equal to their mapped OWL type. Since each record can have at most one value per attribute, each data type property can be marked as a functional property. When an attribute has a NOT NULL constraint, rule 5b allows us to put an additional cardinality restriction on the property. Rule 5c allows us to infer stronger range restrictions on attributes with enumerated list (CHECK IN) constraints.

Table 4. Parts of an ontology corresponding to the University Database, produced automatically using our transformation rules. The output format is OWL Abstract Syntax. The underlined parts highlight the differences compared to the human-developed ontology shown in Table 2.

Automatically Produced Ontology
<pre> Ontology(<urn:sql2owl> ObjectProperty(<REG> domain(<STUDENT>) range(<SEMESTER>)) ObjectProperty(<REG_I> inverseOf(<REG>)) ObjectProperty(<OFFER.CNO> <u>Functional</u> domain(<OFFER>) range(<OFFER>)) ObjectProperty(<OFFER.CNO_I> <u>InverseFunctional</u> inverseOf(<OFFER.CNO>)) ObjectProperty(<STUDENT.ID> <u>Functional InverseFunctional</u> domain(<STUDENT>) range(<PERSON>)) DatatypeProperty(<COURSE.CNO> Functional domain(<COURSE>) range(xsd:integer)) DatatypeProperty(<SEMESTER.YEAR> Functional domain(<SEMESTER>) range(xsd:date)) DatatypeProperty(<SEMESTER.SESSION> Functional domain(<SEMESTER>) range(oneOf("SPRING" "SUMMER" "FALL"))) range(xsd:string)) ... Class(<PERSON> partial ...) Class(<PROFESSOR> partial <PERSON> ...) Class(<STUDENT> partial <u>restriction(<STUDENT.ID> cardinality(1))</u> <u>restriction(<STUDY.RNO_I> minCardinality(0))</u> ...) Class(<COURSE> partial restriction(<COURSE.DEPTCODE> cardinality(1)) restriction(<COURSE.CNO> cardinality(1)) ...) ... </pre>

In some cases, it may be possible to apply more than one rule to an attribute. In such cases, all possible rules should be applied to extract more semantics out of the relational schema. Some data type properties extracted from our sample university database schema are:

$DTP(ID1, PERSON, xsd:integer), FP(ID1), Crd(ID1, PERSON, 1)$
 $DTP(SESSION, SEMESTER, xsd:string \cap \{SPRING, SUMMER, FALL\}), FP(SESSION)$
 $DTP(NAME1, PERSON, xsd:string), FP(NAME1), Crd(NAME1, PERSON, 1)$

Identifying Inheritance

Inheritance allows us to form new classes using already defined classes. It relates a more specific class to a more general one using subclass relationships [OWLGde].

Inheritance relationships between entities in a relational schema can be modeled in a variety of ways. Since most of these models are not limited to expressing inheritance alone, it is hard to identify subclass relationships.

The following rule describes a special case that can be used only for inheritance modeling in a normalized database design.

Rule Set 6:

$$\text{Subclass}(r,s) \leftarrow \text{Rel}(r) \wedge \text{Rel}(s) \wedge \text{PK}(x,r) \wedge \text{FK}(x,r,s)$$

This rule states that an entity represented by a relation r is a subclass of an entity represented by relation s , if the primary key of r is a foreign key to s . In our sample university schema, we can clearly identify that $\text{Subclass}(\text{PROFESSOR}, \text{PERSON})$ holds.

As a result of applying our rules on the given relational schema, we get the ontology shown in Table 4

A comparison of the ontologies produced by the domain expert (Table 2) with the one produced automatically using our rules (Table 4) shows a number of differences. For example, our rules are unable to capture the subclass relationship of *Student* with *Person*, or the symmetric and transitive characteristics of the co-location relationship among *Offer* instances. These examples clearly show that automatic translation of a relational schema to an ontology has some limitations, and that these limitations are inline with the disparities we have identified earlier.

5.4 Implementation

The FOL expression of our transformation system is stratified enabling direct integration of the transformation system with databases supporting Datalog interpreters.

Theorem: *The transformation system defined by the union of rules in rule sets 1 through 6 is stratified.*

The proof is left to the reader. *Hint:* The predicates *BinRel* and *NonBinFK* are the only predicates that appear in both the head and body of a rule.

6 Completeness of Transformation

A notion of completeness of a SQL DDL to ontology transformation is that the rules of the transformation system cover the entire range of possible relations that can be described in a SQL schema. The interaction of the foreign keys with primary keys provides clues about the kinds of relationships that exist between the entities, e.g. one-to-one, one-to-many etc.

Theorem: *The space of relations describable in SQL DDL using various combinations of primary key and foreign key references between the relations can be partitioned into 10 disjoint cases of key combinations. Our transformation system covers the entire space of relations.*

The formal proof is beyond the space limits of this paper. The proof involves a syntactic enumeration of the cases and a closure operation over the space of relations. Fig. 1 provides a useful summary of the theorem and its proof.

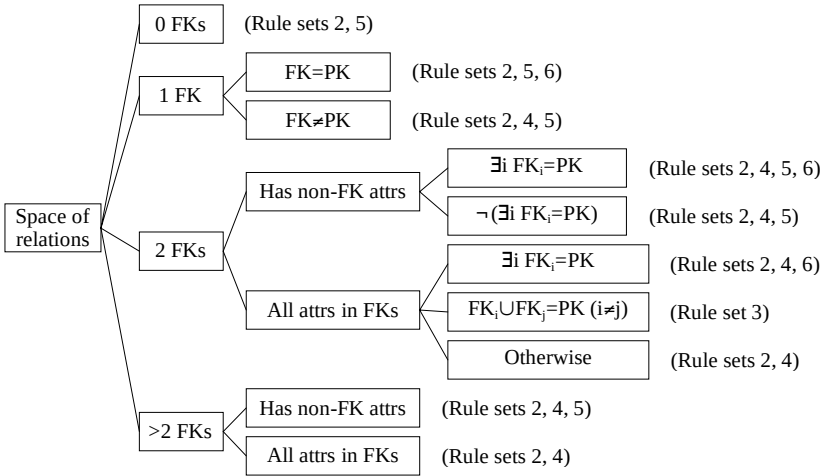


Fig. 1. The tree describes the complete space of relations when all possible combinations of primary and foreign keys are considered. For each branch, applicable rules are listed.

Briefly, we first partition the space by examining the number of foreign keys that a relation contains. All relations without any foreign keys can be easily translated into classes in an ontology. Similarly, relations with more than two foreign keys usually represent N-ary relationships, and the rules for N-ary relationships are applicable to them. The cases for one or two foreign keys are more interesting and give rise to more possibilities like binary relations, inheritance or new classes. However, for each possible branch, we have carefully defined sets of rules for producing ontology classes and properties.

7 Discussion

SQL DDL is a standard for representing the physical schema of applications that use relational databases. Although SQL DDL it is not a knowledge representation language, it is capable of capturing some semantics of the application domain. We have defined a system for automatic transformation of normalized SQL DDL schemas into OWL DL ontologies. We have defined our entire set of transformation rules in first order logic eliminating syntactic and semantic ambiguities and allowing for easy implementation of the system in languages like Datalog.

Once an ontology is defined for a domain represented by a relational schema, the actual database content can be easily translated into a corresponding RDF representation. We have also ensured compatibility with description logics based OWL DL, which is essential to assuring decidability for reasoning represented by the relational model.

We have demonstrated that an automatic transformation system has its deficiencies when it comes to identifying inheritance and other rich semantic elements. Although it is easy to generate specific examples of relational encodings of inheritance, there is

neither a unique encoding, nor an encoding whose syntax, without further qualification, can be strictly interpreted as inheritance. Thus, transformation systems that create inheritance relationships will incorrectly produce too many, or too few. Thus, there may always be an opportunity for human judgment to fill in gap between the expressive power of SQL DDL and OWL.

Independent of the issues that arise from the differences in expressive power, a fair criticism of the automated transformation approach, in general, is that the scope of success may be highly dependent on the amount of domain semantics captured in SQL DDL, which in turn correlates to the age of the database application and the sophistication of its developers. However, if the success of an application of an automated transformation is limited, it is still possible to add missing semantics using the techniques being developed in wrapper-based approaches. Such semi-automated systems have been explored in the context of strict relational data integration [BaM07, Mil00]. Further, functioning relational database applications are prone to schema modification. One can envision a system where an automated transformation bootstraps a more powerful wrapper system. In the advent of database schema evolution a combined system may be able to reason about and propagate the changes.

References

- [AnB05] An, Y., Borgida, A., Mylopoulos, J.: Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences. In: Proceedings of On The Move to Meaningful Internet Systems (2005)
- [Ast07] Astrova, I., Korda, N., Kalja, A.: Rule-Based Transformation of SQL Relational Databases to OWL Ontologies. In: Proceedings of the 2nd International Conference on Metadata & Semantics Research (October 2007)
- [Bar04] Barrasa, J., Corcho, O.: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372, Springer, Heidelberg (2005)
- [BaM07] Barbançon, F., Miranker, D.P.: SPHINX: Schema integration by example. *Journal of Intelligent Information Systems* (in press, available on-line SpringerLink)
- [Biz03] Bizer, C.: D2R MAP - A Database to RDF Mapping Language. In: Proceedings of the Twelfth International World Wide Web Conference (WWW) (2003)
- [Che06] Chen, H., Wang, Y., Wang, H., Mao, Y., Tang, J., Zhou, C., et al.: Towards a Semantic Web of Relational Databases: a Practical Semantic Toolkit and an In-Use Case from Traditional Chinese Medicine. In: Proc. of the 5th International Semantic Web Conference (2006)
- [DuW99] Du, H., Wery, L.: Micro: A normalization tool for relational database engineers. *Journal of Network and Computer Applications* 22(4), 215–232 (1999)
- [HeP07] He, B., Patel, M., Zhang, Z., Chang, K.C.: Accessing the deep web. *Communications of the ACM* 50(5), 94–101 (2007)
- [Hor03] Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. In: Proceedings of the 2nd International Semantic Web Conference (2003)
- [Lab05] de Laborda, C.P., Conrad, S.: Relational. OWL: a data and schema representation format based on OWL. In: Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modeling, vol. 43, pp. 89–96 (2005)

- [Lab06] de Laborda, C.P., Conrad, S.: Database to Semantic Web Mapping using RDF Query Languages. In: 25th International Conference on Conceptual Modeling (November 2006)
- [LiD05] Li, M., Du, X., Wang, S.: Learning ontology from relational database. In: Proceedings of the Fourth International Conference on Machine Learning and Cybernetics (2005)
- [Mil00] Miller, R., Haas, L.L., Hernández, M.: Schema mapping as query discovery. In: Proceedings of the VLDB Conference (2000)
- [Mot07] Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. In: Proceedings of the 16th International Conference on World Wide Web (2007)
- [Noy06] Noy, N., Rector, A. (eds.): Defining N-ary Relations on the Semantic Web. W3C Working Group Note (11/14/2007), <http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/>
- [OWLGde] Smith, M.K., Welty, C., McGuinness, D.L. (eds.): OWL Web Ontology Language Guide. W3C Recommendation /REC-owl-guide-20040210/> (11/15/2007) (2004), <http://www.w3.org/TR/>
- [OWLRef] Dean, M., Schreiber, G. (eds.): OWL Web Ontology Language Reference. W3C Recommendation (11/14/2007), <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
- [RDFSem] Hayes, P. (ed.): RDF Semantics. W3C Recommendation (11/26/2007), <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>
- [Rod06] Rodriguez, J.B., Gomez-Perez, A.: Upgrading relational legacy data to the semantic web. In: Proceedings of the 15th international Conference on World Wide Web (2006)
- [Seq07] Sequeda, J.F., Tirmizi, S.H., Miranker, D.P.: SQL Databases are a Moving Target. In: Position Paper for W3C Workshop on RDF Access to Relational Databases (October 2007)
- [Sto02] Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive web sites into the semantic web. In: Proceedings of the ACM Symposium on Applied Computing (2002)
- [Wan00] Wang, S., Shen, J., Hong, T.: Mining fuzzy functional dependencies from quantitative data. In: IEEE International Conference on Systems, Man and Cybernetics (October 2000)
- [XMLSch] Biron, P.V., Permanente, K., Malhotra, A. (eds.): XML Schema Part 2: Datatypes Second Edition. W3C Recommendation (11/26/2007), <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>

An Agent Framework Based on Signal Concepts for Highlighting the Image Semantic Content

Mohammed Belkhatir

Center for Multimedia Computing, Communications & Applications,
Monash University, Sunway Campus
Belkhatir.mohammed@infotech.monash.edu

Abstract. This paper addresses the image semantic gap (i.e. the difficulty to automatically characterize the image semantic content through extracted low-level signal features) by investigating the formation of semantic concepts (such as *mountains, sky, grass...*) in a population of **image agents**: abstract structures representing the image visual entities. Through the development of processes mapping extracted low-level features to concept-based visual information, our contribution is twofold. First, we propose a learning framework mapping signal (color, texture) and semantic concepts to highlight the image agents. Contrary to traditional architectures considering high-dimensional spaces of low-level extracted signal features, this framework addresses the **curse of dimensionality**. Then, at the image agent population level, the agents communicate about the perceived semantic concepts with no access to global information or to the representations of other agents, they only exchange conceptual information. While doing so they adapt their internal representations to be more successful at conveying the perceived semantic information in future interactions. The image content is therefore soundly inferred through these concept-based linguistic interactions.

The SIR¹ Agent prototype implements our theoretical framework and its architecture revolves around functional modules enabling the characterization of concept-based linguistic structures, highlighting the image agents and enforcing interactions and coordination between them.

Index Terms: Semantic-based Image Indexing & Retrieval, Multi-Agent Systems.

1 Introduction

In order to address the impossibility of the content-based image indexing and retrieval systems to characterize the image semantics (also called **semantic gap** [14]) two classes of automatic semantic extraction architectures have been prevalent in the literature.

The first, dealing with image categorization, operate at the global image level [3,6,13,15]. In [13], several experimental studies lead to the specification of 20 semantic categories or image scenes describing the image content at a global level (such

¹ Signal/Semantic integration for Image Retrieval.

as *group of people*, *cityscapes* and *landscapes*...). Each of these categories is then linked to several low-level features gathered within the **complete feature set**. The most recent automatic annotation models at the image global level are based on statistical approaches [3,6,15]. Blei and Jordan [3] extend Dirichlet's latent allocation model and propose a correlation model linking words and images. The latter is based on the hypothesis that a Dirichlet distribution can be used to generate a combination of latent factors (more than 200) which are then used to generate words and image regions. This parametric model is based on the Expectation-Maximization algorithm to estimate these latent factors. A model which has shown interesting performance improvements [6] is based on a doubly non-parametric approach, for which the probabilities are generated from each element of a training set. These models learn the joint probability of associating words to image features and use it to generate the probability of associating a word to a given query image. In [15], the image indexing approach is guided by the dependencies between annotating words represented by the hierarchy derived from a textual ontology. While the models above predict the probability of an annotating word given an image, one is interested in generating the set of all index words characterizing the image visual entities. Therefore, a second class of architectures, operating at the visual entity level, have been proposed in [4,10,16].

One of the early solutions presented a probabilistic framework based on estimating class likelihoods of local areas, labeled as either man-made vs. natural or inside vs. outside objects [4]. In [16], training sample regions of images are categorized into eleven clusters through a neural network mapping (e.g. *tree*, *fur*, *sand*...). To alleviate the restrained cardinality of the proposed previous sets of visual clusters, a richer index vocabulary consisting of 26 image labels called Visual Keywords (such as *sky*, *people*, *water*...) is specified in [10]. However, this solution relies on a query-by-example solution for querying and no language allowing the manipulation of the extracted semantics has been proposed. The main disadvantage of this second class of frameworks relies on the specification of restrained and fixed sets of semantic classes. Regarding the fact that several artificial objects have high degrees of variability with respect to signal properties such as color and texture variations, an interesting solution is to extend the extracted visual semantics with signal characterizations in order to enrich the image index vocabulary.

Accordingly, a new generation of systems integrating semantics and signal descriptions has emerged, which are based on a loosely-coupled association of textual annotations with a relevance feedback (RF) framework operating on low-level signal features [18].

The contribution of this paper is in the domain of automatic image semantic indexing through the specification of:

1. processes establishing a correspondence between extracted low-level features and concept-based visual (color, texture and spatial) information,
2. a learning framework based on support vector machines highlighting image agents, abstract structures representing visual entities within an image document.
3. an architecture enabling interactions and coordination between these agents in order to highlight the semantic content.

Let us note that at the core of our proposal is the notion of **image agents (IAs)** since their specification is an attempt to operate beyond simple low-level processes

[14] or semantic-based architectures considering the image global level [3,6,13,15]. Their interactions and coordination is indeed fundamental to highlight the semantic content at the visual entity level.

In the remainder, we deal in section 2 with the characterization of the concept-based signal structures. Section 3 details the learning framework highlighting image agents. Section 4 presents the architecture enabling interactions and coordination between these agents and section 5 covers our experimental instantiation.

2 From Low-Level Signal Features to Conceptual Description

2.1 Color Characterization

Our symbolic representation of color information is guided by the research carried out in color naming and categorization. Under the impulsion of Berlin and Kay, works have revolved around stressing a step of correspondence between color stimuli and ‘basic color terms’ [1] which they characterize by the following properties: their application is not restricted to a given object class, i.e. the color characterized by the term “olive color” is not valid; they cannot be interpreted conjointly with object parts, i.e. “the maple leaf color” is not a valid color; their interpretation does not overlap with the interpretation of other color terms and finally they are psychologically meaningful. Further works proposed in [8] consist of an experimental validation of the ‘basic color term’ notion in the HVC perceptive color space. The latter belongs to the category of user-oriented color spaces (as opposed to material-oriented spaces such as RGB), i.e. spaces which define color as being perceived by a human through tonality (describing the color wavelength), saturation (characterizing the quantity of white light in the color spectral composition) and brightness (related to color intensity). Given a series of perceptive evaluations and observations, eleven color concepts (black, blue, cyan, green, grey, orange, purple, red, skin, white, yellow) are highlighted, each described by tonality, brightness and saturation values.

Characterizing the aforementioned symbolic color concepts involves algorithmically transforming the extracted low-level features specified in the RGB space (primary step for low-level color extraction) to tonality, brightness and saturation values in the perceptually uniform HVC space. We detail this process in section 5.1.

2.2 Texture Characterization

The study of texture in computer vision has lead to the development of several computational models for texture analysis used in several content-based image retrieval architectures [9]. However, these texture extraction frameworks mostly fail to capture aspects related to human perception. Therefore, we propose a solution specifying a computational framework for texture extraction which is the closest approximation of the human visual system. The action of the visual cortex, where an object is decomposed into several primitives by the filtering of cortical neurons sensitive to several frequencies and orientations of the stimuli, is simulated by a bank of Gabor filters.

Although several works have proposed the identification of low-level features and the development of algorithms and techniques for texture computation, few attempts have been made to propose an ontology for texture symbolic characterization and

naming. In [2], a texture lexicon consisting of eleven high-level texture categories is proposed as a basis for symbolic texture classification. In each of these categories, a texture concept which best describes the nature of the characterized texture is proposed. We consider the following texture concepts as the representation of each of these categories: bumpy, cracked, disordered, interlaced, lined, marbled, netlike, smeared, spotted, uniform and whirly. We discuss the automatic mapping between the extracted low-level texture features and these concepts in section 5.1.

3 A Learning Framework Highlighting Multimedia Agents

Our architecture operating at the image agent level is characterized by a learning framework which maps signal color and texture concepts to semantic concepts.

It first tackles the issue related to the use of segmentation algorithms which strongly affect the performance of the indexing processes and are moreover ill-suited for processing image corpus of important sizes (>10K) due to their computational cost by considering a compact grid-based index representation.

It then deals with the curse of dimensionality which affects non-parametric models (such as density estimation kernels [3,6]) through the dimensionality reduction of the signal feature representation spaces. Indeed, since low-level signal features used in automatic indexing frameworks are of high dimensionality (typically in the order of 10^2 to 10^3) and data in high-dimension spaces are sparse, it is necessary to gather enough observations to make sure that the estimation is viable and therefore the convergence rate is low. Consequently, it is crucial to consider the dimensionality reduction of the signal feature representation spaces. Moreover, contrary to the state-of-the-art approaches for dimensionality reduction (such as PCS, MDS, SVD...) which are **opaque** (i.e. they operate dimensionality reduction of input spaces without making it possible to understand the signification of elements in the reduced feature space), our framework will itself be based on a **transparent** readable characterization. We propose to reduce the dimensionality of input signal features by taking into account a symbolic signal representation.

We consider a set SC of semantic concepts $c_{sem}[1] \dots c_{sem}[n_{sem}]$ and a knowledge base **K** consisting of annotated training objects (i.e. they correspond to a unique semantic concept with probability 1). Let an object **o** within this set, it is represented by a set of rectangular image regions $r_{ma} = \{r_1, \dots, r_n\}$ and is indexed by a semantic concept $c_{sem}[i]$, sets of color $\{c_{Col_1}, \dots, c_{Col_j}\}$ and texture $\{c_{tex_1}, \dots, c_{tex_k}\}$ concepts (where $j, k \leq 11$)

3.1 Formal Model

In order to highlight image agents and their associated semantic concepts, we consider applying on a new image (i.e. not indexed) a rectangular grid defining the $\{r_1, \dots, r_i, \dots\}$ image regions.

In order to determine which semantic concept is associated with a given image region, we have a set of points $\{x_1, \dots, x_i, \dots\}$ in an n -dimensional input space S^n of signal color and texture concepts (here $n=22$), a set of labels $\{y_1, \dots, y_i, \dots\}$ such that the y_i value equals 1 if x_i corresponds to semantic concept $c_{sem}[i]$ and -1 otherwise. The goal is to determine a function $f: S^N \rightarrow \{\pm 1\}$ which associates each point with its

corresponding label. This function shall provide good results on the training set and be capable of generalizing on images which are not semantically indexed. For this, we consider support vector machines which, for separable problems, are based on algorithms highlighting the unique optimal hyperplane discriminating the data among the class of hyperplanes. This approach is easily extended to non-linearly separable problems. The learning process consists in maximizing a function which considers the distance between each training data and class borders. The optimal position of a class border is obtained as a linear combination of training data within the border neighborhood: they are called support vectors. The latter play a crucial role in the learning process. In the case of non-linearly separable problems, projection kernels are used and support vector machines are then based on the resolution of the following optimization problem: $\min_{w,b,\phi} \frac{1}{2} w^T w + C \sum_{i=1}^l \phi_i$ subject to $y_i (w^T \psi(x_i) + b) \geq 1 - \phi_i$ and $\phi_i \geq 0$.

Here, training vector x_i is set in correspondence in a space of higher dimension (sometimes infinite) through the function ψ . Support vector machines then determine a separating linear hyperplane in this space. $K(x_i, x_j) = \psi(x_i)^T \psi(x_j)$ is the projection kernel and C the penalty parameter of the error term. Among the possible kernels (linear, polynomial, radial basis function, sigmoid...), we choose the radial basis function: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, $\gamma > 0$ where γ is a kernel parameter. It is traditionally used in the case of non-linearity between the class labels and the input attributes. It holds several advantages with respect to other kernels, in particular it requires fine-tuning less hyper-parameters and its computational complexity is reduced.

Support vector machines in their initial formulation are destined to discrimination problems involving two classes. We adopt a particular learning strategy to solve our multi-class problem called “one-against-rest” where a classifier is highlighted for each of the semantic concepts to optimize inter-class separation. However, this approach results in the specification of classifiers which generate a binary output. We would like to associate a confidence value for the proposed classification. For this, we consider the problematic of probabilistic estimation for these classifiers [12] and use a

logistic function of the form $P(y_i=1|f) = \frac{1}{1 + \exp(Af + B)}$ where f is the output of the

support vector machines for input x_i and $y_i = \pm 1$ represents the class label. This doubly parametric function allows linking outputs of support vector machines to corresponding posterior probabilities. This method implies solving a non-linear optimization problem involving the pair of parameters (A, B) such that $\sum_i t_i \log(p_i) + (1-t_i) \log(1-p_i)$ is mini-

mized; where $p_i = \frac{1}{1 + \exp(Af_i + B)}$ is the inferred posterior probability and $t_i = \frac{y_i + 1}{2}$ is

the target binary coding for the pair (x_i, y_i) .

3.2 Application

Once the learning framework has learned the visual vocabulary, the approach subjects an image to be indexed to a multi-scale, grid-based recognition against these semantic

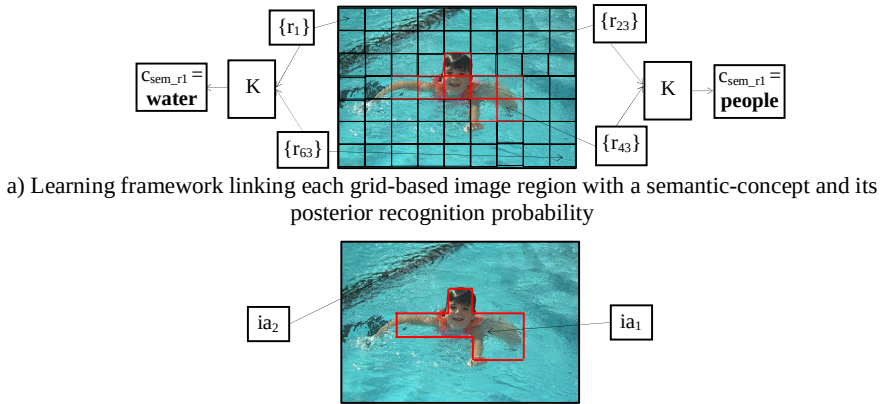


Fig. 1. Architecture for the highlighting of image agents and the characterization of their corresponding semantic concept

concepts. An image to be processed is scanned with grids of several scales. Each one features image regions $\{r_1, \dots, r_i, \dots\}$ characterized by a feature vector of signal color and texture concepts. The latter is compared against signal concept vectors of labeled image patches corresponding to semantic concepts in the knowledge base K (figure 1.a)). Recognition results for all semantic concepts are computed and then reconciled across all grid regions which are aggregated according to configurable spatial tessellation (figure 1.b)) in order to highlight image agents. Each agent is linked to a semantic concept with maximum recognition probability. Let us note that for an image agent, other semantic concepts with non-zero recognition probabilities could be highlighted. We therefore link the latter to a vector structure Sem with n_{sem} elements corresponding to all semantic concepts considered. Values $Sem[i]$, $i \in [1, n_{sem}]$ are real values in the interval $[0, 1]$ corresponding to the recognition probabilities for each of the semantic concepts. We can have cases where recognition probabilities for two distinct semantic concepts are equal or very close. Therefore, to make a decision on the semantic concept to index the image agent, we take into account the global distribution of objects within the documents in the knowledge base. In particular, the relations between them will help us reinforce the recognition values of semantic concepts for a given image agent. The framework instantiating this paradigm is a multi-agent system enforcing interaction and coordination between the agents detailed in the next section.

4 A System Enforcing Interactions and Coordination between the Multimedia Agents

4.1 Specifying the Agents

The agent-based system uses two types of agents: the image agents and the mediator agents. On an individual level, the image agents all have the ability to perceive color

and texture low-level features, to conceptualize their perception through highlighting color and texture concepts and to infer a set of semantic concepts with their recognition probability values (as detailed in section 3).

The mediator agents are responsible for determining the relational context between two image agents ia1 and ia2 and to inspect the knowledge base for highlighting identical relational configurations and their associated semantic concepts. They are then responsible for suggesting ia1 and ia2 to reinforce recognition values for the semantic concepts involved. On the population level, the image agents communicate with each other through mediator agents about a visual relational context and adapt to other agents in order to infer their index semantic concept, i.e. the semantic concept with the highest final recognition value.

4.1 Towards Concept-Based Interaction

When an agent is to communicate about the world, a symbolic representation of the perception is needed. Two interacting image agents communicate through a mediator agent with semantic concepts.

Mediator agents compute **spatial configurations** between image agents, i.e. compact structures summarizing spatial relations which hold between them. In order to model the conceptual (spatial) information, we first consider a subset of the topological relations highlighted in the RCC-8 theory [5]; four relations which are exhaustive and relevant to our context are chosen. Considering 2 image agents (ia1 and ia2), these relations are ($s_1=\mathbf{C}$, ia1, ia2): 'ia1 partially covers (in front of) ia2', ($s_2=\mathbf{C\_B}$, ia1, ia2): 'ia1 is covered by (behind) ia2', ($s_3=\mathbf{P}$, ia1, ia2): 'ia1 is a part of ia2', ($s_4=\mathbf{T}$, ia1, ia2): 'ia1 touches ia2 (is externally connected)' and ($s_5=\mathbf{D}$, ia1, ia2): 'ia1 is disconnected from ia2'. Directional relations Right($s_6=\mathbf{R}$), Left($s_7=\mathbf{L}$), Above($s_8=\mathbf{A}$), Below($s_9=\mathbf{B}$) are invariant to basic geometrical transformations (translation, scaling). Two relations specified in the metric space are based on the distances between visual objects. They are the Near($s_{10}=\mathbf{N}$) and Far($s_{11}=\mathbf{F}$) relations. We will discuss their empirical automatic characterization in section 5.3. A spatial configuration is supported by a vector structure **Sc** with eleven elements corresponding to the previously introduced spatial relations. Values $Sc[i]$, $i \in [1,11]$ are booleans stressing that the spatial relation s_i links the two considered image agents.

4.2 Enabling Interaction and Coordination

For the communication protocol, two image agents ia1 and ia2 are randomly chosen. They are respectively linked to two vector structures Sem1 and Sem2 with n_{sem} elements corresponding to all semantic concepts considered. Values $Sem1[i]$ and $Sem2[j]$ ($i, j \in [1, n_{\text{sem}}]$) are real values in the interval $[0,1]$ corresponding to the recognition probabilities for each of the semantic concepts.

The image agent ia1 starts the protocol by establishing a contact with a mediator agent and communicating its Sem1 structure, i.e. the semantic concepts with non-zero recognition probabilities linked to it (represented by step 1 in fig. 2). The image agent ia2 interacts with the same mediator agent by communicating its Sem2 structure (represented by step 2 in fig. 2).

The mediator agent then computes the spatial configuration between ia1 and ia2 (step 3 in fig. 2) and inspects spatial configurations between pairs of annotated objects within documents in the knowledge base resembling the spatial configuration between ia1 and ia2 (step 4 in fig. 2).

A reinforcement value which is proportional to the number of occurrences of spatial configurations within the knowledge base involving semantic concepts $Sem1[i]$ and $Sem2[j]$ ($i, j \in [1, n_{sem}]$) is then proposed to ia1 (step 5 in fig. 2) and ia2 (step 6 in fig. 2). Let us note that spatial configurations between annotated objects in the knowledge base are pre-computed and stored in lookup tables inspected by the mediator agents.

Image agents ia1 and ia2, respectively linked to vectors of semantic concepts $Sem1$ and $Sem2$, shall finally combine for each of the concepts $Sem1[i]$ and $Sem2[j]$ respectively ($i, j \in [1, n_{sem}]$) (i) their recognition probability values (ii) a reinforcement value proportional to the number of spatial configurations between two objects annotated by $Sem1[i]$ and $Sem2[j]$ within a document of the knowledge base resembling the spatial configuration between ia1 and ia2. We have chosen to use fuzzy sets, which are a generalization of set theory with a membership relation transformed in a function with values in the interval $[0,1]$. The membership function of an element with respect to a fuzzy set A of a universe U , noted μ_A associates each element of the universe with the plausibility that it is an element of A [7]. Let U_{SC} the (discrete) universe of all semantic concepts. In this universe, we define fuzzy sets by taking into account the recognition probability of two given semantic concepts and the relational (spatial) information holding between them. We define a membership function characterizing the plausibility for a given semantic concept $sc \in U_{SC}$ to be the index concept of an image agent by adding (i) and (ii).

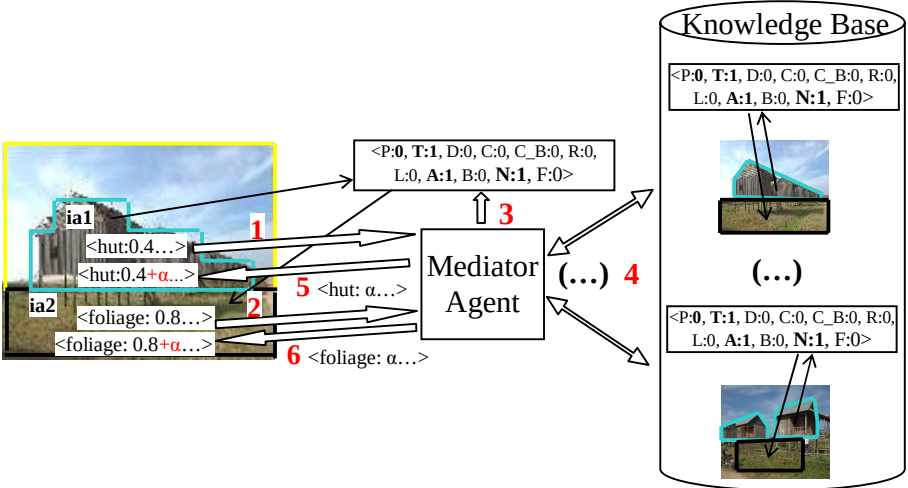


Fig. 2. Communication protocol between two image agents and a mediator agent

5 Experimental Instantiation

The SIR_Agent prototype implements our theoretical framework and its organization revolves around five functional modules:

1. the first provides the extraction of the signal-based content (color, texture and spatial features) and its mapping into concept-based linguistic structures.
2. the second supports the learning framework for highlighting the image agents.
3. the third consists of the architecture enabling interaction and coordination between the image agents.
4. the fourth module is the knowledge base comprising all documents annotated at the visual object level.
5. the communication interface allows the user formulating his query through semantic concepts.

In the remainder, we will detail the automatic characterization of our framework and its experimental validation.

5.1 Automatic Characterization of Concept-Based Signal Features

Highlighting color concepts. After a first step of low-level color extraction in the RGB space for each pixel of a rectangular region, we set up a transformation process for characterizing this information in the HVC space.

Indeed, the use of the RGB color space firsthand is inefficient since the perceptive similarity between color pairs is not taken into account. Consequently, the color information is conveyed in the HVC perceptive space, which is moreover uniform.

The transformation process from the RGB triples to coordinates in the HVC space is adapted from the algorithm described in [11]:

1. The first step consists in transforming the coordinates in the RGB color space into (X, Y, Z) components such that:

$$X = 0.607R + 0.174G + 0.201B; \quad Y = 0.299R + 0.587G + 0.114B; \quad Z = 0.066G + 1.117B$$

2. The second step transforms the (X, Y, Z) in (M₁, M₂, M₃) such that:

$$M_1 = 11.6 [(X/X_0)^{1/3} - (Y/Y_0)^{1/3}]; \quad M_2 = 0.4 \times 11.6 [(Y/Y_0)^{1/3} - (Z/Z_0)^{1/3}]; \\ M_3 = 0.23 \times [11.6 (Y/Y_0)^{1/3} - 1.6]$$

where X₀, Y₀ et Z₀ represent the values of X, Y et Z for the color reference: white.

3. The components in the (H, V, C) space are then determined from (M₁, M₂, M₃)

$$H' = \arctan (M_2 / M_1); \quad S1 = [8.88 + 0.966 \times \cos (H')] \times M_1 ;$$

$$S2 = [8.025 + 2.558 \times \sin (H')] \times M_2$$

$$H = \arctan (S_2/S_1); \quad V = 11.6 (Y/Y_0)^{1/3} - 1.6; \quad C = \sqrt{S_1^2 + S_2^2}$$

Components H, V and C correspond respectively to the values of tonality, luminosity and saturation. They are then mapped to the eleven color concepts introduced in section 2.1.

We iterate this process for all pixels and we finally obtain the pixel percentage corresponding to each color concept for the rectangular region processed. These data constitute a vector structure consisting of eleven dimensions with each dimension representing the pixel percentage for a given color concept.

Characterization of texture concepts. We focus on computational texture extraction at the level of a rectangular regions and characterize it by its Gabor energy distribution within seven spatial frequencies covering the whole spectral domain and seven angular orientations. It is then represented by a 49-dimension vector, with each dimension corresponding to a Gabor energy.

The eleven high-level texture concepts, foundation of our framework for texture symbolic characterization are automatically mapped to the 49-dimension vectors of Gabor energies through support vector machines. We adopt the one-against-rest approach where a separate classifier is designed for each of the eleven texture concepts for reasons of optimized inter-class separation. For each of the eleven texture concepts, the best cross-validation rate is given in table 1. Let us note that the SVMs are able to label new instances of unknown textures with corresponding texture concepts with a high accuracy, cross-validation percentages being all higher than 80%.

Table 1. Cross-Validation Percentages

TW	B	C	D	I	L	M	N	S	S _p	U	W
%	83,7	85,2	88,9	91,9	94,5	98	86,8	83,4	90	97,3	81,4

Generation of symbolic spatial relations. For the automatic characterization of spatial relations, an image agent is characterized by its centre of gravity ia_g as well as two pixel sets: its interior, noted ia_i and its boundary, noted ia_b . To deal with the automatic computation of topological relations, two image agents $ia1$ and $ia2$ are characterized by intersections of their interior and boundary sets: $ia1_i \cap ia2_i$, $ia1_i \cap ia2_b$, $ia1_b \cap ia2_i$ and $ia1_b \cap ia2_b$. Each topological relation is mapped to the results of these intersections, e.g. (DC, $ia1$, $ia2$) iff. $ia1_i \cap ia2_i = \emptyset$, $ia1_i \cap ia2_b = \emptyset$, $ia1_b \cap ia2_i = \emptyset$ and $ia1_b \cap ia2_b = \emptyset$. The interest of this computation method relies on the association of topological relations to the previous set of necessary and sufficient conditions involving attributes of image agents (i.e interior and boundary). The computation of directional relations between $ia1$ and $ia2$ is based on their centers of gravity $ia1_c(x1_c, y1_c)$ and $ia2_c(x2_c, y2_c)$, the minimal and maximal coordinates along the [Ox) axis ($x1_{min}$, $x2_{min}$ and $x1_{max}$, $x2_{max}$) as well as the minimal and maximal coordinates along the [Oy) axis ($y1_{min}$, $y2_{min}$ and $y1_{max}$, $y2_{max}$) of their four extremities. Finally, to distinguish between near and far relations we use the D_{nf} constant given by $D_{nf} = d(\vec{0}, 0.5*[\sigma_1, \sigma_2]^T)$ where d is the Euclidean distance between the null vector $\vec{0}$ and $[\sigma_1, \sigma_2]^T$ is the vector of standard deviations of the localization of centers of gravity for each visual object in each dimension from the overall spatial distribution of all

visual objects in the corpus. D_{nf} is therefore a measure of the spread of the distribution of centers of gravity. This distance agrees with results from psychophysics and can be interpreted as the bigger the spread, the larger the distances between centers of gravity are. We will say that two image agents are **near** if the Euclidean distance between their centers of gravity is inferior to D_{nf} , **far** otherwise.

5.2 Experimental Highlighting of Multimedia Agents

Automatic Extraction. As far as the feature extraction processes are concerned, our algorithm is summarized below:

- Given an image in the index corpus
- We set a rectangular grid over it highlighting the visual regions $\{r_1, \dots, r_i\dots\}$ of size 35x35 pixels, regions overriding by 12 pixels with respect to [Ox] and [Oy).
- For each region, we characterize the color and texture concepts as presented in section 5.1. It is then described by a 22-dimension structure.
 - o After a step of low-level characterization in the RGB space, the color concepts are highlighted in the perceptual HVC space.
 - o For the textures, the Gabor energy matrices are linked to the texture concepts.

Learning Process. The algorithm for the learning process is as follows:

- Given a 'positive' set of annotated training objects in the knowledge base, i.e. corresponding to the semantic concept being learnt:
 - o We set over it a rectangular grid highlighting the regions with size 35x35 pixels.
 - o We extract the color and texture concepts for each region with their associated recognition probabilities.
 - o Features corresponding to the region are used as training input for the learning framework.

Highlighting the multimedia agents. For the recognition step, the algorithmic process is based on five steps:

- Given an image segmented into regions
- For each semantic concept, we use the probabilistic classifier which provides an output value.
- We obtain for each region the probabilities linked to the visual semantic concepts
- We consider as the representative visual semantic concept the one associated to the maximum recognition probability
 - o In case of a « conflict » (i.e. a block such as two distinct visual semantic concepts have the same maximum probability values), the decision will be based on taking into account the visual semantic concepts with the maximum recognition probabilities in adjacent regions.
- We agglomerate the regions with respect to the protocol described in section 3.2 in order to highlight the image agents.

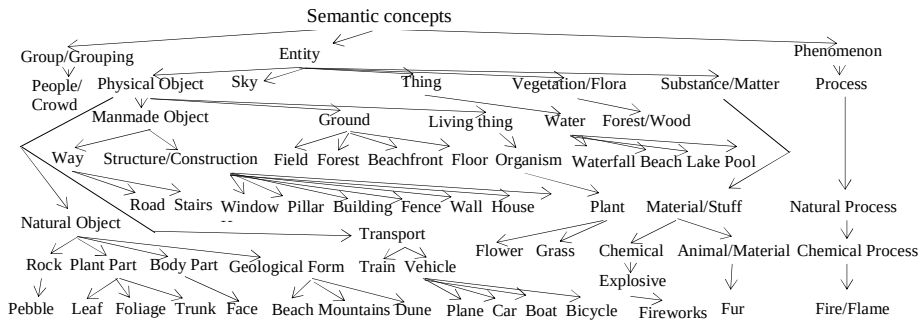


Fig. 3. Lattice organizing semantic concepts

5.3 Validation

Semantic concepts are organized within a visual ontology (provided as a lattice-based structure in fig. 3).

The evaluation is carried out on a corpus of color personal photographs instead of the Corel collection since it has been argued that the latter is much easier to annotate and retrieve; and in fact does not capture the difficulties inherent in real world data-sets (example images are provided in [10]). It is based on the notion of *visual relevance* which consists in quantifying the correspondence between query and index image document. We compare our framework with an image retrieval system based on a semantic approach: the *Visual Keyword* (VK) system S_1 [10] and a state-of-the-art loosely-coupled system S_2 combining a text-based framework for querying on semantics and a relevance feedback process operating on color and texture features to enrich the semantic characterization with additional **signal information** (such as in [18]). We test our framework by proposing semantic concept queries corresponding to all concepts in the lattice of fig. 3. Recall/precision curves of fig. 4 illustrate the average results obtained for these queries. The average precision of SIR_Agent (0.3625) is here higher than the average precisions of the VK system S_1 (0.334) and the loosely-coupled S_2 system (0.305).

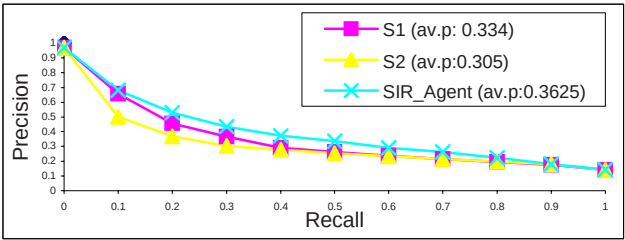


Fig. 4. Recall/Precision Curves for all semantic concept queries

6 Conclusion

We proposed in this paper the specification of the SIR_Agent system which explores the formation of semantic concepts in a population of image agents, abstract structures representing visual entities within an image document. These agents are individually highlighted through a statistical model which considers the joint distribution of signal (color, texture) and semantic concepts. The strength of our approach relies on avoiding the pre-processing of images with computationally-expensive automatic segmentation processes, and on the use of “transparent” processes as well as compact grid-based representations for the semantic characterization. Moreover, contrary to traditional frameworks considering high-dimensional spaces of low-level extracted signal features, this model addresses the curse of dimensionality.

Then, at the image agent population level, the agents communicate about the perceived semantic concepts through mediator agents with no access to global information or to the representations of other agents, they only exchange conceptual information. While doing so they adapt their internal representations to be more successful at conveying the perceived semantic information in future interactions. The image content is therefore soundly inferred through these concept-based linguistic interactions. We have proposed an experimental instantiation of our theoretical framework and evaluated its quality in an image retrieval task. The results obtained, after comparison with two state-of-the-art image retrieval systems and considering a corpus of personal photographs instead of the “easy-to-process” Corel dataset, allowed us to validate our proposal on a set of semantic concept queries.

References

- [1] Berlin, B., Kay, P.: Basic Color Terms: Their universality and Evolution. UC Press (1991)
- [2] Bhushan, N., et al.: The Texture Lexicon: Understanding the Categorization of Visual Texture Terms and Their Relationship to Texture Images. *Cognitive Science* 21(2), 219–246 (1997)
- [3] Blei, D., Jordan, M.: Modeling Annotated Data. *ACM SIGIR*, 127–134 (2003)
- [4] Bradshaw, B.: Semantic based image retrieval: a probabilistic approach. *ACM MM*, 167–176 (2000)
- [5] Egenhofer, M.: Reasoning about binary topological relations. In: Günther, O., Schek, H.-J. (eds.) *SSD 1991*. LNCS, vol. 525, pp. 143–160. Springer, Heidelberg (1991)
- [6] Feng, S., et al.: Multiple Bernoulli Relevance Models for image and video annotation. In: *CVPR*, pp. 1002–1009 (2004)
- [7] Gacogne, L.: Elements of fuzzy logics. Hermes Editions (1997)
- [8] Gong, Y., et al.: Image Indexing and Retrieval Based on Color Histograms. *Multimedia Tools and App. II*, 133–156 (1996)
- [9] Leow, W.K., Lai, S.Y.: Invariant matching of texture for content-based image retrieval. *MMM*, 53–68 (1997)
- [10] Lim, J., Jin, J.S.: A structured learning framework for content-based image indexing and visual query. *Multimedia Systems* 10(4), 317–331 (2005)
- [11] Miyahara, M., Yoshida, Y.: Mathematical Transform of (R,G,B) Color Data to Munsell (H,V,C) Color Data. In: *SPIE*, vol. 1001, pp. 650–657 (1988)

- [12] Platt, J.C.: Probabilities for Support Vector Machines. *Advances in Large Margin Classifiers*, pp. 61–74. MIT Press, Cambridge (1999)
- [13] Mojsilovic, A., Rogowitz, B.: Capturing image semantics with low-level descriptors. In: *ICIP*, pp. 18–21 (2001)
- [14] Smeulders, A., et al.: Content-based image retrieval at the end of the early years. *IEEE PAMI* 22(12), 1349–1380 (2000)
- [15] Srikanth, M., et al.: Exploiting Ontologies for Automatic Image Annotation. *ACM SIGIR*, 1349–1380 (2005)
- [16] Town, C.P., Sinclair, D.: *CBIR Using Semantic Visual Categories*. TR2000-14. AT&T Labs, Cambridge (2000)
- [17] Vapnik, V.: *Statistical Learning Theory*. Wiley, NYC (1998)
- [18] Zhou, X., Huang, T.: Unifying Keywords and Visual Contents in Image Retrieval. *IEEE Multimedia* 9(2), 23–33 (2002)

Supporting Proscriptive Metadata in an XML DBMS

Hao Jin¹ and Curtis Dyreson²

¹ Expedia, Seattle, WA, USA
hjin@expedia.com

² Utah State University, Logan UT, USA
Curtis.Dyreson@usu.edu

Abstract. MetaXQuery is a language for querying data enhanced with metadata. The MetaXQuery data model (MetaDOM) attaches metadata to each element in an XML data collection, and extends XQuery with several constructs to process and query metadata. In this paper we show how to extend a native XML DBMS, namely eXist, to support MetaXQuery. The additional query functionality can be efficiently implemented by judicious reuse of eXist's indexes and query evaluation engine.

Keywords: XML, metadata, XQuery, indexing.

1 Introduction

Metadata is an integral component of many data collections. It plays a key role in *describing*, *interpreting*, and *proscribing* data. Specific kinds of metadata, in particular, knowledge classification metadata and temporal metadata have heretofore been studied in great detail. But data models and query languages that combine the disparate varieties of metadata, such as security, reliability, cost, time, locale, privacy, language, and content descriptors, are rare. Each kind of metadata often has a unique semantics, especially for query evaluation. For instance, security metadata is active in restricting access to data whereas content descriptors are inert; they simply provide additional information.

This paper outlines the implementation of a system to support the manifold varieties of metadata in a single, XML-based framework. XML is an important format for data exchange and representation on the web. In our approach both data and metadata are described in XML (and perhaps related using RDF [22]). The data and metadata are combined in a unified data model (a MetaDOM [8]) that can be queried with MetaXQuery [14],[15].

This paper is organized as follows. First we briefly review MetaXQuery and describe the implementation challenges. Next we present an algebra for evaluating a MetaXQuery query. Finally, we outline extensions to a native XML DBMS to support MetaXQuery and report on experiments to measure the cost of processing metadata. More on the project, including all code and an on-line demo, is available from the project's home page.¹

¹ <http://www.cs.usu.edu/~cdyreson/pub/MetaXQuery>

1.1 Motivation

Suppose that a bank uses an XML document, `customers.xml`, to store the account and balance information of its customers. Figure 1(a) shows a part of the XML document, and Figure 1(b) is the data model for this document fragment.

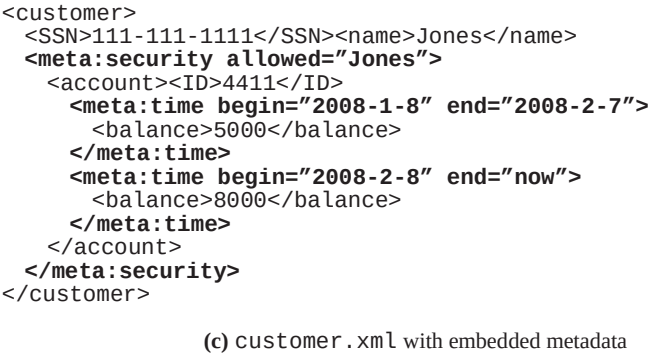
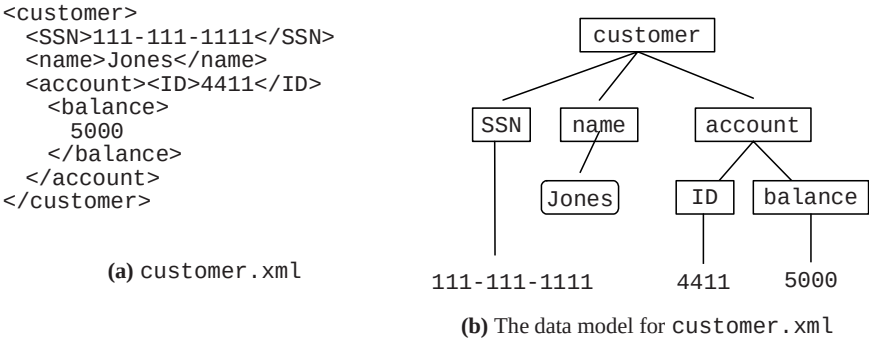


Fig. 1. A motivating example

This is a simplified data model for demonstration purposes since white space has been removed. The document in Figure 1(a) is incomplete because the balance of the account changes over time, and we'd like to record how it changes, in addition to the current balance. Moreover, when the data is provided to a user, we certainly don't want everyone to be able to see the entire document. In particular, we want only authorized users to see the account data.

Figure 1(c) gives an example of how time and security metadata might be embedded in the XML data. **The lines in bold denote metadata.** The account balance changes over time, and we also want to add security constraints to tell the application which users can access which part of the data. The **<meta:security>** element restricts access to the customer Jones. The **<meta:time>** element that wraps the balance element tells us that this account has a balance of 5000 from 2008-1-8 to 2008-2-7, and then a balance of 8000 from 2008-2-8 to now. The "meta:" namespace is used to specify that this information is metadata rather than data. This

example is very simple; in general, the metadata can be complicated and involve complex data types.

Every query uses the metadata to constrain the data. Consider a query to find the current account balance, e.g., “//account“. Implicit in the evaluation of this query is a user’s *perspective*. The perspective is a combination of metadata that represents the *context* in which the query is posed. For instance, suppose that the user’s perspective is a security of “Jones” (set when the user “logged into” the system) and the time “now” (a default setting). A query filters the data by matching the user’s perspective to the metadata. The data that is visible from that perspective is shown in Figure 1(a), i.e., the current snapshot of the data. Effectively, the evaluation of a query computes a view of the data that “conforms to” the metadata in the user’s perspective.

Finally, let’s consider the role of meta-metadata. Suppose that Jones closes his account. The security on Jones’s account should evolve to denote that Jones *used to have* access (prior to now), but from now on he *no longer* has access. The lifetime of each security restriction is meta-metadata, that is, it is metadata for the security metadata. The reason that it is important to model the evolving security is that Jones should be denied access to his “no-longer-current” account (e.g., ATM cash withdrawals) because the account is closed, but he should still have access to old account information (e.g., for tax purposes). When Jones changes his query to “roll back” the database to 2008-2-6, he should see his account with a balance of 5000. It is rare that an information system supports meta-metadata (for instance relational DBMSs do not support versioned security) but if it did Jones could still access the archived information on his account after his account has closed.

1.2 Review of MetaDOM

MetaDOM [8] extends the Document Object Model (DOM) by adding an optional `meta` property to a node’s Information Set. But in all other respects, the data model is the same as DOM. The value of the `meta` property is a reference to the root node of a

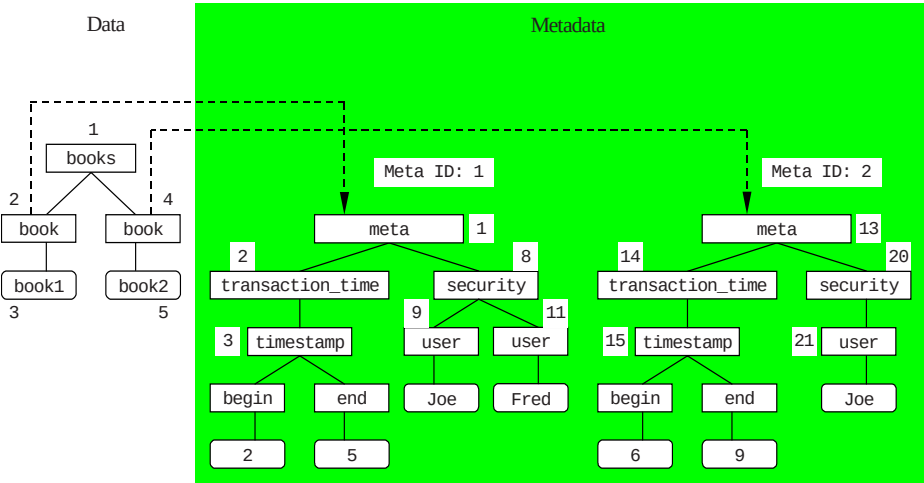


Fig. 2. An example MetaDOM

nested MetaDOM, which describes the metadata for a node. The metadata can be recursively nested, that is, a node in a MetaDOM may itself have a *meta* property. Each level of nesting adds another level of metadata.

Figure 2 shows an example MetaDOM. The two book nodes in the data are annotated with metadata concerning security restrictions and transaction time information. The security metadata proscribes access to the node (and its descendents) to just Joe or Fred. The transaction time metadata records when the node is stored in the database, e.g., it was present between times 2 and 5. Note that each piece of metadata is itself a MetaDOM, which allows meta-metadata to be added to nodes in the metadata.

1.3 Review of MetaXQuery

There are several kinds of queries that can be evaluated on a metadata-enhanced data model. Elsewhere we described queries that involve grouping [14] and cleaning the metadata [15], but here we focus on queries that access the data, and are constrained by the metadata.

All queries implicitly utilize the metadata. Implicit in each query is the *perspective* of the user that issues the query. The perspective covers the user's security rights, privacy settings, and locale, among other metadata settings. Whenever a query is evaluated the metadata must be consulted to ensure that the user's perspective matches the metadata. Usually a mismatch will crop the user's view of the data, e.g., a user without the proper security certificate cannot access secure data.

As an example consider the following query to find books from the perspective of user Fred as of time 4. The perspective is explicitly given at the start of the query in this example, usually the perspective will be implicitly acquired via a login process to the DBMS and applied to all queries in that session.

```
PERSPECTIVE
  <security>Fred</security>
  <transaction_time>4</transaction_time>
FOR $b IN //book
RETURN <booksAvailableToFred>
      {$b}
      </booksAvailableToFred>
```

Enforcing perspective is very important for the system because each user of the system should be allowed access to only the data that matches their perspective. Since materializing the view of the data for each perspective would be prohibitively expensive, the perspective is dynamically matched during a query by the **filterByPerspective** function that is added to each *path expression* in the query. The function is defined below (the style of the definition is from the XQuery spec. [23]).

Definition [filterByPerspective]. The **filterByPerspective** function takes a sequence of data nodes and the root node of a perspective MetaDOM. It filters the data sequence, keeping only those nodes that match the perspective.

```
meta-fn:filterByPerspective($node as meta-dm:node()*,
                             $pNode as meta-dm:node())
as meta-dm:node()*
```



In the query given above, the path expression in the FOR clause would be modified to the following:

```
FOR $b IN meta-fn:filterByPerspective(//book, P)
```

where P is the root of the perspective MetaDOM, but the rest of the query remains the same. For a sequence of queries, a user can set or modify the implicit perspective using the `pragma` keyword. Pragmas are typically used to furnish hints for query evaluation. MetaXQuery adds a `perspective` pragma, which identifies a perspective document. An example of loading an XML file as the perspective is given below.

```
(::pragma perspective perspective.xml ::)
```

1.4 Implementation Challenges

The overarching goal of implementing MetaXQuery is reusing existing standards and technology. For instance, MetaXQuery is upwards-compatible with XQuery, as is MetaDOM with DOM. Ideally, few changes will have to be made to a native XML DBMS (XDBMS) to implement MetaXQuery. But there are two key implementation challenges. First, MetaXQuery introduces *data scopes* into the data model. In the data model, metadata must be (logically) separate from the data so that wildcard queries (e.g., a descendent axis) explore only within the intended scope. Only the **meta** axis can bridge scopes. Unfortunately, XDBMSs do not support separate scopes for data. The second important challenge is supporting the **filterByPerspective** function. The function applies additional constraints to nodes identified by *every* path expression in a query. There is one check that must be performed for each kind of metadata. Additional levels of metadata add even more constraints. So efficient implementation of **filterByPerspective** is critically important to building support for (proscriptive) metadata into an XDBMS.

2 The Metadata Tree Algebra

In this section we describe how to extend an algebra for XML to support MetaXQuery. We chose to extend the Tree Algebra for XML (TAX) [12]. We first review TAX and then describe how to extend it.

2.1 TAX

TAX provides low-level operations for the evaluation of XQuery queries. XQuery queries can be translated to TAX expressions for fast evaluation. Typical operators in TAX (selection, projection, etc.) take a collection of data trees as input, a *pattern tree* and an *adornment* as parameters, and produce a collection of data trees as output. A pattern tree is a simple, intuitive specification of how to locate nodes of interest. Each node in a pattern tree represents a variable that is bound to some nodes in the data model (e.g., a DOM node). Each edge represents a relationship between a pair of

bound variables. A TAX pattern tree has two types of edges, parent-child (**pc**) and ancestor-descendant (**ad**). A **pc** edge is used for a parent or child axis in a path expression while an **ad** edge represents an ancestor or descendent axis. Additionally, a pattern tree has an adornment which is a Boolean formula of predicates. **Fig. 3** shows a simple pattern tree for the path expression “/books/book”. Variables \$1 and \$2 are related by a parent-child edge meaning that \$1 must be a parent of \$2 in the data model. When both variables are bound to a node, the associated adornment can be evaluated. The adornment tests to ensure that the name attribute of \$1 is “books” and the name attribute of \$2 is “book”.

2.2 MetaTAX

In this paper we introduce MetaTAX, which is an extension of TAX to support metadata.

2.2.1 Meta Edges

MetaTAX introduces **meta** edges to the pattern tree. The **meta** edge is inserted into the pattern tree whenever a meta axis is used in a MetaXQuery path expression. Figure 4 gives an example of a pattern tree for the path expression “/books/book/meta::security.”

2.2.2 Perspective Filtering

In addition to the meta axis MetaXQuery has a **filterByPerspective** function that is invoked in most queries. The **filterByPerspective** function combines a function to project metadata, called **getMetadataValues**, with a function to select data using metadata, called **filterByMetadataValues**. In the rest of this section we show how to translate each of these functions into MetaTAX operators or plans.

The **getMetadataValues** function retrieves a specified type of metadata value for the input data nodes. In MetaTAX the function call translates to a simple pattern tree with a **meta** edge and a selection list (SL) on the metadata type node. For example, the function **getMetadataValues**("security", /books/book) would translate into a pattern tree shown in Figure 5. The pattern tree specifies which nodes are of interest in this query (books, book, meta, and security), and the Selection List (SL) acts as the adornment parameter. It lists the nodes (and their descendants) that are output from the evaluation of the pattern tree.

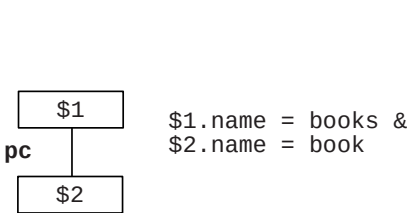


Fig. 3. A pattern tree for /books/book

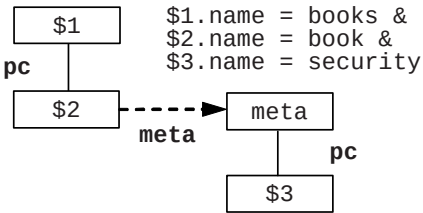


Fig. 4. A MetaTAX pattern tree that explores the meta axis

The **filterByMetadataValues** operation selects data nodes that satisfy a given metadata condition. In MetaTAX, the operation is modeled as a *projection* of the data nodes with certain metadata conditions. Projection in TAX is different from projection in the relational algebra. In relational algebra, selection and projection are orthogonal operations: selection chooses rows and projection chooses columns. But in a tree data model, there are no such obvious orthogonal dimensions like rows and columns, so the role of projection is quite similar to selection. Projection selects only certain nodes, eliminating others. A sample pattern tree for the function **filterByMetadataValues**("security", /books/book, "Joe") is shown in Figure 6. The pattern tree specifies which part of the input data is of interest. The projection list (PL) tells the query processor which nodes to preserve in the output. In projection, no matter whether the nodes are in the projection list or not, all of their contents are preserved from the input.

The **filterByPerspective** function is a combination of the **getMetadataValues** and **filterByMetadataValues** functions. Theoretically it iterates through each metadata type element in the perspective, getting the value of the element. For each kind of metadata it calls the **filterByMetadataValues** function to determine whether the data node matches the metadata in the perspective. Those that match are kept in the node list.

For persistent data collections a better strategy is to use indexes on the metadata to quickly find which metadata matches a given perspective. For each kind of metadata, the index lookup will return a list of metadata trees that match the perspective. The lists for each kind of metadata are then joined together to produce a final list of metadata that matches the perspective, and that list is in turn joined with the data to produce a result. Assuming there are N kinds of metadata, this strategy will require N index lookups and N joins ($N-1$ joins of the different kinds of metadata and one join with the data). The join order can be rearranged to improve efficiency. Most of the time there will be far more data than metadata, so the join with the data should be delayed as long as possible. The metadata candidate lists should be joined first to find out which combinations of metadata match the perspective. The resulting combinations are then joined with the data.

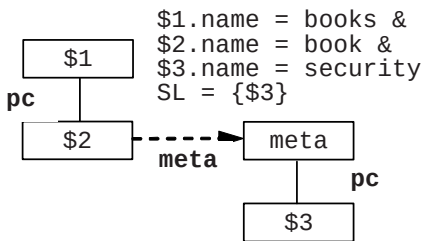


Fig. 5. Sample Pattern Tree for the getMetadataValues Function

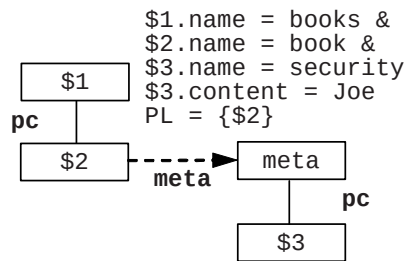


Fig. 6. Sample Pattern Tree for the filterByPerspective Function

3 Plan for Extending a Native XML Database

The algebra outlined in Section 2 to support metadata is relatively straightforward to implement in a native XML DBMS (XDBMS). XDBMSs store XML documents in a

back-end database or flat file. Usually, one or more indexes are created to improve query evaluation efficiency. Storing metadata is straightforward since the metadata is also an XML document. Each chunk of metadata is identified by a “metaID” attribute in the metadata. An element in the data subsequently references a chunk of metadata with a “metaRef” attribute. Typically, many elements will share the same metadata or have no metadata, so the metadata will be much smaller in size than the data.

Many XDBMS query evaluation engines use a path index to efficiently evaluate a query. A path index locates nodes for a given path expression, which saves on the (prohibitive) cost of traversing the data model to find the nodes. The result of the path index lookup is then combined with other index lookups (e.g., a text or word index) to process additional search conditions in a query, but in this section we will focus on a path index to illustrate how we plan to incorporate metadata in query evaluation.

Let’s use the data model in Figure 2 as an example. In the data part of Figure 2, the path index would map “/books/book” to the list of nodes with ID 2 and 4, as shown in Table 1. We extend the path index with an additional column to record the Meta Ref value for a list of nodes, as shown in Table 2. Figure 2 shows that the metadata chunk with a Meta ID of 1 is associated with the data node with ID.

Table 1. Original XML Path Index

Data Path	Node List
/books/book	(2, 4)

Table 2. New XML Path Index

Data Path	Node List	Meta Ref
/books/book	(2)	1
/books/book	(4)	2

A perspective includes constraints on the metadata. For instance, a user might query from a transaction time perspective of 3 (i.e., rollback the database to time 3) and a security of Joe. Additional indexes are constructed to efficiently search for chunks of metadata that satisfy a specific constraint. Table 3 shows an index for transaction time metadata, while Table 4 illustrates one for security metadata. The Node ID column identifies the source of the metadata in the metadata document. Since meta-metadata could be present, each row in the index includes a Meta-meta Ref column. The data model in Figure 2 has no meta-metadata so that column contains NULLs.

Now let’s demonstrate the use of the indexes with an example. Suppose we have the following query: “Find book data that is available to the user Joe and exists in the database at time 3.” The steps in the query execution plan for this query are shown in Figure 7. The transaction time index is used to find intervals that include time 3. Similarly the security index is used to find metadata chunks for the user Joe (Figure 7(a)). The path index is then used to locate nodes that match the path expression “//book” (Figure 7(b)). The results of the index lookups are joined on the Meta ID column (the join order is determined during query optimization) generating a result (Figure 7(c)).

Table 3. Level 1 Index on Transaction Time

Time	Meta ID	Node ID	Meta-meta Ref
[2, 5]	1	3	NULL
[6, 9]	2	15	NULL

Table 4. Level 1 Index on Security

Security	Meta ID	Node ID	Meta-meta Ref
Joe	1	9	NULL
Fred	1	11	NULL
Joe	2	21	NULL

4 Implementation in eXist

eXist [16] is an open-source, native XML DBMS. We chose to modify eXist both because the source is available, and also eXist outperformed other systems in our benchmark system.

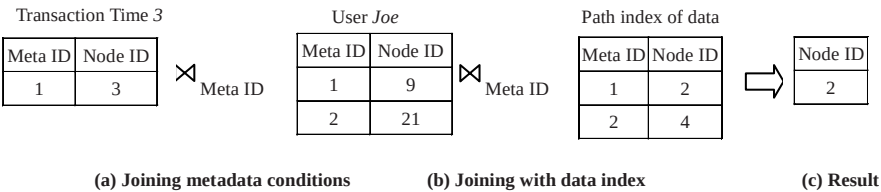


Fig. 7. Use of indexes to solve the example query

4.1 Storage

eXist stores an XML document by adding it to a *collection*, which can hold a set of XML documents. A document is serialized and stored in a paged, data file when it is added to the collection. We use a “MetaDocRef” processing instruction in the data to identify its metadata. When the data is parsed, the metadata is also parsed and added to a separate metadata collection. Information about the metadata is placed into the *metadata indexes*. As discussed in the previous section there is an index for each kind of metadata, e.g., a security index. The index is built when a metadata is parsed and stored. We used eXist’s B⁺-tree index classes for each metadata index rather than using a specialized index, e.g., a temporal index.

4.2 Meta Axis

The implementation of meta axis is straightforward. We had to modify the XQuery parser to recognize the meta axis. The XQuery parser in eXist is created by ANTLR (ANother Tool for Language Recognition), so it is easy to modify. The parsing rules in eXist closely follow the EBNF defined in the W3C XQuery standard. We extended the *axisStep* rule to accept a meta axis step. We also added a class to evaluate the step, and added to the abstract syntax tree so that the new class would be invoked when the meta axis was used.

4.3 Perspective

In order to help the users access and query the metadata, we extended eXist with the **filterByPerspective** function. Essentially the function implements the query evaluation plan discussed in Section 3.

5 Experiments

We chose the XMark benchmark [20] to test our MetaXQuery implementation. We designed experiments to evaluate the scalability of our system and also to compare the performance of different query strategies mentioned above. We performed the experiments on a single-processor 1.7GHz Pentium IV machine with 1GB of memory running Windows XP Professional and Sun JDK version 1.4.2. We chose a subset of the XMark queries that were both supported by eXist (e.g., Q3 was not supported) and within a certain performance bound. For the original XMark queries, please refer to [20]. Table 5 shows the parameters for the data generation of each experiment. The Factor column is the XMark factor used to generate the data. Different factors generate documents of different sizes as shown in the Document Size column. The experments also varied the kinds of metadata (# of Metadata Properties) and how varied the metadata is (# Metadata Trees).

Table 5. Experiment Parameters

Experiment	Factor	Document Size (MB)	# Metadata Properties	# Metadata Trees
1	0.01	1.1	0 to 3	# of elements
2	0.01, 0.02, 0.03, 0.04 0.05	1.1, 2.3, 3.5, 4.8, 5.8	2	1

Experiment 1 tests the scaling factor of the MetaXQuery processor with increasing types of metadata. The experiment fixes the size of the data and metadata, and increases the types of metadata from 0 to 3. The queries used in this experiment are the optimized and the metadata fragment is the same as the total number of data elements. All of these fragments have different values so that the effect of the index is maximized. Figure 8 shows the performance of selected queries in experiment 2. We only choose some of the queries here to make the graph clearer. All of the queries show a sub-linear increase in time with the increasing number of metadata types, which indicates that the system will scale as the number of metadata types increases.

Experiment 2 compares two query execution strategies of the **filterByPerspective** function. As mentioned in Section 2.2, there could be a naïve implementation of **filterByPerspective** that iterates through every data node and checks if its metadata matches the perspective. We suggested that a much better strategy would be to use the index to find all the metadata fragments that match the perspective first and then join

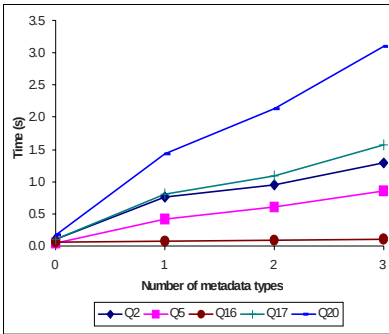


Fig. 8. Performance data of experiment 2

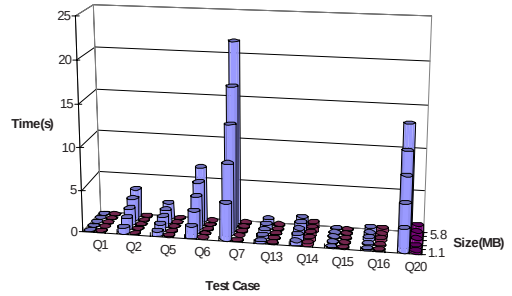


Fig. 9. Performance data comparing the traversal method with the index join method

them with the data nodes. In this experiment we compare these two query evaluation strategies. We call the first strategy the *traversal method* and the second the *index join method*. We tested a series of data documents from factor 0.01 to 0.05. For each test we used the same metadata value for all the elements. For the traversal method, it is pretty much the same as using different metadata fragments or values because it checks the metadata value of every data node. For the index join algorithm on the other hand, the index doesn't help much so it's mostly the cost of joins. This is actually reasonable because with the help of data structures like B^+ -trees, index lookup is always a very fast operation. So the dominant part of the query execution is still going to be the joins. We use the same experimental setup as the previous experiments so that the results are the same as the original data. Figure 9 shows the performance of the experiment. As we expected, the index join method outperforms the traversal method in most of the test cases. And furthermore, it shows much better scalability. The cost of the traversal method increases linearly with the data size, but the indexes and joins (a merge-join) show sub-linear increase. The results justify our work in developing the more efficient metadata association join algorithm in filtering data with metadata conditions.

6 Related Work

There are few papers on systems to manage metadata in native XML databases. The most widely used language on the web for annotating a document with metadata is the RDF [22]. Several strategies for unifying the representation of XML and RDF have been proposed [11], but query languages have largely targeted either RDF or XML. There have been several RDF query languages proposed in the literature. For a comparison of these RDF query languages, please refer to [10].

There are several systems that support metadata similar to MetaXQuery. Mihaila et al. suggest annotating data with quality and reliability metadata and discuss how to query the data and metadata in combination [18]. The SPARCE system wraps or

superimposes a data model with a layer of metadata [19]. The metadata is active during queries to direct and constrain the search for desired information. Stavrakas et al. embed descriptive metadata in a semistructured data model to query information in different contexts [21]. A similar approach is taken by Jagadish et al. in researching Colorful XML [13]. But proscriptive metadata, the focus of this paper (and our earlier work [7],[8]), is not considered in either approach. Systems that provide mappings between metadata models at the schema-level are also popular [17]. MetaXQuery differs from these systems by focusing on XQuery extensions to support metadata, and by building a framework whereby the semantics of individual kinds of metadata can be specified as “plug-in” components.

Support for particular kinds of metadata has been researched. Two of the most important and most widely discussed types of (proscriptive) metadata are temporal metadata and security metadata. Temporal extensions of almost every W3C recommendation exist, for instance, τ XQuery [9] and τ XSchema [6]. There has also been research on security in XML management systems, e.g., [4]. Our approach is to build an infrastructure that supports a wide range of different kinds of metadata in the same vein as our previous efforts with the semistructured data model [7] and XPath data model [8].

It is common to transform queries into efficient low-level, algebraic operators. The XML algebras include the Tree Algebra for XML (TAX) [12], the XML Query Algebra [3], and an algebra on a graph structure [2]. Research on query execution plans and especially join algorithms include containment queries [24], structural joins [1], and twig joins [5]. We extended the TAX algebra to support our metadata functionalities and also borrowed ideas from some of the join algorithms for our metadata association join.

7 Conclusion

Support for metadata is lacking from most native XML DBMSs (XDBMSs). In this paper we describe how to efficiently store and query metadata in an XDBMS. We focused on the challenge of efficiently matching a metadata perspective during query evaluation. Efficient matching is essential to ensuring that proscriptive metadata is quickly and correctly handled. This paper makes three primary contributions. First, it extends the Tree Algebra for XML (TAX) with constructs to support metadata. Meta edges were added to pattern trees to provide access to the metadata. We also showed how to model various metadata-related functions using the extended pattern trees. The second contribution is the implementation of MetaDOM and MetaXQuery in an open source XDBMS. The implementation stores metadata in a separate metadata collection. The collection is consulted during evaluation of a MetaXQuery query, especially as the query’s perspective is matched to the metadata. The matching process uses indexes and joins already available in an XDBMS, so it can be implemented with few changes to XDBMS’s architecture. Finally, we experimentally showed that the cost of evaluated MetaXQuery queries is modest, especially when those queries can be optimized.

References

- [1] Al-Khalifa, S., Jagadish, H.V.: Multi-level operator combination in XML query processing. In: CIKM, pp. 134–141. McLean, Virginia (November 2002)
- [2] Al-Khalifa, S., Jagadish, H.V., Patel, J.M., Wu, Y., Koudas, N., Srivastava, D.: Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In: ICDE, San Jose, California, pp. 141–152 (February–March 2002)
- [3] Beech, D., Malhotra, A., Rys, M.: A Formal Data Model and Algebra for XML. W3C XML Query working group note (September 1999)
- [4] Bertino, E., Castano, S., Ferrari, E., Mesiti, M.: Specifying and Enforcing Access Control Policies for XML Document Sources. *WWW Journal* 3(3), 139–151 (2000)
- [5] Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: optimal XML pattern matching. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 2002, pp. 310–321 (2002)
- [6] Currim, F., Currim, S., Dyreson, C., Snodgrass, R.T.: A Tale of Two Schemas: Creating a Temporal XML Schema from a Snapshot Schema with τ XSchema. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) *EDBT 2004*. LNCS, vol. 2992, pp. 348–365. Springer, Heidelberg (2004)
- [7] Dyreson, C., Böhlen, M., Jensen, C.: Capturing and Querying Multiple Aspects of Semistructured Data. In: *VLDB*, Edinburgh, Scotland, pp. 290–301 (September 1999)
- [8] Dyreson, C., Böhlen, M., Jensen, C.: “METAXPath”. In: Proceedings of the Inter. Conf. on Dublin Core and Metadata Applications, Tokyo, Japan, pp. 17–23 (2001)
- [9] Gao, D., Snodgrass, R.T.: Temporal Slicing in the Evaluation of XML Queries. In: *VLDB*, Berlin, Germany, September 2003, pp. 632–643 (2003)
- [10] Haase, P., Broekstra, J., Eberhart, A., Volz, R.: A Comparison of RDF Query Languages, <http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query>
- [11] Hunter, J., Lagoze, C.: Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles. In: *WWW*, Hong Kong, pp. 457–466 (May 2001)
- [12] Jagadish, H.V., Lakshmanan, L.V.S., Srivastava, D., Thompson, K.: TAX: A Tree Algebra for XML. In: Ghelli, G., Grahne, G. (eds.) *DBPL 2001*. LNCS, vol. 2397, pp. 149–164. Springer, Heidelberg (2002)
- [13] Jagadish, H.V., Lakshmanan, L.V.S., Scannapieco, M., Srivasta, D., Wiwatwattana, N.: Colorful XML: one hierarchy isn’t enough. In: *SIGMOD*, Paris, France, pp. 251–262 (2004)
- [14] Jin, H., Dyreson, C.E.: Grouping in MetaXQuery. In: Proceedings of WISE Conference, Brisbane, Australia, pp. 688–693 (2004)
- [15] Jin, H., Dyreson, C.E.: Sanitizing using Metadata in MetaXQuery. In: *ACM SAC* (2005)
- [16] Meier, W.: eXist: An Open Source Native XML Database, <http://exist.sourceforge.net>
- [17] Melnik, S., Rahm, E., Bernstein, P.A.: Rondo: A Programming Platform for Generic Model Management. In: *SIGMOD*, San Diego, California, June 2003, pp. 193–204 (2003)
- [18] Mihaila, G.A., Raschid, L., Vidal, M.-E.: Using Quality of Data Metadata for Source Selection and Ranking. In: Suciu, D., Vossen, G. (eds.) *WebDB 2000*. LNCS, vol. 1997, pp. 93–98. Springer, Heidelberg (2001)
- [19] Murthy, S., Maier, D., Delcambre, L.M.L., Bowers, S.: Super-imposed Applications using SPARCE. In: *ICDE*, Boston, MA, p. 861 (March 2004)
- [20] Schmidt, A., Waas, F., Kersten, M.L., Carey, M.J., Manolescu, I., Busse, R.: XMark: A Benchmark for XML Data Management. In: Proceedings of VLDB Conference, Hong Kong, China, pp. 974–985 (2002)

- [21] Stavrakas, Y., Pristouris, K., Efandis, A., Sellis, T.: Implementing a Query Language for Context-Dependent Semistructured Data. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 173–188. Springer, Heidelberg (2004)
- [22] World Wide Web Consortium. RDF Primer, W3C Recommendation (February 2004), <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>
- [23] World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Data Model, W3C Working Draft (October 2004), <http://www.w3.org/TR/2004/WD-xpath-datamodel-20041029/>
- [24] Zhang, C., Naughton, J.F., DeWitt, D.J., Luo, Q., Lohman, G.M.: On Supporting Containment Queries in Relational Database Management Systems. In: SIGMOD, Santa Barbara, California (2001)

XPath Rewriting Using Multiple Views

Junhu Wang¹ and Jeffrey Xu Yu²

¹ Griffith University, Gold Coast, Australia

J.Wang@griffith.edu.au

² Chinese University of Hong Kong, Hong Kong, China

yu@se.cuhk.edu.hk

Abstract. We study the problem of tree pattern query rewriting using multiple views for the class of tree patterns in $P^{\{//, \square\}}$. Previous work has considered the rewriting problem using a single view. We consider two different ways of combining multiple views, define rewritings of a tree pattern using these combinations, and study the relationship between them. We show that when rewritings using single views do not exist, we may use such combinations of multiple views to rewrite a query, and even if rewritings using single views do exist, the rewritings using combinations of multiple views may provide more answers than those provided by the union of the rewritings using the individual views. We also study properties of intersections of tree patterns, and present algorithms for finding rewritings using intersections of views.

1 Introduction

Query rewriting using views has many applications including data integration, query optimization, query caching, and support of physical data independence [5]. Given a query, it studies finding another query using only the views to produce correct answers to the original query. In the literature, two types of query rewritings are studied, namely, equivalent rewritings and contained rewritings. Given a view, V , and a query, Q , an equivalent rewriting produces all answers to the original query Q using view V , whereas a contained rewriting may produce a subset of the answers to Q using V . Both types of rewritings have been extensively studied in the relational database context [5].

Recently, XML query rewriting has attracted attention because of the rising importance of XML data [2, 3, 6, 7, 9, 14]. XPath lies in the center of all XML languages, where the major classes of XPath expressions that have been studied are tree patterns [1, 8]. Among previous work on rewriting XPath queries using views, Xu and Ozsoyoglu [14] studied the complexity of finding equivalent rewritings for four types of tree patterns studied in [8] and presented an approach for finding and minimizing such rewritings. Mandhani and Suciu [7] presented, in addition to cache organization, an efficient but incomplete method for finding equivalent rewritings of tree patterns involving $/$, $//$, $\square$ and $*$ when the patterns are assumed to be minimized and may have value-based predicates. In [6], Lakshmanan et al. studied maximal contained rewritings of tree patterns where

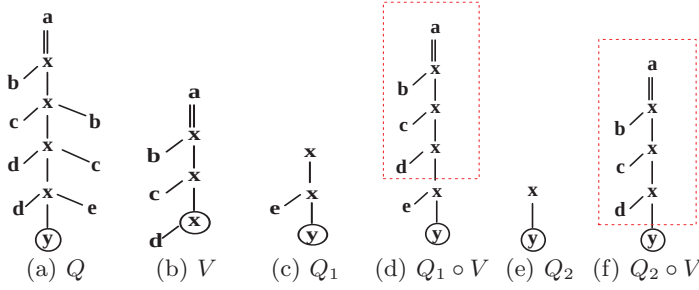


Fig. 2. Q has no equivalent rewriting using V according to conventional definition, but Q can be fully answered using V : $Q_1 \circ V \cap Q_2 \circ V = Q$

It can be verified that

$V_1 \cap V_2 = \text{paper//subsection//example/reference},$

and evaluating the pattern **reference//paper** over the answers of $V_1 \cap V_2$ will produce the same answers as Q , that is, **reference//paper** is an equivalent rewriting of Q using $V_1 \cap V_2$. However, the maximal contained rewriting of Q using V_1 is

reference//example/reference//paper,

and the maximal contained rewriting of Q using V_2 is

reference//subsection//example/reference//paper.

As we show next, the rewriting using $V_1 \cap V_2$ produces strictly more answers than the union of the answers produced by the maximal contained rewritings using V_1 and V_2 individually. For the XML tree t shown in Fig.1, evaluating V_1 over t produces (the subtrees rooted at) n_1 and n_2 , and evaluating V_2 over t produces (the subtrees rooted at) n_2 and n_3 . Therefore, the maximal contained rewriting using V_1 or V_2 will produce no answers for Q , but the rewriting using $V_1 \cap V_2$ will produce the **paper** node under n_2 .

The next example demonstrates observation 3.

Example 2. Consider the query Q and view V shown in Fig.2 (a) and (b) respectively. It can be verified that Q has no equivalent rewritings using V according to the conventional definition. But given any XML tree t , we can find $Q(t)$ using the view as follows. We evaluate the query $Q_1 = x/x[e]/y$ over the subtrees in $V(t)$, and denote the results as $Q_1(V(t))$; we then evaluate $Q_2 = x/y$ over the subtrees in $V(t)$, and obtaining a set denoted as $Q_2(V(t))$. Finally, we take the intersection of $Q_1(V(t))$ and $Q_2(V(t))$. It can be verified that $Q(t) = Q_1(V(t)) \cap Q_2(V(t))$.

We will study contained and equivalent rewritings of tree patterns in $P\{/,[]\}$ using multiple views in the absence and presence of DTDs, with the intersection and union operations. Our main contributions are summarized below.

- We define (contained and equivalent) rewritings of a tree pattern using two different combinations of views, and show the relationship between these rewritings.
- We show the intersection of some tree patterns, if not empty, can be expressed as the union of tree patterns, and provide an efficient algorithm to translate the intersection into union.
- Based on the above, we provide algorithms for finding the maximal contained rewritings and equivalent rewritings using the intersection of views. We also show the effect of an non-recursive DTD on the rewritings.

The rest of the paper is organized as follows. Section 2 provides the background and notations. Section 3 presents the algorithm for reformulating intersections of tree patterns into union, and studies rewritings using the intersection of views. Section 4 defines rewritings using a different combination views, and compare them with rewritings using the intersection. Section 5 considers the effects of non-recursive DTDs. Section 6 surveys related work. Finally Section 7 concludes the paper.

2 Preliminaries

2.1 XML Trees and Tree Patterns

Let Σ be an infinite set of tags. An XML tree is a tree such that every node is labeled with a tag in Σ . A *tree pattern* (TP) is a tree with a unique *distinguished node*, and with every node labeled with a tag in Σ , and every edge labeled with either / or //. Such a TP corresponds to an XPath expression involving the child axis, descendant axis, and branching condition []. A tree pattern has a tree representation. Fig.2 show several TPs, and the TP in Fig.2 (b) represents the XPath expression $a//x[b]/x[c]/x[d]$. Here, single and double lines represent /-edges and //-edges respectively. The distinguished node is indicated by a circle. Note: the TPs in our discussion correspond to the fragment $P^{\{/, []\}}$ defined in [8]. A subset of $P^{\{/, []\}}$, denoted $P^{\{//\}}$, contains all TPs that has a single root-to-leaf path.

Let $\mathcal{T}$ be an XML tree or a TP, and v be a node in $\mathcal{T}$. We will use $N(\mathcal{T})$ to denote the set of all nodes in $\mathcal{T}$, $rt(\mathcal{T})$ to denote the root of $\mathcal{T}$, and $label(v)$ to denote the label of v . For any TP P , DN_P and DP_P denote, respectively, the distinguished node and *distinguished path* of P (i.e., the path from $rt(P)$ to DN_P).

A *matching* of a TP, P , in an XML tree, t , is a mapping δ from $N(P)$ to $N(t)$ satisfying the following conditions: (1) root-preserving, i.e., $\delta(rt(P)) = rt(t)$, (2) label-preserving, i.e., $\forall v \in N(P), label(v) = label(\delta(v))$, and (3) structure-preserving, i.e., for every edge (x, y) in P , if it is a /-edge, then $\delta(y)$ is a child of $\delta(x)$; if it is a //-edge, then $\delta(y)$ is a descendant of $\delta(x)$, i.e, there is a path from $\delta(x)$ to $\delta(y)$. Each matching δ produces a subtree of t rooted at $\delta(DN_P)$, denoted $sub_{\delta(DN_P)}^t$, which is called an *answer* to the TP. We use $P(t)$ to denote the *answer set* of P on t :

$$P(t) = \{sub_{\delta(DN_P)}^t \mid \delta \text{ is a matching of } P \text{ in } t\}.$$

Let T be a set of XML trees. We use $P(T)$ to denote the union of answer sets of Q on the trees in T . That is, $P(T) = \bigcup_{t \in T} P(t)$.

2.2 Intersection and Extension of TPs

Two tree patterns, P and Q , are said to be *comparable* if $\text{label}(\text{rt}(P)) = \text{label}(\text{rt}(Q))$ and $\text{label}(\text{DN}_P) = \text{label}(\text{DN}_Q)$. Let $P_1, \dots, P_n$ be comparable TPs. For any XML tree t , the *intersection* of $P_1, \dots, P_n$, denoted $P_1 \cap \dots \cap P_n$, returns $P_1(t) \cap \dots \cap P_n(t)$. The union of $P_1, \dots, P_n$, denoted $P_1 \cup \dots \cup P_n$, returns $P_1(t) \cup \dots \cup P_n(t)$.

Let P and Q be TPs such that $\text{label}(\text{DN}_P) = \text{label}(\text{rt}(Q))$. The *extension* of P using Q , denoted $Q \circ P$, is the TP obtained by merging DN_P with $\text{rt}(Q)$. The distinguished node of $Q \circ P$ is DN_Q . Fig.2 (d) shows the extension of V (Fig.2 (b)) using Q_1 (Fig.2 (c)). It is easy to see that, for any XML tree t , $(Q \circ P)(t) = Q(P(t))$, that is, $(Q \circ P)(t)$ is equivalent to the union of answer sets of Q on the subtrees in $P(t)$.

Let $P_1, \dots, P_n$ be comparable TPs, and Q be a TP such that $\text{rt}(Q)$ and DN_{V_i} have identical labels. We denote by $Q \circ (P_1 \cap \dots \cap P_n)$ the extension of $P_1 \cap \dots \cap P_n$ using Q , which returns, for any t , $Q(P_1(t) \cap \dots \cap P_n(t))$.

2.3 TP Containment

Let P and Q be TPs. P is said to be *contained* in Q , denoted $P \subseteq Q$, if for every XML tree t , $P(t) \subseteq Q(t)$. It is shown in [1] that $P \subseteq Q$ iff there is a containment mapping from Q to P . Recall: A *containment mapping* (CM) from Q to P is a mapping δ from $N(Q)$ to $N(P)$ that is label-preserving, root-preserving as discussed in the last section, structure-preserving (which now means for any $/$ -edge (x, y) in Q , $(\delta(x), \delta(y))$ is a $/$ -edge in P , and for any $//$ -edge (x, y) , there is a path from $\delta(x)$ to $\delta(y)$ in P) and is output-preserving, which means $\delta(\text{DN}_Q) = \text{DN}_P$. A *homomorphism* from Q to P is similar to a CM, except there is no requirement of output-preserving.

The following lemma is proved in [13].

Lemma 1. For TPs $P_1, \dots, P_n, P \in P^{\{/, //, \square\}}$, $P \subseteq P_1 \cup \dots \cup P_n$ iff there is $i \in [1, n]$ such that $P \subseteq P_i$.

3 TP Rewriting Using Intersections of Views

A *view* is an existing TP. We define rewritings using intersections of views as follows.

Definition 1. Let Q be a query, and $V_1, \dots, V_n$ be comparable views. If $V_1 \cap \dots \cap V_n$ is non-empty, and there exists Q' such that $\text{label}(\text{rt}(Q')) = \text{label}(\text{DN}_{V_1})$, and $Q' \circ (V_1 \cap \dots \cap V_n) \subseteq Q$, then we say $Q' \circ (V_1 \cap \dots \cap V_n)$ is a *contained rewriting* (CR) of Q using $V_1 \cap \dots \cap V_n$. If $Q' \circ (V_1 \cap \dots \cap V_n) = Q$, we say Q' is an *equivalent rewriting* (ER) using V . The *maximal contained rewriting* (MCR) of Q using $V_1 \cap \dots \cap V_n$, denoted $\text{MCR}(Q, V_1 \cap \dots \cap V_n)$, is the union of all CRs of Q using $V_1 \cap \dots \cap V_n$.

Note that when $n = 1$, the above definition reduces to those in [6] (CR and MCR) and [14] (ER).

3.1 Properties of Intersections

In this section we identify conditions under which the intersection of TPs are not empty, and present algorithms to translate the intersection into the union of TPs. We present these results using two TPs. Generalization to more TPs is simple.

In many cases the intersection of two TPs P and Q is an *empty query*, that is, it always returns the empty set. For example, when $P = a/b/x$ and $Q = a/c/x$. The question arises: when is $P \cap Q$ a non-empty query? We have the follow answer to this question.

Lemma 2. $P \cap Q$ is non-empty iff there is a path P' in $P^{\{//\}}$ such that $P' \subseteq DP_P$, and $P' \subseteq DP_Q$.

The lemma is true because, when there is no DTD, every TP is non-empty, and $P \cap Q$ is non-empty iff $DP_P \cap DP_Q$ is non-empty.

Let's call the path P' in lemma 2 a *common distinguished path* of P and Q . If DP_P and DP_Q do not have $//$ -edges, then only when $DP_P = DP_Q$ there is a CDP of P and Q , which is DP_P itself. However, if DP_P and DP_Q do have $//$ -edges, then there may be (infinitely) many CDPs of P and Q . For example, if $P = a//x[b]//y$, $Q = a//x[c]//y$, then P and Q have the CDPs $a//x//y$, $a//x//x//y$, $a//x//x//x//y$ and so on. But there are only a finite number of CDPs that are of interest to us. These CDPs are annotated to form annotated CDPs.

Definition 2. Let P and Q be comparable TPs in $P^{\{//, []\}}$. An annotated CDP (ACDP) of P and Q is a CDP P' of P and Q such that (1) every node in P' is annotated with either 1, or 2, or both, (2) there is a (unique) CM m_P from DP_P to P' such that every node in DP_P is mapped a node annotated with 1 (or both 1 and 2), and for every node v in P' annotated with 1 (or both 1 and 2), there is a unique node u in DP_P such that $m_P(u) = v$, (3) there is a (unique) CM m_Q from DP_Q to P' such that every node in DP_Q is mapped a node annotated with 2 (or both 1 and 2), and for every node v in P' annotated with 2 (or both 1 and 2), there is a unique node u in DP_Q such that $m_Q(u) = v$.

Intuitively, an ACDP of P and Q is a path in $P^{\{//\}}$ which contains exactly one “copy” of DP_P and one “copy” of DP_Q (the nodes annotated with 1 form a copy of DP_P , and the nodes annotated with 2 form a copy of DP_Q) and every node appears in at least one of the copies. Fig.3 shows all ACDPs of $P = a//x[b]//y$ and $Q = a//x[c]//y$.

Next we present an algorithm that finds all ACDPs of P and Q . Let $P_1 = DP_P$ and $P_2 = DP_Q$. We can find all ACDPs of P and Q by calling the function $MERGE(P_1, P_2)$. In the function, a *position* in P_j refers to either a node or a $//$ -edge in P_j . Each position in P_j is given a unique position number, with

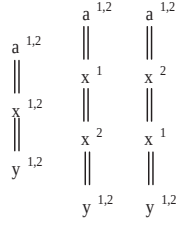


Fig. 3. ACDPs of $a//x[b]//y$ and $a//x[c]//y$

the position number of $rt(P_j)$ being 0, and each subsequent position's position number increases by 1. The function $MERGE(P_i, P_j)$ finds the ACDPs of P_i and P_j by “inserting” the nodes of P_i into P_j . To do so, it needs to find a position, in P_j , for every node in P_i . There are three stages in the process. The first stage (lines 1-6) is to scan P_i top-down, marking each node v in P_i with a set of markings. Each marking is of the form $\{s : A(v, s)\}$ (except for $rt(P_i)$, which is marked $\{0\}$), where s is a position, and $A(v, s)$ is a set of positions. The meaning of this marking is that if $\text{parent}(v)$ is merged into P_j in position s , then the possible positions where v can be inserted in P_j are those in $A(v, s)$. The criterion to choose the positions in $A(v, s)$ is label-preservation and structure-preservation. The second stage (lines 8-10) removes the impossible positions from the markings, going from bottom-up. The only possible position for DN_{P_i} is the last position in P_j . Based on structure-preservation, if impossible positions for v are found and deleted, then impossible positions for $\text{parent}(v)$ may be found and deleted. In the last stage (lines 13-29), if every node has a possible position, we pick a position s_v for each node v in P_i according to the markings, and construct a ACDP. For each combination of positions (each combination has one position for every node in P_i) there is a ACDP constructed. Finally the function returns all ACDP constructed this way.

Example 3. Let $P_1 = a//x//y/y$ and $P_2 = a//x//x//y$. Fig.4 shows the process of running $MERGE(P_1, P_2)$.

First, we identify and label each position in P_2 . There are 7 positions labeled $0, 1, \dots, 6$, as shown in the figure. In stage 1, we mark each node in P_2 with its possible positions. The a -node is marked $\{0\}$. If the a -node is put in position 0, then the possible positions of the next node are 1,2,3,4,5. Therefore, the first x -node is marked with $(0 : \{1, 2, 3, 4, 5\})$. If the x -node is put to position 1, then the next node may be put to positions 1,3 or 5, to preserve label and structure of P_i . If the x -node is put to position 2, then the possible positions for its child are 3 and 5, ..., if the x -node is put to position 5, then its child must be put to position 5 (after the x -node). Therefore, the child of the x -node is marked with the markings as shown in the figure. Finally, the last node of P_1 must be put to the last position, position 6, in P_2 . Thus if its parent is put to position 1 or 3, then there are no possible positions for it. If its parent is put to position 5, the

Algorithm 1. MERGE(P_i, P_j)

```

1: let  $pos(rt(P_i)) = \{0\}$ ,  $S = \emptyset$ 
2: for every subsequent node  $v$  in  $P_i$  do
3:   for all  $s \in pos(\text{parent}(v))$  do
4:     let  $\text{position}(v, s) = \text{FIND\_POSITION}(v, P_i, s, P_j)$ 
5:     mark  $v$  with  $(s : \text{position}(v, s))$ 
6:   let  $pos(v) = \bigcup_{s \in pos(\text{parent}(v))} \text{position}(v, s)$ 
7:   if  $pos(v) = \emptyset$  then return  $\emptyset$ 
8: for each node  $v$  in  $P_i$  (starting from  $v = \text{DN}_{P_i}$ ) do
9:   while  $\exists$  marking  $(s : \text{position}(v, s))$  for  $v$  such that  $\text{position}(v, s) = \emptyset$  do
10:    delete  $s$  from  $\text{position}(\text{parent}(v), s')$  for all  $s'$ 
11:   if  $pos(v) = \emptyset$  return  $\emptyset$ 
12: for each node  $v$  in  $P_i$  choose a position  $s_v$  from its markings. Initially,  $v = rt(P_i)$  and  $s_v = 0$ .
    For each subsequent node  $v$ , pick a position from  $\text{position}(v, s_{\text{parent}(v)})$ 
13: for each combination of positions found above do
14:   let  $P' = P_j$ , annotate every node with  $j$ 
15:   for  $v$  in  $P_i$  do
16:     if position  $s_v$  points to node  $u$  in  $P'$  then
17:       annotate  $u$  with  $i$ .
18:       if  $\text{prt}(u)$  is annotated with  $i$  or  $i, j$  and  $(\text{parent}(v), v)$  is  $/$ -edge then
19:         change  $(\text{prt}(u), u)$  to  $/$ -edge
20:       else if position  $s_v$  represents  $//$ -edge  $(u_1, u_2)$  in  $P'$  then
21:         insert a  $\text{label}(v)$ -node  $u_0$  between  $u_1$  and  $u_2$ 
22:         if  $u_1$  is annotated with  $i$  or  $i, j$  then
23:           let edge  $(u_1, u_0)$  be of the same type as  $(\text{parent}(v), v)$ 
24:         else
25:           let edge  $(u_1, u_0)$  be of type  $//$ 
26:           let edge  $(u_0, u_2)$  be of type  $//$ ; let position  $s_v$  point to this edge  $(u_0, u_2)$ 
27:   add  $P'$  to  $S$ 
28: return  $S$ 

```

it can go to position 6. This explains the markings of the last node of P_1 . In stage 2, we remove the impossible positions in the markings. Because position 1 and 3 of the first y -node prohibits the second y -node to be put in position 6 (as indicated by the markings $(1, \{\})$ and $(3, \{\})$, we know positions 1 and 3 are impossible for the first y -node. Thus we delete them from its markings. In stage 3, for each combination of the positions, we construct an ACDP. A combination of position is made of a position for each node in P_1 . In this example, the combinations of positions are $(0, 1, 5, 6)$, $(0, 2, 5, 6)$, $\dots$, $(0, 5, 5, 6)$. We use $(0, 5, 5, 6)$ to explain the construction process. Initially $P' = P_2$ and every node in P' is annotated with 2. Position 0 points to the root of P_2 , therefore we annotate $rt(P_1)$ with 1. Position 5 points to a $/$ -edge, therefore we insert an x -node, x_0 , in this position and annotate this node with 1. The edge (x, x_0) and (x_0, y) are to be of type $//$ in this case. Now position 5 points to the edge (x_0, y) . Since the position for the next node is also 5, we insert another x -node, x_1 , between x_0 and y , and annotate x_1 with 1. The last node of P_1 has position 6, so we annotate the y -node in P_2 with 1. Since x_1 is annotated with 1, we can change the the edge type of (x_1, y) to that of the corresponding edge in P_1 , in this case, $/$. The resulting P' is a ACDP: it is $a^{1,2} // x^2 // x^2 // x^1 // y^1 / y^{1,2}$, where the superscripts indicate the annotation. The ACDPs constructed using other combinations are: $a^{1,2} // x^1 // x^2 // x^2 // y^1 / y^{1,2}$, $a^{1,2} // x^{1,2} // x^2 // y^1 / y^{1,2}$, $a^{1,2} // x^2 // x^1 // x^2 // y^1 / y^{1,2}$, and $a^{1,2} // x^2 // x^{1,2} // y^1 / y^{1,2}$.

Algorithm 2 FIND_POSITION(v, P_i, s, P_j)

```

1: if  $s$  is node  $u$  in  $P_j$  then
2:   if (parent( $v$ ),  $v$ ) is //-edge then
3:     let  $A$  consist of all //-edges after  $u$  and all nodes labeled  $label(v)$  after  $u$ .
4:   else if (parent( $v$ ),  $v$ ) is /-edge then
5:     if  $u$  has child  $u'$  then
6:       if ( $u, u'$ ) is //-edge then add this //-edge to  $A$ 
7:       if  $label(u') = label(v)$  then add  $u'$  to  $A$ 
8: if  $s$  is //-edge ( $u_1, u_2$ ) in  $P_j$  then
9:   add  $s$  to  $A$ 
10:  if (parent( $v$ ),  $v$ ) is //-edge then
11:    add all //-edges after  $u_2$  and all nodes labeled  $label(v)$  after  $u_1$  to  $A$ 
12:  if (parent( $v$ ),  $v$ ) is /-edge and  $label(u_2) = label(v)$  then add  $u_2$  to  $A$ 
13: if  $v$  is the last node in  $P_i$  then
14:   delete from  $A$  all positions except that of  $\text{DN}_{P_j}$ 
15: return  $A$ 

```

Number of ACDPs. Let n and m be the number of edges in P_i and P_j respectively, and $f(m, n)$ be the worst-case number of ACDPs of P_i and P_j (which occurs when all nodes in P_i and P_j have the same label, and all edges are // -edges). The following theorem can be proved using induction on m and n .

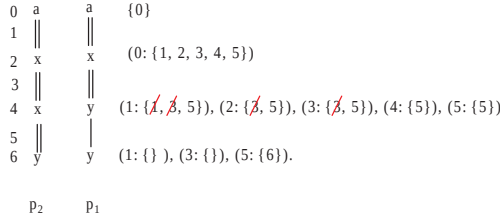


Fig. 4. Finding ADCPs of P_1 and P_2

Theorem 1. $f(m, n) = f(n, m)$, which can be calculated recursively as follows:

$$\begin{aligned}
 f(m, n) = & f(m-1, n) + 2(f(m-1, n-1) \\
 & + f(m-1, n-2) + \dots + f(m-1, 1)).
 \end{aligned}$$

For example, $f(m, 1) = 1$, $f(2, 2) = 3$, $f(3, 2) = 5$, $f(3, 3) = 13$, $f(4, 3) = 25$, $f(4, 4) = 63$ and so on. Thus $f(m, n)$ grows exponentially in general. However, in most practical cases, the number of ACDPs is much smaller than $f(m, n)$.

Complexity. Algorithm MERGE(P_i, P_j) runs in $O(|P_i| \times |P_j|^2)$, where $|P_i|$ is the number of nodes in P_i ; function FIND_POSITION runs in $O(|P_j|)$, the top-down scan visits each node in P_i once, and for each node in P_i , the function FIND_POSITION is called at most $2|P_j|$ times. The bottom-up scan and the construction of ACDPs can be done in $O(|P_i| \times |P_j|)$.

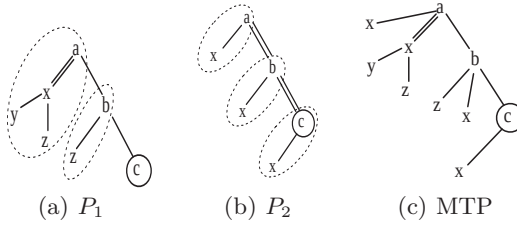


Fig. 5. TPs P_1 , P_2 and the MTP of P_1, P_2

Let P be a TP in $P^{\{//, \square\}}$. For every node v in P , we use P_v to denote the subpattern of P rooted at v . Let v be a node in DP_P , and u be the child of v on DP_P (if u exists). We call the subpattern obtained by removing P_u from P_v the *branching subtree at v* . For example, in Fig.5 (a), the branching subtrees are indicated by the dotted oval. Next we define *merged TPs (MTPs)* of P_1 and P_2 .

Definition 3. Let P_1 and P_2 be comparable TPs, and P' be an ACDP of P_1 and P_2 . Let δ_i be the unique CM from P_i to P' that maps every node in P_i to a node in P' annotated with i . The merged TP (MTP) of P_1 and P_2 wrt to P' is the TP obtained as follows: for every node v in P_i , add the branching subtree at v_i under $\delta_i(v_i)$.

Fig.5 (c) shows the only MTP of the TPs in Figures 5 (a) and (b).

The following theorem is straightforward.

Theorem 2. Let P_1 and P_2 be TPs in $P^{\{//, \square\}}$. The union of all MTPs of P_1 and P_2 is equivalent to $P_1 \cap P_2$.

Note also that the subpatterns of the MTPs rooted at the distinguished nodes are all identical.

3.2 Finding MCRs Using $V_1 \cap V_2$

Let Q be the query and V_1 and V_2 be comparable views. We assume $V_1 \cap V_2$ is not empty, and $V_1 \not\subseteq V_2$, $V_2 \not\subseteq V_1$. Suppose $V_1 \cap V_2$ is equivalent to $V'_1 \cup \dots \cup V'_k$. Clearly $Q' \circ (V_1 \cap V_2) \subseteq Q$ if and only if $Q' \circ V'_i \subseteq Q$ for all $i \in [1, k]$. In other words, Q' is a CR of Q using $V_1 \cap V_2$ iff it is a CR of Q using V'_i for all $i \in [1, k]$. Therefore, to find the MCR of Q using $V_1 \cap V_2$, we can find the MCR of Q using each V'_i and intersect them. That is,

$$\text{MCR}(Q, V_1 \cap V_2) = \bigcap_{i=1}^k \text{MCR}(Q, V'_i).$$

Example 4. Consider the views V_1, V_2, V'_1, V'_2 and the query Q in Fig.6. $V_1 \cap V_2 = V'_1 \cup V'_2$. We find the MCR of Q using V'_1 , which is y/z , and the MCR of Q using V'_2 , which is also y/z . Thus y/z is the MCR of Q using $V_1 \cap V_2$.

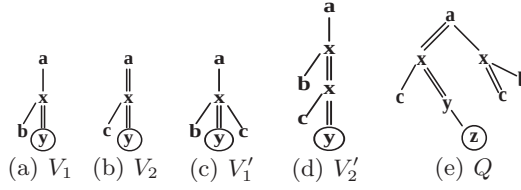


Fig. 6. Finding MCR using intersection

Algorithm 3. Finding equivalent rewriting

```

1: for  $i = 1$  to  $n$  do
2:   if exists node  $v$  such that  $DP_Q^v$  is isomorphic to  $p_i$  then
3:     if  $sub_Q^v \circ V_i' = Q$  then
4:       if  $\forall j \in [1, k], V_j' \subseteq V_i'$  then
5:         return  $sub_Q^v$ 

```

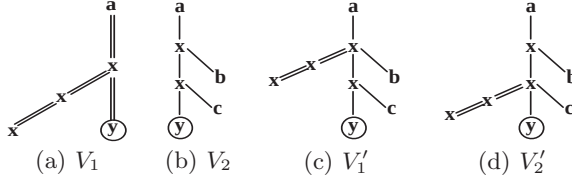


Fig. 7. Example for illustrating ER using intersection

3.3 Finding ERs Using $V_1 \cap V_2$

Suppose Q has an ER Q' using $V_1 \cap V_2$, and $V_1 \cap V_2 = V_1' \cup \dots \cup V_k'$, that is, $Q' \circ (V_1' \cup \dots \cup V_k') = Q$. By Lemma 1, there exists $i \in [1, k]$, such that $Q' \circ V_i' = Q$. Hence for all $j \in [1, k]$, $Q' \circ V_j' \subseteq Q' \circ V_i'$, and thus there is a CM from V_i to V_j , so the length of $DP_{V_j'}$ cannot be longer than that of $DP_{V_i'}$. Furthermore, $DP_{Q' \circ V_i'}$ is isomorphic to DP_Q , and the subpattern of Q rooted at the node that corresponds to $DN_{V_i'}$, is an ER of Q using V_i' (by Lemma 4.8 of [14]).

Using the properties above, we provide a heuristic algorithm, Algorithm 3, for finding ERs using $V_1 \cap V_2$. In the algorithm, we assume $p_1, \dots, p_n$ ($n \leq k$) are the shortest ACDPs of V_1 and V_2 , and the corresponding MTPs are $V_1', \dots, V_n^1$. For any node v on DP_Q , the path from $rt(Q)$ to v is denoted DP_Q^v , and the subpattern rooted at v is denoted sub_Q^v . The basic idea of the algorithm is as follows. For each shortest ACDP p_i , we first check whether it is isomorphic to some DP_Q^v , if not, there is no ER using V_i' ; if yes, we further check whether sub_Q^v is an ER using V_i' . If yes, we further check whether all other MTPs of V_1 and V_2 are contained in V_i . If yes, sub_Q^v is returned as an ER of Q using $V_1 \cap V_2$. It is easy to prove the following theorem.

¹ Note that although there may be many ACDPs of V_1 and V_2 , usually there are few shortest ones.

Theorem 3. (1) If Q has an ER using $V_1 \cap V_2$, then $V_1 \cap V_2$ is equivalent to a single TP V . (2) Algorithm 3 finds the ER if it exists.

Example 5. (1) Consider the views in Fig.6. Since $V_1 \cap V_2 = V'_1 \cup V'_2$, $DP_{V'_1}$ is the shorter than $DP_{V'_2}$, and $V'_2 \not\subseteq V'_1$, we know there is no ER of Q using $V_1 \cap V_2$ for any Q . (2) Consider the two views V_1 and V_2 in Fig.7. There are two MTPs V'_1 and V'_2 , and $V'_2 \subseteq V'_1$. Therefore, for any TP Q , Q' is an ER of Q using $V_1 \cap V_2$ iff Q' is an ER using V'_1 .

4 Rewriting Using Other Combinations of Views

We now define a second type of rewritings of Q using multiple views that does not require the views to be comparable.

Definition 4. Let $V_1, \dots, V_n$ be some views with identical root label (possibly $V_i = V_j$ for some i, j), and Q be a query. If there are TPs $Q_1, \dots, Q_n$ such that $\bigcap_{i=1}^n (Q_i \circ V_i) \subseteq Q$, and $\bigcap_{i=1}^n (Q_i \circ V_i)$ is non-empty, then we say $\langle Q_1, \dots, Q_n \rangle$ is a contained rewriting of Q using $\langle V_1, \dots, V_n \rangle$. If $\bigcap_{i=1}^n (Q_i \circ V_i) \supseteq Q$ also holds, we call the CR an equivalent rewriting.

Intuitively, if there is a contained rewriting $\bigcap_{i=1}^n (Q_i \circ V_i)$, then to partially answer Q over t , we can evaluate Q_i over $V_i(t)$ and then find the intersection $\bigcap_{i=1}^n Q_i(V_i(t))$. Note that, when $n = 1$, the definition reduces to that in [6] (CR and MCR) and [14] .

4.1 Relationship between Rewritings Using $\langle V_1, V_2 \rangle$ and Rewritings Using $V_1 \cap V_2$

First, it is easy to prove the following lemma.

Lemma 3. Let V_1 and V_2 be comparable views. Then $Q \circ (V_1 \cap V_2) \subseteq (Q \circ V_1) \cap (Q \circ V_2)$.

However, generally $Q \circ (V_1 \cap V_2) \not\subseteq (Q \circ V_1) \cap (Q \circ V_2)$. Consider $V_1 = a//b/x$ and $V_2 = a//c/x$, and $Q = x//z$. Clearly $V_1 \cap V_2 = \emptyset$, hence $Q \circ (V_1 \cap V_2) = \emptyset$. However, $(Q \circ V_1) \cap (Q \circ V_1) \neq \emptyset$. This example also shows that sometimes even if $V_1 \cap V_2$ is empty, it is still possible to have CRs of Q using $\langle V_1, V_2 \rangle$, although there are clearly no CRs of Q using $V_1 \cap V_2$.

The next lemma identifies some special cases where $Q \circ (V_1 \cap V_2) = (Q \circ V_1) \cap (Q \circ V_2)$.

Lemma 4. Let V_1 and V_2 be comparable views. If one of the following conditions holds, then $Q \circ (V_1 \cap V_2) = (Q \circ V_1) \cap (Q \circ V_2)$.

- (1) $DP_{V_1} = DP_{V_2}$, and all edges in DP_{V_1} are $/$ -edges.
- (2) Every edge in DP_Q is a $/$ -edge.
- (3) $V_1 \subseteq V_2$ or $V_2 \subseteq V_1$.

Using the above lemmas, we can prove the following theorem:

Theorem 4. *Let V_1, V_2 be comparable views. For any query Q , if there is a CR of Q using $V_1 \cap V_2$, then there is a CR of Q using $\langle V_1, V_2 \rangle$.*

Note that the above theorem does not say there is CR using $\langle V_1, V_2 \rangle$ which contains the CR using $V_1 \cap V_2$. The next example shows that it is possible for Q to have a CR using $V_1 \cap V_2$, and this rewriting is not contained in any CRs of Q using $\langle V_1, V_2 \rangle$.

Example 6. Let $V_1 = a//x//z$, $V_2 = a//y//z$, and $Q = a//x//y//z//c$. It can be verified that $V_1 \cap V_2 = a//x//y//z$. Now let $Q' = z//c$, then $Q' \circ (V_1 \cap V_2) = Q$. Thus Q' is an ER of Q using $V_1 \cap V_2$.

If there are Q_1 and Q_2 such that $(Q_1 \circ V_1) \cap (Q_2 \circ V_2) \subseteq Q$, then DP_{Q_1} must be z/c or $z//c$, and DP_{Q_2} must be z/c or $z//c$, because $\text{rt}(Q_i)$ must be labeled z and DN_{Q_i} must be labeled c . One can verify that if $\text{DP}_{Q_2} = z//c$, then $(Q_1 \circ V_1) \cap (Q_2 \circ V_2)$ is not contained in Q , and if $\text{DP}_{Q_2} = z/c$, then $(Q_1 \circ V_1) \cap (Q_2 \circ V_2)$ is not equivalent to Q either.

5 The Presence of Non-recursive DTDs

In the following, we assume every TP P is satisfiable under a non-recursive DTD G , that is, there is an XML tree t which conforms to G , and $P(t) \neq \emptyset$.

In the presence of G , no label in an XML tree can appear in a path more than once. Thus any TP that is satisfiable under G cannot have a path that contains two or more nodes with the same label. Therefore, for any comparable views V_1 and V_2 , there is at most one ACDP of V_1 and V_2 , and at most one MTP of V_1 and V_2 . In other words, $V_1 \cap V_2$ is equivalent to a single TP V' under G . Therefore, to find the MCR or ER of Q using $V_1 \cap V_2$, we only need find the MCR or ER of Q using V' , and this can be done using the method of [6]. Furthermore, we can prove (see full version of this paper) the following theorem, which implies that, in the presence of G , Q' is a CR using $V_1 \cap V_2$ iff $\langle Q', Q' \rangle$ is a CR of Q using $\langle V_1, V_2 \rangle$.

Theorem 5. *Let V_1 and V_2 be comparable views, and $\text{label}(\text{rt}(Q')) = \text{label}(\text{DN}_{V_1})$. In the presence of G , $Q' \circ (V_1 \cap V_2)$ is equivalent to $(Q' \circ V_1) \cap (Q' \circ V_2)$.*

6 More Related Work

Besides the recent works discussed in Section 1, several other papers dealt with tree pattern query rewriting. In particular, [10] studied the problem of query answerability using views for general XPath queries, that is, given Q and $V_1, \dots, V_n$, whether there are $Q_1, \dots, Q_n$ such that $Q_1 \circ V_1 \cup \dots \cup Q_n \circ V_n = Q$. [3] addressed the problem of answering XPath queries using a single materialized view where, for the view, a combination of node references, typed data values, and full paths may be stored. However, the way in which a query is answered using the view

is different from ours (and those in Section 1): one can follow node references to go to the original document, so the original XML tree cannot be discarded. [2] studied a different type of equivalent rewriting using multiple views in the presence of *structural summaries* and integrity constraints: the answer sets of the views are nodes rather than subtrees, and the answers to the new query are obtained by combining answers to the views through a number of algebraic operations. [11] studied *correct rewritings* of TPs, using a single view, which can be seen as a special form of contained rewritings. [4] attempted to speed-up the finding of MCRs using single views by combining the views into a single tree. [12] studied *equivalently answering XPath* queries using multiple views based on the assumption that the Dewey codes are stored in the materialized views so that the common ancestors of nodes in different views can be found. Our work is clearly different from all of the above.

7 Conclusion

We studied the problem of rewriting TP queries using multiple views for the class $P^{\{\text{/,}\cdot\}}$, and defined rewritings using two different combinations of views. We studied the relationship between the two types of rewritings and presented efficient algorithms to reformulate the intersection of TPs into a union of TPs, as well as algorithms for finding the MCRs and ERs using intersections of views. Our definitions and algorithms enable us to make better use of the views in order to answer a query.

Acknowledgement. This work is partially supported by Griffith University New Researcher's Grant (GUNRG36621) and grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (CUHK418205). The authors are grateful for helpful comments by Professor Rodney Topor.

References

1. Amer-Yahia, S., Cho, S., Lakshmanan, L.V.S., Srivastava, D.: Minimization of tree pattern queries. In: SIGMOD (2001)
2. Arion, A., Benzaken, V., Manolescu, I., Papakonstantinou, Y.: Structured materialized views for XML queries. In: VLDB (2007)
3. Balmin, A., Özcan, F., Beyer, K.S., Cochrane, R., Pirahesh, H.: A framework for using materialized XPath views in XML query processing. In: VLDB (2004)
4. Gao, J., Wang, T., Yang, D.: MQTree based query rewriting over multiple XML views. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 562–571. Springer, Heidelberg (2007)
5. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. 10(4) (2001)
6. Lakshmanan, L.V.S., Wang, H., Zhao, Z.J.: Answering tree pattern queries using views. In: VLDB (2006)
7. Mandhani, B., Suciu, D.: Query caching and view selection for XML databases. In: VLDB (2005)
8. Miklau, G., Suciu, D.: Containment and equivalence for an XPath fragment. In: PODS (2002)

9. Onose, N., Deutsch, A., Papakonstantinou, Y., Curtmola, E.: Rewriting nested XML queries using nested views. In: SIGMOD (2006)
10. Tajima, K., Fukui, Y.: Answering XPath queries over networks by sending minimal views. In: VLDB (2004)
11. Tang, J., Zhou, S.: A theoretic framework for answering XPath queries using views. In: XSym. (2005)
12. Tang, N., Yu, J.X., Özsu, M.T., Choi, B., Wong, K.-F.: Multiple materialized view selection for xpath query rewriting. In: ICDE (2008)
13. Wang, J., Yu, J.X., Liu, C.: On tree pattern rewriting using views. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 1–12. Springer, Heidelberg (2007)
14. Xu, W., Özsoyoglu, Z.M.: Rewriting XPath queries using materialized views. In: VLDB (2005)

Superimposed Code-Based Indexing Method for Extracting MCTs from XML Documents

Wenxin Liang^{1,4}, Takeshi Miki², and Haruo Yokota^{3,4}

¹ CREST, Japan Science and Technology Agency (JST)

² Nomura Research Institute

³ Department of Computer Science, Tokyo Institute of Technology

⁴ Global Scientific Information and Computing Center, Tokyo Institute of Technology
{wxliang, takeshi}@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp

Abstract. With the exponential increase in the amount of XML data on the Internet, information retrieval techniques on tree-structured XML documents such as keyword search become important. The search results for this retrieval technique are often represented by minimum connecting trees (MCTs) rooted at the lowest common ancestors (LCAs) of the nodes containing all the search keywords. Recently, effective methods such as the stack-based algorithm for generating the lowest grouped distance MCTs (GDMCTs), which derive a more compact representation of the query results, have been proposed. However, when the XML documents and the number of search keywords become large, these methods are still expensive. To achieve more efficient algorithms for extracting MCTs, especially lowest GDMCTs, we first consider two straightforward LCA detection methods: keyword B⁺ trees with Dewey-order labels and superimposed code-based indexing methods. Then, we propose a method for efficiently detecting the LCAs, which combines the two straightforward indexing methods for LCA detection. We also present an effective solution for the false drop problem caused by the superimposed code. Finally, the proposed LCA detection methods are applied to generate the lowest GDMCTs. We conduct detailed experiments to evaluate the benefits of our proposed algorithms and show that the proposed combined method can completely solve the false drop problem and outperforms the stack-based algorithm in extracting the lowest GDMCTs.

1 Introduction

Recently, there has been an exponential increase in the amount of data, such as life science data [22, 20], bibliography data [24] and online encyclopedia data [23], that are disseminated and shared over the Internet in the form of XML documents. These are often modeled as ordered labeled trees. Information retrieval techniques on tree-structured XML documents such as keyword search are therefore important. Keyword search allows users to find relevant information without any prior knowledge of the schema of the underlying data or any need to learn complex queries [1, 2, 4, 25, 11, 14]. For example, assume an XML document consists

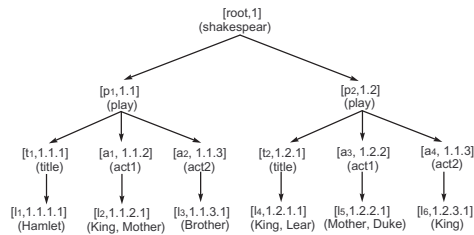


Fig. 1. An example XML document tree (labeled by Dewey number)

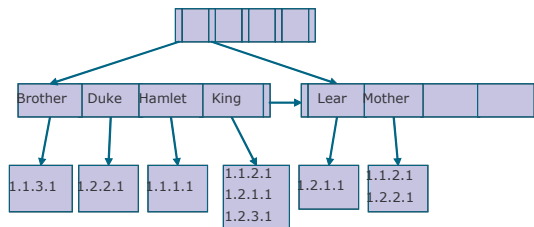


Fig. 2. An example keyword B+tree

of Shakespeare's plays in Figure 1. Users might be interested in finding the relationship between the query keywords *king* and *mother*. The search system returns the relevant answers corresponding to the query keywords that might all appear within the same act or in different acts but within the same play, and so on.

In keyword searches over XML documents, the search results are often represented by *minimum connecting trees* (MCTs) rooted at the *lowest common ancestors* (LCAs) of the nodes containing all the query keywords. Therefore, keyword searching over XML document trees resolves the problem of detecting the LCAs of the nodes that contain all the query keywords. For example, the answer for the query by keywords *king* and *mother* over the XML tree of Figure 1 might be the subtrees rooted at $[a_1]$ and $[p_2]$.

Recently, many studies have been conducted on detecting the LCAs of the nodes containing query keywords over XML documents [25, 8, 14, 12, 18]. However, these methods only focus on finding LCAs and do not consider techniques for extracting XML subtrees rooted at the LCAs. To provide effective query answers to the users, the query system must efficiently extract the MCTs that are the subtrees rooted at the LCAs containing all the keywords. Hristidis et al. [10] propose an efficient stack-based algorithm for computing the MCTs and the lowest grouped distance MCTs (GDMCTs) that derive more compact representation of the query results. However, Hristidis's algorithm still results in expensive time complexity when handling large XML documents with complex keyword queries. Therefore, a more efficient algorithm for extracting MCTs, especially the lowest GDMCTs containing all the query keywords, is necessary.

We make the following main technical contributions in this paper: 1) To effectively detect the LCAs of the nodes containing all the query keywords, we first consider two straightforward indexing methods: keyword B^+ trees with Dewey-order labels and superimposed code-based methods. In the first method, the nodes of the XML tree are labeled by Dewey numbers and stored in a B^+ tree index. Then, the LCAs of the nodes containing the query keywords can be found by comparing the Dewey-order label of each node containing the corresponding keyword. In the superimposed code-based method, fixed-length superimposed codes (signatures) are first assigned to the nodes of the XML tree. Second, the query signature is determined by the logical OR of the signatures of all the query keywords. Finally, the LCAs of nodes containing all the query keywords are determined by the logical AND of the query signature and the signature of each node of the XML tree. However, false drop problems may occur in the superimposed code-based method; 2) Keyword B^+ trees with Dewey-order labels and superimposed code-based methods are effective in finding the LCAs but expensive in query cost. To reduce the query cost, we propose an efficient LCA detection method that combines the keyword B^+ tree with the Dewey-order label and the superimposed code-based indexing methods. In this method, both the superimposed codes and the Dewey-order labels of the nodes are first stored in a B^+ tree index. Second, it searches for the leaf nodes in the B^+ tree by the signature of any one query keyword. After finding the leaf nodes, the LCAs are determined by a logical AND operation between the query signature and those of the corresponding nodes. The combined method reduces the comparisons of many internal nodes and the query can be completed using only one query keyword. Therefore, it is superior in performance compared with the two original methods. However, the false drop problem still cannot be avoided; 3) We also present an effective method to solve the false drop problem. In this method, the signatures of all the keywords are used to find the corresponding leaf nodes in the B^+ tree index. Then, the LCAs can be determined by detecting the common Dewey-order labels of the corresponding nodes; 4) We apply the proposed LCA detection methods to generate the lowest GDMCTs. We perform experiments to evaluate the performance of detecting LCAs comparing the proposed combined method with the original ones. The experimental results indicate that our proposed LCA detection method is cost-efficient and can completely solve the false drop problem. We also conduct experiments to compare the query time using our proposed methods with that using the stack-based method. The experimental results show that the proposed combined method can completely solve the false drop problem and outperforms the previously known stack-based method in extracting the lowest GDMCTs.

The remainder of the paper is organized as follows. In Section 2, we briefly introduce related work. Section 3 describes the notation and definitions of LCAs and MCTs. In Section 4, we discuss two straightforward methods for finding LCAs and propose an efficient method combining the two methods. In Section 5, we describe the lowest GDMCTs extraction algorithms based on the proposed LCA detection methods and compare the query costs of the proposed methods

and the SA algorithm. Section 6 describes the experimental evaluation, shows the benefits of our approach and compares it with the stack-based algorithm. Finally, Section 7 concludes this paper and outlines future work.

2 Related Work

The first research area relevant to this work is LCA computation to detect the LCA of two or more nodes over tree-structured data such as XML documents. Computation of the LCA of two nodes has been intensively studied over the past 30 years. References [3] and [21] first introduced the problem of finding LCAs in trees. In [9], Harel and Tarjan introduced upper and lower bounds for the problem of LCAs. In [17], Schieber et al. presented a simpler algorithm with optimal asymptotic bounds for finding the LCAs in trees. References [9] and [15] showed that the LCA query can be computed in constant time after linear-time preprocessing in arbitrarily directed trees. However, these algorithms are still too complicated to implement effectively. In [25], Xu et al. proposed efficient algorithms for computing the smallest LCAs in XML databases using Dewey-order labels. Computing the LCA of nodes utilizing these labels does not require any disk access that would degrade the performance of the query. The concept of a binary superimposed code was first introduced by Kautz and Singleton [13]. Since then, it has been extensively studied and applied to many areas such as data security and cryptology [19, 7], broadcasting in radio networks [5] and so on. In our proposed method, we use Dewey-order labels and superimposed codes to find the LCAs.

The second area of research relevant to this paper is the work on keyword search over XML documents. XRANK [8] considered the problem of producing ranked results for keyword search queries in hierarchical and hyperlinked XML documents. A specific document fragment, namely the subtree, is returned as the keyword search results in XRANK. In [6], Cohen et al. proposed a semantic search engine over XML documents that employed more techniques of information retrieval than XRANK. However, these studies did not consider the problem of extracting the MCTs containing the query keywords. In [10], Hristidis et al. proposed an effective stack-based algorithm (SA) for computing the lowest GDMCTs rooted at the LCAs of nodes containing the query keywords. However, as XML documents and the number of query keywords become larger because of more complex queries, Hristidis's algorithm still causes high query costs.

3 Notation and Definitions

In this section, we introduce notation and definitions of the LCAs and MCTs used in this paper¹. An XML document is represented by the conventional order labeled tree T . Each node v of the XML tree T corresponding to an XML element or leaf is labeled with a tag or string value $\lambda(v)$. Each node is assigned a unique

¹ Some notation and definitions described in this section refer to the reference [10].

id(v) and a Dewey-order label $lab(v)$ as a 2-tuple column $[id(v), lab(v)]$, as shown in Figure 1.

Definition 1 (LCA). Given a set of n nodes $v_1, \dots, v_n$ and an input XML tree T , the LCA of the set of nodes $v_1, \dots, v_n$, $lca(v_1, \dots, v_n)$ is defined as the node v in T that is ancestor to all the nodes $v_1, \dots, v_n$, and is farthest from the root.

Definition 2 (MCT). Given a set of n nodes $v_1, \dots, v_n$ and an input XML tree T , the MCT of nodes $v_1, \dots, v_n$ is the minimum subtree T_m that connects $v_1, \dots, v_n$. Conversely, the MCT of nodes $v_1, \dots, v_n$ is the subtree rooted at the LCA of $v_1, \dots, v_n$.

Given a list of m keywords, $k_1, \dots, k_m$, the MCT of keywords $k_1, \dots, k_m$ is the MCT of nodes $v_1, \dots, v_n$ that contain $k_1, \dots, k_m$. For example, given two query keywords (*King*, *Mother*) for the XML tree in Figure 1, the MCTs containing both keywords are shown in (1 – 6) as follows, where each root of (1 – 6) is an LCA that contains the two keywords.

$$\begin{array}{ccc} \text{root} \rightarrow p_1 \rightarrow a_1 \rightarrow l_2 & p_2 \rightarrow t_2 \rightarrow l_4 & \text{root} \rightarrow p_1 \rightarrow a_1 \rightarrow l_2 \\ \searrow p_2 \rightarrow t_2 \rightarrow l_4 & \searrow a_3 \rightarrow l_5 & \searrow p_2 \rightarrow a_1 \rightarrow l_5 \\ (1) & (2) & (3) \end{array}$$

$$\begin{array}{ccc} a_1 \rightarrow l_2 & \text{root} \rightarrow p_1 \rightarrow a_1 \rightarrow l_2 & p_2 \rightarrow a_3 \rightarrow l_5 \\ (4) & \searrow p_2 \rightarrow a_4 \rightarrow l_6 & \searrow a_4 \rightarrow l_6 \\ & (5) & (6) \end{array}$$

Assume a list $l_i (1 \leq i \leq m)$ of nodes that contain keywords k_i . The number N of MCTs for keywords $k_1, \dots, k_m$ can be determined by the length of the list $|l_i|$ as follows:

$$N = |l_1| \times |l_2| \times \dots \times |l_m|. \quad (7)$$

Therefore, larger numbers of query keywords generate more MCTs. To reduce the redundancy of MCTs, a set of MCTs can be combined into *grouped distance trees*. We first define the distance MCT (DMCT) as follows.

Definition 3 (DMCT). Given a set of nodes $v_1, \dots, v_n$ and an input XML tree T , the DMCT T_d of the MCT T_m of nodes $v_1, \dots, v_n$ is the tree such that:

1. T_d contains $v_1, \dots, v_n$;
2. T_d contains $lac(v_i, v_j)$, where $v_i, v_j \in [v_1, \dots, v_n], i \neq j$; and
3. there is an edge labeled with the distance d between each node $v_1, \dots, v_n$ and $lac(v_i, v_j)$.

The DMCTs corresponding to (1 – 6) are shown in (8 – 13) as follows:

$$\begin{array}{ccc} \text{root} \xrightarrow{3} l_2 & p_2 \xrightarrow{2} l_4 & \text{root} \xrightarrow{3} l_2 \\ \searrow^3 l_4 & \searrow^2 l_5 & \searrow^3 l_5 \\ (8) & (9) & (10) \end{array}$$

$$a_1 \xrightarrow{1} l_2 \quad (11)$$

$$\begin{array}{c} \text{root} \xrightarrow{3} l_2 \\ \searrow^3 l_6 \end{array} \quad (12)$$

$$\begin{array}{c} p_2 \xrightarrow{2} l_5 \\ \searrow^2 l_6 \end{array} \quad (13)$$

However, because the number of DMCTs is the same as the number of MCTs, the problem of exponential explosion in the number of subtrees still cannot be resolved. Next, we define grouped DMCTs as follows.

Definition 4 (GDMCT). A grouped DMCT (GDMCT) T_g contains the DMCT T_d if T_d and T_g are isomorphic. Assume $\mathcal{M}$ is the mapping of the nodes in T_d to those in T_g , and $\mathcal{M}'$ is a corresponding mapping of the edges of T_d to those of T_g , which must meet the following conditions.

1. If v_d and v_g are nodes of T_d and T_g , respectively, and $\mathcal{M}(v_d) = v_g$, then the label of v_g contains the ID of v_d .
2. If e_d and e_g are edges of T_d and T_g , respectively, and $\mathcal{M}'(e_d) = e_g$, then the labels of v_d and v_g are the same.

For example, the following GDMCTs (14), (15) and (16) contain DMCTs (8, 10, 11), (9, 13) and (12), respectively.

$$\begin{array}{c} \text{root} \xrightarrow{3} [l_2, l_5] \\ \searrow^3 [l_2, l_4, l_6] \end{array} \quad (14) \quad \begin{array}{c} p_2 \xrightarrow{2} [l_5] \\ \searrow^2 [l_4, l_5] \end{array} \quad (15) \quad a_1 \xrightarrow{1} [l_2] \quad (16)$$

Therefore, grouping the DMCTs into GDMCTs can effectively reduce the number of results. However, there might be some GDMCTs with roots that are ancestors of other GDMCTs. The lowest GDMCT rooted at the smallest LCA does not contain any roots of other GDMCTs. The following gives the definitions of the smallest LCA and lowest GDMCT.

Definition 5 (Smallest LCA). Given a set of nodes $v_1, \dots, v_n$ in an input tree T , the smallest LCA of $v_1, \dots, v_n$ is the node v such that:

1. v is the LCA of $v_1, \dots, v_n$; and
2. v is not an ancestor of any other LCAs of $v_1, \dots, v_n$.

Definition 6 (Lowest GDMCT). Given a set of nodes $v_1, \dots, v_n$ in an input tree T , a GDMCT is a lowest GDMCT if it is rooted at the smallest LCAs of $v_1, \dots, v_n$.

For the same example we used before, the lowest GDMCTs of GDMCT (14), (15) and (16) are as follows:

$$\begin{array}{c} p_2 \xrightarrow{2} [l_5] \\ \searrow^2 [l_4, l_5] \end{array} \quad (17) \quad a_1 \xrightarrow{1} [l_2] \quad (18)$$

4 LCA Detection Method

In this section, we first consider two straightforward methods for detecting LCAs of nodes containing query keywords: the method using keyword B⁺ trees with Dewey-order labels and the method based on superimposed codes. However, the query cost is expensive using these methods because both require traversal of all the nodes of the XML tree. To reduce the query cost, we propose an efficient method combining both methods. We also present an effective solution for the false drop problem caused by the superimposed code.

4.1 Keyword B⁺tree with the Dewey-Order Label Method

Dewey order assigns a vector to each node representing the path from the root of the tree to the node. The containment relationship (parent-child and ancestor-descendant relationships) between two nodes can be conveniently and simply detected by the path: the common ancestor of a set of nodes can be found by comparing the Dewey-order labels of the nodes. For example, assume a query by keywords (*Hamlet*, *King*) in the XML tree of Figure 1. The common ancestor of the nodes $[l_1, 1.1.1.1]$ and $[l_2, 1.1.2.1]$ that contain the query keywords can be found by their Dewey-order labels: their common ancestor is the node $[p_1, 1.1]$, which has the common label of $[l_1, 1.1.1.1]$ and $[l_2, 1.1.2.1]$.

In this method, each query keyword is assigned to a B⁺tree index, and each entry of the B⁺tree stores all the leaf nodes that contain the keyword, together with their Dewey-order labels. For example, the XML tree of Figure 1 can be transformed into the B⁺tree shown in Figure 2. In the query phase, assume a query with keywords (*Hamlet*, *King*). We scan the keyword B⁺tree for each query keyword to find the corresponding leaf nodes. In this example, the corresponding node for keyword *Hamlet* is $[a_1, 1.1.1.1]$, and those for keyword *King* are $[a_2, 1.1.2.1]$, $[a_4, 1.2.1.1]$ and $[a_6, 1.2.3.1]$. Therefore, the LCA for the two query keywords is the node $[s_1, 1.1]$, because of the common label between node a_1 and a_2 . This method is effective for finding the LCAs, but it results in expensive query cost, as we will discuss in Section 5.2.

4.2 Superimposed Code-Based Method

Signature file partitioning techniques based upon superimposed codes are widely applied in such research areas as information retrieval and data security. Assume the size of the signature file F is S ; according to superimposed coding, each query keyword yields a word signature, i.e., a bit sequence of size S . These bit sequences are OR-ed together to form the signature file F . To create a word signature, each word is hashed to m bit positions in the range $1 - S$. The corresponding bits are set to “1”, while all the other bits are set to “0”. For example, consider the two files F_1 and F_2 of Table 1. The signature of each file can be generated by OR-ing the word signatures of all the keywords.

Given a query signature Q , for the signature S_i of file F_i , if $Q \wedge S_i = Q$, F_i is a candidate of the query result, which is called a *drop*. However, some drops

Table 1. Examples of file signatures

F_1		F_2	
Keyword	Signature	Keyword	Signature
Lear	1000001	Hamlet	0100001
King	0100010	King	0100010
Duke	0101000	Mother	0100100
Brother	1100000	Brother	1100000
File signature	1101011	File signature	1100111

Table 2. Examples of drops

Query keywords	Query signature	F_1	F_2
King, brother	1100010	actual drop	actual drop
King, mother	0100110	no match	actual drop
Lear, King	1100011	actual drop	false drop

actually do not correspond to all the query keywords. These drops are called *false drops*, while the drops that actually satisfy the query predicate are called *actual drops*. Table 2 shows example drops for different queries.

In this section, we propose an LCA detection method based on superimposed codes. Assume the set of keywords of an XML document is K and the keywords in the leaf node KN_i are $k_j \in K (1 \leq j \leq |KN_i|)$. The superimposed code S_i for the leaf node KN_i can then be calculated by OR-ing the signatures of the keywords $S(k_j)$:

$$S_i = S(k_1) \vee S(k_2) \vee \dots \vee S(k_{|KN_i|}). \quad (19)$$

Next, the superimposed code S_{pi} of the parent node PN_i can be computed by:

$$S_{pi} = S_{c1} \vee S_{c2} \vee \dots \vee S_{cm}, \quad (20)$$

where $S_{c1}, S_{c2}, \dots, S_{cm}$ denotes the signatures of the child nodes $CN_1, CN_2, \dots, CN_m$ of PN_i .

In the query, the query signature Q is determined by OR-ing the signature of each query keyword. Then, we investigate whether the query signature and each node signature NS_i satisfy the condition:

$$Q \wedge NS_i = Q. \quad (21)$$

All the nodes satisfying the above condition are candidate LCAs that may contain all the query keywords. However, there may be some false drops in the candidate LCAs.

4.3 Combined Method

The B⁺tree with the Dewey-order label and superimposed code-based methods are effective in finding the LCAs. However, they result in expensive query costs. To reduce the query cost, in this section we propose an efficient method that combines B⁺trees with Dewey-order labels and superimposed codes.

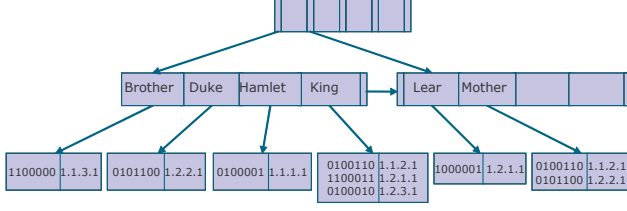


Fig. 3. An example keyword B+tree with superimposed codes

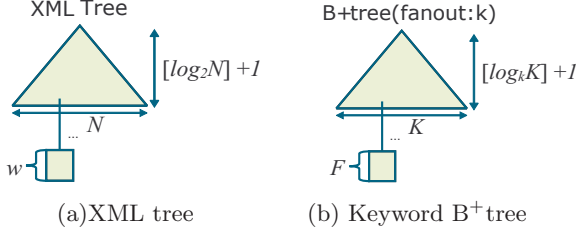


Fig. 4. Notations for query cost comparison

```

getLCA( $w_1, \dots, w_k$ ) {
     $Q = H(w_1) \vee \dots \vee H(w_k)$ ;
    For ( $i = 0$  to  $k$ ) {
         $NS_i = \text{getNodeSignature}(w_i)$ ; // Get the word
        signature of  $w_i$  from the keyword B+tree
         $L_i = \text{getResultSI}(Q, NS_i)$ ;
    }
    Return getTrueDropLCA( $L_1, \dots, L_k$ );
}

getResultSI( $Q, NS$ ) {
     $L = \text{Null}$ ;
    For ( $i = 0$  to  $|NS|$ ) {
        If ( $Q \wedge NS(i) = Q$ ) {
             $L.add(NS(i))$ ;
        }
    }
    Return L;
}

getTrueDropLCA( $L_1, \dots, L_k$ ) {
     $N.addAll(L_1)$ ;
    For ( $i = 1$  to  $|L_1|$ ) {
        For ( $j = 1$  to  $k$ ) {
            For ( $s = 1$  to  $|L_j|$ ) {
                If ( $L_1(i) = L_j(s)$ ) {
                    Break;
                }
                If ( $j = |L_i|$ ) {
                     $N.remove(L_1(i))$ ;
                }
            }
        }
    }
    Return N;
}

```

Fig. 5. False drop resolution algorithm

```

getLowestGDMCT( $l_1, \dots, l_n, KL_1, \dots, KL_k$ ) {
    Assume L represents the list of ( $l_1, \dots, l_n$ ) and G
    represents the list of GDMCT
     $L = \text{getSmallestLCA}(l_1, \dots, l_n, KL_1, \dots, KL_k)$ ;
    For ( $i = 0$  to  $n$ ) {
         $G = \text{getGDMCT}(l_1, \dots, l_n, KL_1, \dots, KL_k, i)$ ;
        Return ( $L, G$ );
    }
}

getSmallestLCA( $l_1, \dots, l_n, KL_1, \dots, KL_k$ ) {
     $L = \text{Null}$ ;
    For ( $i = 0$  to  $n$ ) {
        For ( $j = 0$  to  $k$ ) {
            If ( $i \neq j$ ) {
                If ( $l_j.\text{substring}(l_i)$ ) {
                    Break;
                }
                If ( $j = n$ ) {
                     $L.add(l_i)$ ;
                }
            }
        }
    }
    Return G;
}

getGDMCT( $KL_1, \dots, KL_k, n$ ) {
    For ( $i = 0$  to  $k$ ) {
        For ( $j = 0$  to  $|KL_i|$ ) {
            If ( $KL_i(j).\text{substring}(n)$ ) {
                 $G \leftarrow KL(j)$ ;
            }
        }
    }
    Return G;
}

```

Fig. 6. Lowest GDMCT detection algorithm

In the combined method, each node of the XML tree is assigned a superimposed code as in the original superimposed code-based method, and then both the superimposed code and the Dewey-order labels of nodes are stored in a key-

word B^+ tree, as shown in Figure 3. Assume a query with keywords $k_1, \dots, k_m$. The query signature Q is determined by OR-ing the signature of each keyword $S_i, 1 \leq i \leq m$, namely, $Q = S_1 \vee S_2 \vee \dots \vee S_m$. The candidate LCAs can be determined by scanning the B^+ tree with any keyword $k_i, 1 \leq i \leq m$ of the query, if the query signature Q and the node signature NS corresponding to the keyword satisfy the condition $Q \wedge NS = Q$. However, the candidate results may still have false drops.

We present an effective resolution for the false drop problem. For any keyword $k_i, 1 \leq i \leq m$ in the query, it is evident that the node with signature NS in the B^+ tree satisfying $Q \wedge NS = Q$ must contain the keyword k_i itself. For $1 \leq i \leq m$, we can obtain the node sets $N_1, N_2, \dots, N_m$ for each query keyword $k_i, 1 \leq i \leq m$ by using the condition $Q \wedge NS = Q$. Each node set $N_i, 1 \leq i \leq m$ must contain the corresponding keyword $k_i, 1 \leq i \leq m$. Assume the Dewey-order label sets of $N_i, 1 \leq i \leq m$ are $L_i, 1 \leq i \leq m$. The nodes that have the common label among $L_i, 1 \leq i \leq m$ are the LCAs that must contain all the query keywords. The algorithm for false drop resolution is shown in Figure 5.

5 Extracting Lowest GDMCTs

5.1 KBDLM and KBSIM

In this section, we present two methods for computing the lowest GDMCTs based on the keyword B^+ tree with Dewey-order labels method and the combined method, which are called the keyword B^+ tree with Dewey-order label method (KBDLM) and the keyword B^+ tree with superimposed code method (KBSIM), respectively. In the KBDLM and KBSIM, the LCAs are first determined by the keyword B^+ tree with Dewey-order labels or the superimposed codes. Then, the smallest LCAs, i.e., the root of the lowest GDMCTs, can be found by comparing the Dewey-order labels among the LCAs. The length of each edge can be computed by comparing the length between the label of the root and that of each leaf node. The lowest GDMCT detection algorithm is illustrated by Figure 6.

5.2 Query Cost Comparison

In this section, we evaluate the query cost for the proposed methods, KBDLM and KBSIM, and the SA algorithm. Firstly, we give some notation for query cost comparison shown in Table 3; some parameters are illustrated in Figure 4.

Query Cost of the SA Algorithm. According to reference [10], the query cost of the SA algorithm is:

$$O(\log_2 N * F^{2m}) = O(\log N * N^{2m}). \quad (22)$$

Query Cost of the KBDLM. In the KBDLM, the height of the keyword B^+ tree is $\lceil \log_k K \rceil + 1$, as shown in Figure 4 (b). In the worst case, with one query keyword, it is necessary to scan all the entries of the B^+ tree. The number

Table 3. Notation for query cost comparison

notation	description	notation	description
m	number of query keywords	K	number of keyword
k	fanout (node entries) of B^+ tree	N	number of leaf nodes in the XML tree
w	mean number of keywords in each node	F	mean number of nodes for each keyword
b	comparison cost for one bit	B	comparison cost for one string
r	mean number of strings of the labels	l	bit length of the signature
n	mean string length of the query keywords	d	mean length of the Dewey-order labels

of required comparisons for finding the nodes containing the keyword is $m \times k \times ([\log_k K] + 1)$. Therefore, the cost of finding the nodes containing all the m keywords is:

$$m \times k \times ([\log_k K] + 1) \times n \times B. \quad (23)$$

The cost for finding the LCAs by comparing their Dewey-order labels is:

$$F^{m-1} \times d \times B. \quad (24)$$

In the worst case, the number of parents of the lowest GDMCTs is N , so the cost for computing the lowest GDMCT is $N \times m \times F$. Therefore, the total query cost of KBDLM is the sum of the above costs:

$$m \times k \times ([\log_k K] + 1) \times n \times B + F^{m-1} \times d \times B + N \times m \times F. \quad (25)$$

F can be represented by N and K , because $F = \frac{wN}{K}$. Therefore, the query cost of the KBDLM can be expressed by the following equation in terms of the order of N :

$$O\left(\left(\frac{wN}{K}\right)^{m-1} + \left(\frac{wN^2}{K}\right)\right) = O(N^{m-1} + N^2). \quad (26)$$

When $m \leq 3$, the cost is $O(N^2)$; when $m \geq 4$, it is $O(N^{m-1})$.

Query Cost of the KBSIM. In the KBSIM, the cost for finding the nodes containing the keywords is the same as for the KBDLM. Then, the cost for detecting the LCAs of the nodes containing the keywords is $b \times l \times (F - 1) \times m$. Next, in the worst case, the cost of finding the false drops in the detected LCAs is $(m - 1)N^2$. Finally, the cost for computing the lowest GDMCTs is the same as the KBDLM, $N \times m \times F$. Therefore, the total query cost of the KBSIM can be calculated by:

$$m \times k \times ([\log_k K] + 1) \times n \times B + b \times l \times (F - 1) \times m + (m - 1)N^2 + N \times m \times F. \quad (27)$$

The total cost of the KBSIM is $O(N^2)$ by substituting $F = \frac{wN}{K}$ in the above equation. Therefore, when $m > 2$, the proposed combined method KBSIM has lower costs than either the KBDLM or SA algorithms.

Table 4. Experimental environment

CPU	AMD Opteron 2.2 GHz
Memory	6 GB
OS	Linux version 2.6.9
Database	PostgreSQL 8.1.3
Hard Disk	DRAILD
Java	1.5.0_07

Table 5. False drop rate of detected LCAs

	1 MB, 2 keywords	5 MB, 2 keywords
Original method	$\mathcal{R} = 7.6\%$	$\mathcal{R} = 10.3\%$
Combination method	$\mathcal{R} = 0\%$	$\mathcal{R} = 0\%$

6 Experimental Evaluation

We conducted experiments to evaluate the benefits of the proposed combined method, KBSIM. Firstly, we show that the KBSIM can completely solve the false drop problem in contrast to the original superimposed code-based method. Then, we compare the execution time for keyword queries using the KBSIM with that using the KBDLM and SA algorithms with different numbers of query keywords and different sizes of XML documents.

The experiments were performed in the environment shown in Table 4. The XML data collections used in the experiments were generated by *xmlgen* of the XMark benchmark [16]. Three kinds of data were generated by using different scaling factors of 0.01, 0.05 and 0.1, respectively. The sizes of the generated XML data are about 1 MB, 5 MB and 10 MB, respectively. The experiments were performed using sets of keywords having different frequencies introduced in [10]. Namely, *low*, corresponding to keywords with frequencies between 1 and 10, *medium*, corresponding to keywords with frequencies 11–200, and *high*, corresponding to keywords with frequencies greater than 200.

6.1 Evaluating False Drop Resolution

We applied the original superimposed code-based method and the combined method with false drop resolution to detect the LCAs using the XMark data of 1 MB and 5 MB with two query keywords. Table 5 shows the false drop rates $\mathcal{R}$ of the detected LCAs. We can see that the proposed combined method can completely solve the false drop problem in LCA detection caused by the original superimposed code-based method.

6.2 Evaluating Query Performance

To evaluate the query performance of the proposed algorithms, we firstly performed experiments to observe and compare the query time with different numbers of query keywords using the KBSIM, KBDLM and SA algorithms. The

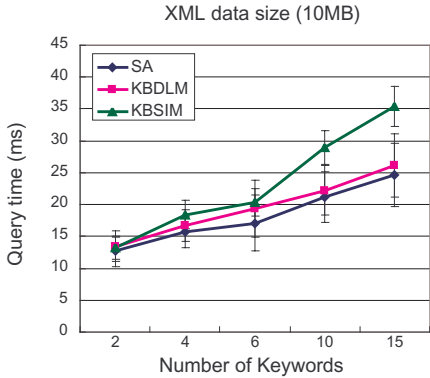


Fig. 7. Query time for keywords with low frequency

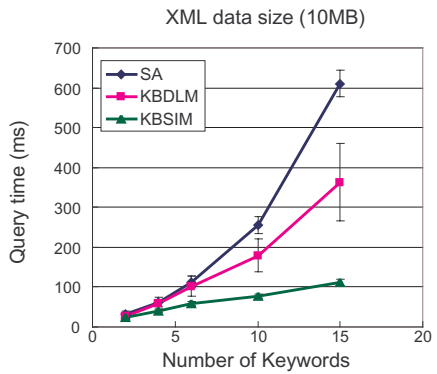


Fig. 8. Query time for keywords with medium frequency

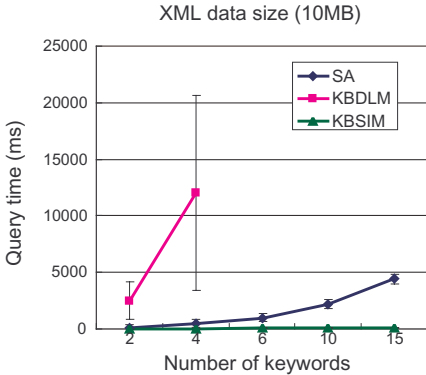


Fig. 9. Query time for keywords with high frequency

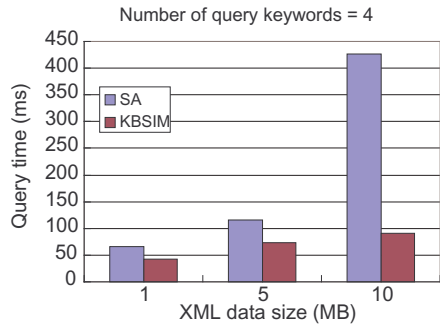


Fig. 10. Query time for different sizes of XML data

size of the XMark data used in the experiments was 10 MB, and the keyword frequencies ranged from *low* to *high*. Figure 7 presents the performance of each method as the number of keywords increases for keywords at low frequencies. It shows that the SA performs slightly better than both proposed algorithms for the low-frequency keywords. Figure 8 shows the query time as the number of keywords with medium frequency increases. This figure indicates that both the proposed algorithms are superior to the SA algorithm, and the KBSIM performs better than the KBDLM at these keyword frequencies. Figure 9 presents the results of query time as the number of keywords at high frequencies increases. It is evident that KBSIM is overwhelmingly superior to the other two methods.

However, the query time using the KBDLM increases extremely quickly when the number of high-frequency keywords is more than four².

We next conducted experiments to observe the performance of the KBSIM and SA algorithms with different sizes of XML documents. We measured the query times for XMark data of 1 MB, 5 MB and 10 MB with four high-frequency query keywords. Figure 10 represents the query time of the two methods for each XML document. This figure shows that as the XML document becomes larger, the proposed KBSIM performs more efficiently than the SA algorithm.

7 Conclusion and Future Work

With the exponential increase in the amount of XML data on the Internet, information retrieval techniques on tree-structured XML documents such as keyword search become important. In keyword searches over XML documents, the search results are often represented by MCTs rooted at the LCAs of the nodes containing all the query keywords. In this paper, we first considered two straightforward LCA detection methods: keyword B⁺ trees with Dewey-order labels and superimposed code-based indexing methods. Then, we proposed a method for efficiently detecting the LCAs, which combines the two straightforward indexing methods. We also presented an effective resolution for the false drop problem caused by the superimposed codes. Finally, we applied the proposed LCA detection method to generate the lowest GDMCTs over XML documents. We also conducted detailed experiments to evaluate the benefits of our proposed algorithms and to show that the proposed combined method can completely solve the false drop problem and outperform the previously known stack-based algorithm in extracting the lowest GDMCTs.

In the future, we plan to consider more sophisticated keyword searches over XML documents such as the user context-based keyword search. We will also work on the issue of search result ranking so that we can provide more effective search results to the users.

Acknowledgment

This work was partially supported by the Grant-in-Aid for Scientific Research of MEXT Japan #16016232, by CREST of JST (Japan Science and Technology Agency) and by the TokyoTech 21COE Program “Framework for Systematization and Application of Large-Scale Knowledge Resources”.

References

1. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: A System for Keyword-Based Search over Relational Databases. In: ICDE, pp. 5–16 (2002)
2. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: Enabling Keyword Search over Relational Databases. In: SIGMOD, p. 627 (2002)

² The query time using the KBDLM for more than four keywords is not shown in Figure 9.

3. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: On Finding Lowest Common Ancestors in Trees. In: STOC, pp. 253–265 (1973)
4. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword Searching and Browsing in Databases using BANKS. In: ICDE, pp. 431–440 (2002)
5. Clementi, A.E.F., Monti, A., Silvestri, R.: Selective Families, Superimposed Codes, and Broadcasting on Unknown Radio Networks. In: SODA, pp. 709–718 (2001)
6. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSearch: A Semantic Search Engine for XML. In: VLDB, pp. 45–56 (2003)
7. Dyer, M., Fenner, T., Frieze, A., Thomason, A.: On Key Storage in Secure Networks. *J. of Cryptology* 8(4), 189–200 (1995)
8. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRank: Ranked Keyword Search over XML Documents. In: SIGMOD, pp. 16–27 (2003)
9. Harel, D., Tarjan, R.E.: Fast Algorithms for Finding Nearest Common Ancestors. *SIAM J. Comput.* 13(2), 338–355 (1984)
10. Hristidis, V., Koudas, N.: Keyword Proximity Search in XML Trees. *IEEE TKDE* 18(4), 525–539 (2006)
11. Hristidis, V., Papakonstantinou, Y.: DISCOVER: Keyword Search in Relational Databases. In: VLDB, pp. 670–681 (2002)
12. Kaae, R., Nguyen, T.-D., Nørgaard, D., Schmidt, A.: Kalchas: A Dynamic XML Search Engine. In: CIKM, pp. 541–548 (2005)
13. Kautz, W.H., Singleton, R.C.: Nonrandom Binary Superimposed Codes. *IEICE Trans. Inform. Theory* 10(4), 363–377 (1964)
14. Li, Y., Yu, C., Jagadish, H.V.: Schema-Free XQuery. In: VLDB, pp. 72–83 (2004)
15. Nykänen, M., Ukkonen, E.: Finding Lowest Common Ancestors in Arbitrarily Directed Trees. *Inf. Process. Lett.* 50(6), 307–310 (1994)
16. XML Benchmark Project, <http://www.xml-benchmark.org>
17. Schieber, B., Vishkin, U.: On Finding Lowest Common Ancestors: Simplification and Parallelization. *SIAM J. Comput.* 17(6), 1253–1262 (1988)
18. Schmidt, A., Kersten, M.L., Windhouwer, M.: Querying XML Documents Made Easy: Nearest Concept Queries. In: ICDE, pp. 321–329 (2001)
19. Stinson, W.D.R., van Trung, T., Wei, R.: Secure Frameproof Codes, Key Distribution Patterns, Group Testing Algorithms and Related Structures. *J. of Statistical Planning and Inference* 86, 595–617 (2000)
20. Swiss-Prot, <http://www.ebi.ac.uk/swissprot/>
21. Tarjan, R.E.: Applications of Path Compression on Balanced Trees. *J. ACM* 26(4), 690–715 (1979)
22. TrEMBL, <http://www.ebi.ac.uk/trembl/>
23. The Free Encyclopedia: Wikipedia, <http://www.wikipedia.org/>
24. XML Version of DBLP, <http://dblp.uni-trier.de/xml/>
25. Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: SIGMOD, pp. 537–538 (2005)

Fast Matching of Twig Patterns

Jiang Li and Junhu Wang

School of Information and Communication Technology

Griffith University, Gold Coast, Australia

Jiang.Li@student.griffith.edu.au, J.Wang@griffith.edu.au

Abstract. Twig pattern matching plays a crucial role in XML data processing. Existing twig pattern matching algorithms can be classified into two-phase algorithms and one-phase algorithms. While the two-phase algorithms (e.g., **TwigStack**) suffer from expensive merging cost, the one-phase algorithms (e.g., **TwigList**, **Twig²Stack**, **HolisticTwigStack**) either lack efficient filtering of useless elements, or use over-complicated data structures. In this paper, we present two novel one-phase holistic twig matching algorithms, **TwigMix** and **TwigFast**, which combine the efficient selection of useful elements (introduced in **TwigStack**) with the simple lists for storing final solutions (introduced in **TwigList**). **TwigMix** simply introduces the element selection function of **TwigStack** into **TwigList** to avoid manipulation of useless elements in the stack and lists. **TwigFast** further improves this by introducing some pointers in the lists to completely avoid the use of stacks. Our experiments show **TwigMix** significantly and consistently outperforms **TwigList** and **HolisticTwigStack** (up to several times faster), and **TwigFast** is up to two times faster than **TwigMix**.

1 Introduction

The importance of fast processing of XML data is well known. *Twig pattern matching*, which is to find all matchings of a query tree pattern in an XML data tree, lies in the center of all XML processing languages. Therefore, finding efficient algorithms for twig pattern matching is an important research problem.

Over the last few years, many algorithms have been proposed to perform twig pattern matching. Al-Khalifa et al [3] gave an algorithm which breaks a query tree into binary (parent-child and ancestor-descendant) relationships, finds solutions for them, and merges such partial solutions to get the final solutions. One problem of this approach is the large number of partial solutions and hence the high cost in the merging phase. To overcome this problem, Bruno et al [4] proposed a holistic twig join algorithm called **TwigStack**, which breaks the query tree into root-to-leaf paths, finds individual root-to-path solutions, and merges these partial solutions to get the final result. One vivid feature of **TwigStack** is the efficient filtering of useless partial solutions through the use of function *getNext()*. It is shown that when there are only //-edges, every root-to-leaf path solution returned by the algorithm will contribute to some final solutions. Later on several improvements of **TwigStack** were made either to deal with /-edges

(e.g., `TwigStackList` [9]), or to make use of index structures (e.g., `TSGeneric+` [7], `iTwigJoin` [6]). Chen et al [5] observed that the holistic two-phase algorithms still suffer from high merging costs, and they proposed a one-phase algorithm, `Twig2Stack`, which avoids the merging phase by storing final solutions in hierarchical stacks. It is claimed that `Twig2Stack` outperforms `TwigStack`. Qin et al [10] proposed another one-phase algorithm, `TwigList`, which uses a much simpler data structure, a set of lists, to store the final solutions. Due to the simpler data structure and hence the reduction in random memory access, `TwigList` achieves better performance than `Twig2Stack` [10]. `Twig2Stack` and `TwigList` can avoid the high cost of the merging phase, but they lose an important ability of the holistic approach, which is efficiently locating twig occurrences and discarding useless elements. More recently Jiang et al [8] proposed a one-phase holistic twig matching algorithm called `HolisticTwigStack`, which maintains the overall solutions in linked stacks. However, a considerable amount of time is taken to maintain the linked stacks.

In this paper, we present two novel one-phase holistic twig matching algorithms, `TwigMix` and `TwigFast`, which combine the efficient selection of useful elements introduced in [4] with the simple data structure for storing final solutions introduced in [10]. `TwigMix` simply introduces the `getNext()` function of `TwigStack` into `TwigList` to avoid manipulation of useless elements in the stack and lists. `TwigFast` further improves this by introducing some pointers in the lists to completely avoid the use of stacks, based on the observation that the overhead of maintaining the pointers is generally negligible compared with the pushing/popping-up of elements into/from the stack. We conducted extensive experiments with both real and synthetic data. Our experiments show that (1) `TwigMix` significantly and consistently outperforms `TwigList` and `HolisticTwigStack` (up to several times faster), and `TwigFast` performs even better (up to two times faster) than `TwigMix`; (2) compared with `TwigList`, `TwigMix` saves an average of 75.93% of elements from being pushed into stack and an average of 70.19% of elements from being appended into the result lists. Since the result lists built by our algorithms are far shorter than those built by `TwigList`, our algorithms relieve the problem of memory consumption.

The rest of the paper is organized as follows. Section 2 provides background knowledge and recalls the major features of `TwigStack` and `TwigList`. `TwigMix` is presented in detail in Section 3. In Section 4, we present `TwigFast`. The experiment results are reported in Section 5. Finally, Section 6 concludes this paper.

2 Background

2.1 Terminology and Notations

An XML document is modeled as a node-labeled tree, referred to as the *data tree*. A *twig pattern* is also a node-labeled tree, but it has two types of edges: */*-edge and *//*-edges, which represent parent-child and ancestor-descendent relationships respectively. The *twig matching* problem is to find all occurrences of the twig

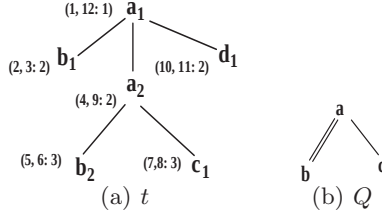


Fig. 1. Example data tree t and tree pattern Q

pattern in the data tree. Fig.1 shows a data tree t (where we use a_i to denote nodes labeled a , and so on) and a twig pattern Q . There is one occurrence of Q in t : (a_2, b_2, c_1) .

For data trees, we adopt the same region-based coding scheme used in **TwigStack**. Each node v is coded with a tuple of three values: $(v.start, v.end: v.level)$. Such a coding scheme has several useful properties: (1) ancestor-descendant and parent-child relationships can be identified in constant time: $\forall v_1, v_2 \in Nodes(t)$, v_1 is an ancestor of v_2 iff $v_1.start < v_2.start \leq v_2.end < v_1.end$, and v_1 is the parent of v_2 iff it is the ancestor of v_2 , and $v_2.level - v_1.level = 1$. (2) v_1, v_2 do not have ancestor-descendant relationship, and v_1 lies in a path to the left of the path where v_2 lies iff $v_1.end < v_2.start$ (See Fig. 1 (a)). These properties will be used extensively in our algorithms.

Below, we will use *elements* to refer to nodes in a data tree, and *nodes* to refer to nodes in a twig pattern. We will also use x -child (resp. x -descendant, x -element) to refer to a child (resp. descendant, element) labeled x . As in **TwigStack**, for each node n , there is a stream, T_n , consisting of all elements with the same label as n arranged in ascending order of their *start* values. Note that an element may appear in several streams if there are nodes with identical labels in Q . For each stream T_n , there exists a pointer PT_n pointing to the current element in T_n . The function $Advance(T_n)$ moves the pointer PT_n to the next element in T_n . The function $getElement(T_n)$ retrieves the current element of T_n . The function $isEnd(T_n)$ judges whether PT_n points to the position after the last element in T_n . In addition, for node n , the functions $isRoot(n)$ (resp. $isLeaf(n)$) checks whether node n is the root (resp. leaf), and $parent(n)$ (resp. $children(n)$) returns the parent (resp. set of children) of n .

2.2 TwigStack and TwigList

To facilitate our explanation, we briefly recall the major features of **TwigStack** and **TwigList** here.

As mentioned earlier, **TwigStack** uses a function $getNext(q)$ to efficiently filter useless elements. For self-containment, we copy the function into Algorithm 1. In the function, $nextL(T_n)$ and $nextR(T_n)$ return $getElement(T_n).start$ and $getElement(T_n).end$ respectively. The function has the following properties: if q is $root(Q)$ (the root of Q), then $getNext(q)$ always returns a node n that has

Algorithm 1. getNext(q) [4]

```

1: if (isLeaf(q)) return q
2: for  $q_i \in \text{children}(q)$  do
3:    $n_i = \text{getNext}(q_i)$ 
4:   if ( $n_i \neq q_i$ ) return  $n_i$ 
5:  $n_{min} = \text{minarg}_{n_i} \text{nextL}(T_{n_i})$ 
6:  $n_{max} = \text{maxarg}_{n_i} \text{nextL}(T_{n_i})$ 
7: while ( $\text{nextR}(T_q) < \text{nextL}(T_{n_{max}})$ ) do
8:   Advance( $T_q$ )
9: if ( $\text{nextL}(T_q) < \text{nextL}(T_{n_{min}})$ ) return q else return  $n_{min}$ 

```

a *minimal descendant extension* [4], i.e., (1) for each child n' of n , the current element of T_n has a descendant which is the current element of $T_{n'}$, and each child of n recursively has this property; (2) the current element of n has the minimum *start* value among all nodes that have property (1). The function also moves the pointer $PT(T_{n_i})$ when the current element in T_{n_i} no longer has descendants in T_{n_j} , for some of child n_j of n_i (lines 7,8).

TwigList is based on the following observation [10]: for each a -element v , its b -descendants can be arranged in a minimal interval, such that every b -descendant of v falls into this interval, and b -elements that are not descendants of v do not fall into the interval. As a consequence, we can use a pair of position values, v_{start_b} and v_{end_b} , to specify the interval for all b -descendants of v . For example, for the data tree shown in Fig.2 (a), all descendants of the a -nodes can be arranged in a list b_1, b_2, b_3, b_5, b_4 , and $a1_{start_b} = a1_{end_b} = 1$, $a2_{start_b} = 2$, $a2_{end_b} = 3$, $a3_{start_b} = 4$ and $a3_{end_b} = 5$ will tell us the b -descendants of each a -element. The data structure used in **TwigList** is thus a set of lists, one list, L_n , for each node n in Q . Each element v in L_n has pairs of start and end pointers pointing to the start and end positions of descendant intervals (one interval for each child of n). These lists are used to store the final solutions. For instance, for the data tree and query in fig.2 (a),(b), the lists built by **TwigList** are shown in fig.2 (d). In the figure, $a1$, $a2$ are not put into list L_a because they do not have c -descendants. The main algorithm of **TwigList** is a procedure to construct the lists, once this is done, it uses another procedure **TwigList-Enumerate** to efficiently enumerate the final solutions. To construct the lists, **TwigList** uses a stack, S . Elements are pushed into the stack in pre-order, and $top(S)$ is popped up when a non-descendant of $top(S)$ arrives, and it is then checked to see whether it should be appended to the corresponding list.

3 TwigMix: Introducing Efficient Element Filtering into TwigList

3.1 Overview of TwigMix

We explain the basic ideas used in **TwigMix** using the example in Fig. 2. **TwigMix** uses the same data structure as **TwigList**, but it introduces the *getNext()* function to avoid pushing useless elements into the stack S and appending useless

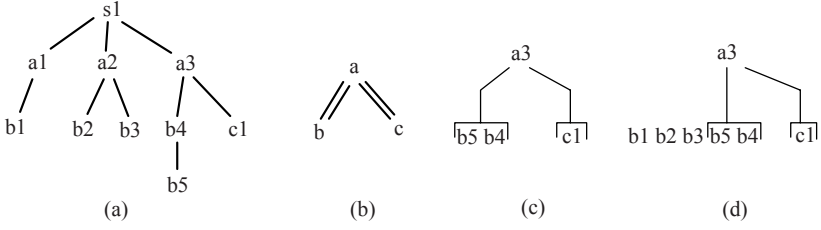


Fig. 2. An example to explain the basic ideas of **TwigMix**

elements into the lists. In Fig. 2, if we apply the **TwigList** algorithm, all of the elements will be pushed into S . When the elements are popped up from the stack, the algorithm will determine whether to append them to the result lists. For this example, $a1$ and $a2$ are not appended to the result lists because they can not find their c -descendants. However, $b1, b2$ and $b3$ are still appended to the result list although they do not contribute to the final solutions. Fig. 2 (d) shows the structure of the final lists constructed by **TwigList**. For **TwigMix**, due to the introduction of *getNext()*, $a1$ and $a2$ can be directly abandoned and will not be pushed into S . The elements $b1, b2, b3$ will not be pushed into S either because they can not find their ancestors in S . The final result lists are shown in Fig. 2 (c). Therefore, **TwigMix** does not waste time in pushing/popping-up $b1, b2$, and $b3$ into/from stack and appending them to result list L_b . It also saves memory because $b1, b2$ and $b3$ do not need to be stored in the lists. If the data tree is large, the savings of time and space will be quite significant (see Section 5 for examples).

3.2 TwigMix

TwigMix differs from **TwigList** in its way of constructing the final result lists. Once the lists are constructed, it uses the same procedure **TwigList-Enumerate** in [10] to enumerate all final solutions.

Our new algorithm for building the result lists, **TwigMix-Construct**, is shown in Algorithm 2. Like **TwigList**, we use a stack S to achieve bottom-up processing of elements. For each node $n_i \in Nodes(Q)$, we use a counter $n_i.counter$ to record the number of elements in stack S for that query node. In Algorithm 2, after initialization, the function *getNext(q)* is repeatedly called (lines 3,4) to get the query node which has a minimal descendant extension (see Section 2.2). The loop will stop until there are no elements not processed for any of the leaf nodes (see the *end(q)* function). Line 7 is particularly important. If the returned query node n_{act} is the root, its current element is directly pushed into the stack S . However, if it is not the root, the counter of $parent(n_{act})$ is checked to see whether any elements of $parent(n_{act})$ are in the stack. We push the current element of $T_{n_{act}}$ into S only when there are elements of $parent(n_{act})$ in S (this is why the elements $b1, b2$ and $b3$ in Fig.2 (a) are not pushed into stack). The counters are maintained at line 11 and line 20, when an element is pushed into

Algorithm 2. TwigMix-Construct(Q)

```

1: initialize stack  $S$  as empty;
2: initialize the list  $L_{n_j}$  as empty,  $n_j.counter$  as 0, for all nodes  $n_j \in Nodes(Q)$ ;
3: while  $\neg end(Q)$  do
4:    $n_{act} = getNext(root(Q))$ 
5:    $v_{act} = getElement(n_{act})$ 
6:    $toList(S, region(v_{act}))$  //  $region(v)$  denotes the interval  $(v.start, v.end)$ 
7:   if  $isRoot(n_{act})$  OR  $parent(n_{act}).counter > 0$  then
8:     for  $n_k \in children_{n_{act}}$  do
9:        $v_{act}.start_{n_k} = length(L_{n_k}) + 1$ 
10:     $push(S, v_{act})$ 
11:     $n_{act}.counter++$ 
12:     $Advance(T_{n_{act}})$ 
13:   $toList(S, (\infty, \infty))$ 
14: procedure  $END(q)$ 
15:   return  $\forall n_i \in Nodes(q) : isLeaf(n_i) \Rightarrow isEnd(T_{n_i})$ 
16: procedure  $toList(S, r)$ 
17:   while  $S \neq \emptyset$  AND  $r \notin reg(top(S))$  do
18:      $v_j = pop(S)$ 
19:     let  $v_j$ 's type be  $n_j$  // the type  $n_j$  is memorized when  $v_j$  is pushed into  $S$ 
20:      $n_j.counter--$ 
21:     for  $n_k \in children_{n_{act}}$  do
22:        $v_j.end_{n_k} = length(L_{n_k})$ 
23:     append  $v_j$  into list  $L_{n_j}$ 

```

or popped up from S . When an element is pushed into S , the start positions of its descendant intervals are set (lines 8,9). In the sub-procedure $toList(S, r)$, we check whether the current element in the node returned by $getNext(root(Q))$ is a descendant of $top(S)$, if not, we pop up $top(S)$, set the end positions of its descendant intervals, and append it directly to the corresponding list. Note that, unlike the procedure in **TwigList**, we do not need to check whether $top(S)$ can be appended to list because all elements pushed into the stack are guaranteed to appear in some final solution (provided Q has no $/$ -edges). At the end of the algorithm, we apply an infinite interval to $toList$ in order to pop up all elements from S .

Example 1. Consider the twig pattern and the data tree in Fig. 2. Initially, the current elements of the query nodes are $(a1, b1, c1)$. All the first three calls of $getNext(a)$ return node b . Because the counter of b 's parent a is 0, the elements $b1, b2, b3$ are not pushed into the stack S . The fourth call of $getNext(a)$ returns node a . Node a is the root of the query tree, so $a3$ is directly pushed into S and the start positions of its descendant intervals are recorded. The counter of node a increases by 1. The next two calls of $getNext(a)$ return node b . Because the counter of node a is 1, the elements $b5, b4$ are pushed into the S stack. Next, $getNext(a)$ returns node c . The coming of $c1$ results in $b5$ and $b4$ being popped up and appended to L_b . Finally, the range (∞, ∞) makes $c1$ and $a3$ pop up and they are appended to L_c and L_a respectively. When $a3$ is appended to L_a , the end positions of its descendant intervals are recorded.

3.3 Analysis of TwigMix

In this section, we show the correctness of **TwigMix** and analyze its time and space complexity. We prove the following lemma first.

Lemma 1. *Suppose Q has no $/$ -edges. **TwigMix** pushes an element into stack S iff the element contributes to some final solutions.*

Proof [sketch] (only if) If $getNext(root(Q))$ returns q_{act} , then q_{act} has a minimal descendant extension (see Section 2.2). Therefore, the current element v_{act} of $T_{q_{act}}$ (line 5) has a descendant in T_{n_i} for each child n_i of q_{act} . Line 7 and line 10 make sure that only if q_N is $root(Q)$ or S contains an element of type $parent(q_{act})$ do we push v_{act} into S . In both cases, v_{act} participates in at least one final solution, since we assume there are only $/$ -edges in Q .

(if) If an element v of type n participates in some final solution, $getNext(root(Q))$ will return n when the current element of T_n is v . If n is the root, v will be pushed into S directly. Otherwise, the stack S will contain at least one element of type $parent(n)$ when v is returned in line 5, because elements are pushed into stack in pre-order, and an element will be popped up from S after its descendants have been popped-up (line 17-18). Hence v will also be pushed into S . $\perp$

Theorem 1. *Given a twig pattern (that has $/$ -edges only) and an XML data tree, **TwigMix** correctly builds up the final result lists.*

Proof [sketch] We only need to show that (1) elements contributing to final solutions will be appended to the lists, and (2) for each element in the list, its descendant intervals are correctly set. (1) is true because of Lemma 1 and the fact that every element pushed in the stack S is appended in the result list. (2) is true because, for any element v_{act} satisfying the condition in line 7, it is pushed into S before any of its descendants are pushed into S . Therefore, the lists of the children nodes of n_{act} has no descendants of v_{act} before v_{act} is pushed into S at line 10. However, after v_{act} is pushed into S , the next element in T_{n_i} pushed into S for any child n_i of n must be a descendant of v_{act} . Therefore, line 9 correctly sets the start positions of the descendant intervals for v_{act} . Furthermore, v_j is popped up from S only when all of its descendants have been popped up and appended to lists. Therefore, line 21-22 correctly sets the end positions of v_j 's descendant intervals. $\perp$

Complexity analysis. Algorithm 2 scans each stream T_n from start to end once, through the functions $getNext()$ and $Advance(T_{n_{act}})$ at line 12. For each element in T_n it may push it into stack, pop it up from stack, append it to list, and set its start and end positions for its descendants. Suppose d is the maximum degree of nodes in Q . For each element appended to result lists, at most d intervals need to be recorded and recording an interval needs constant time. Pushing/popping-up an element into/from the stack S can be finished in constant time. Therefore, the worst-case time complexity is $O(d \cdot N)$ (N is the sum of the sizes of the input streams), which is linear in N . The worst-case space complexity is linear in the sum of the sizes of the occurrences of the twig pattern (the sum of the sizes of the final lists).

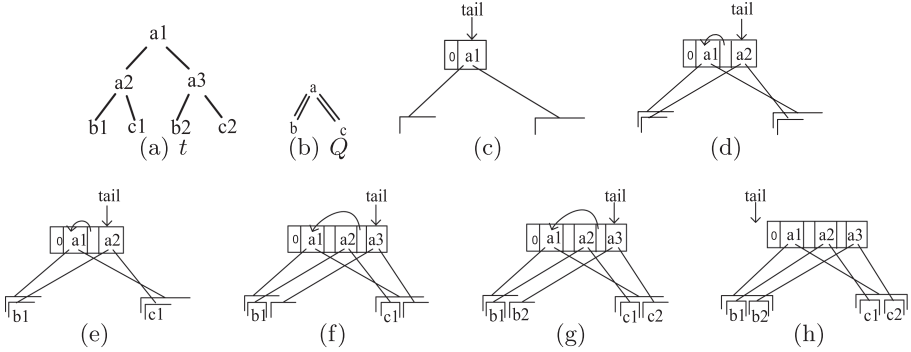


Fig. 3. An example to illustrate the basic ideas of TwigFast

Considerations of $/$ -edges. *getNext(q)* does not guarantee the returned node can be expanded to a solution when $/$ -edges exist. Therefore, Algorithm 2 does not guarantee all of the elements moved into the stack S and result lists will appear in final solutions when $/$ -edges exist. To make sure the final results enumerated are still correct, we need to modify the enumeration algorithm so that it checks the satisfaction of parent-child relationship, for $/$ -edges, when outputting final solutions.

To improve the efficiency of enumeration, one can use the strategy of adding sibling links as in [10]. This strategy can not prevent useless elements from being pushed into the stack S and appended into the result lists. To reduce the manipulation of useless elements, we can incorporate the *getNext(q)* function of algorithms that try to reduce the useless intermediate path solutions when $/$ -edges exist (e.g. TwigStackList [9], iTwigJoin [6], etc). However, these algorithms may result in the elements of the query nodes returned by *getNext(q)* are not in *pre-order*. Therefore, TwigMix-Construct needs to be adjusted.

4 TwigFast: Avoiding Manipulation of Elements in Stacks

4.1 Limitations of TwigMix

TwigMix integrates the holistic approach into TwigList, so only potentially useful elements are pushed into stack S and result lists. The time taken by pushing/popping-up elements into/from stack will become significant for large data trees. In order to get a glimpse of the number of elements that pass through S , we implemented TwigMix and did some experiments over the DBLP data set. The selected queries are listed in Table 1. As shown in the table, for all three queries, the number of elements pushed into S is very large. Therefore, if we can directly build up the final lists without using the stack, the performance can be significantly improved.

Algorithm 3. TwigFast(Q)

```

1: initialize the list  $L_{n_i}$  as empty, and set  $n_i.tail = 0$ , for all  $n_i \in Nodes(Q)$ ;
2: while  $\neg end(Q)$  do
3:    $n_{act} = getNext(root(Q))$ 
4:    $v_{act} = getElement(n_{act})$ 
5:   if  $\neg isRoot(n_{act})$  then
6:      $SetEndPointers(parent(n_{act}), v_{act}.start)$ 
7:   if  $isRoot(n_{act}) \vee parent(n_{act}).tail \neq 0$  then
8:     if  $\neg isLeaf(n_{act})$  then
9:        $SetEndPointers(n_{act}, v_{act}.start)$ 
10:      for  $n_k \in children(n_{act})$  do
11:         $v_{act}.start_{n_k} = length(L_{n_k}) + 1$ 
12:       $v_{act}.cancestor = n_{act}.tail$ 
13:       $n_{act}.tail = length(L_{n_{act}}) + 1$ 
14:      append  $v_{act}$  into list  $L_{n_{act}}$ 
15:       $Advance(T_{n_{act}})$ 
16:  $SetRestEndPointers(Q, \infty)$ 

17: procedure SETENDPOINTERS( $n, actL$ )
18:   while  $n.tail \neq 0$  do
19:      $v_n = element(n.tail)$ 
20:     if  $v_n.end < actL$  then
21:       for  $n_k \in children(n)$  do
22:          $v_n.end_{n_k} = length(L_{n_k})$ 
23:        $n.tail = v_n.cancestor$ 
24:     else
25:       break

26: procedure SETRESTENDPOINTERS( $n, actL$ )
27:   if  $\neg isLeaf(n)$  then
28:      $SetEndPointers(n, actL)$ 
29:   for  $q_i$  in  $children(n)$  do
30:      $SetRestEndPointers(n_i, actL)$ 

```

Table 1. Limitation of TwigMix

Query	Number of elements pushed into S
//dblp//inproceedings//title//author	915,856
//dblp//article//author//title//year	553,062
//dblp//inproceedings//cite//title//author	149,015

4.2 TwigFast

TwigFast uses a data structure that is essentially the same as that of **TwigMix**, but to avoid the use of stack S , it adds some pointers in the lists. More specifically, each element appended to the result list has a pointer, *cancestor*, that points to its closest ancestor in the same list. With these pointers, the elements on the same path can be linked together. For example, in Fig. 3(f), the element $a3$ has a pointer pointing to its closest ancestor $a1$. For each result list, a *tail* pointer is also maintained to point to the last element that still has potential descendants in the future. Together with the pointers that point to closest ancestors, we can easily maintain a list of elements which still have potential descendants, and these elements must be on the same path. For example, in

Fig.3(f), with the pointers, we can easily find $a3$ and $a1$ still have potential descendants, but $a2$ will not contribute to any new solutions in the future, so it is skipped by the pointer.

The purpose of the *cancezor* and *tail* pointers is to make it possible to correctly set descendant intervals for each element. When an element e is about to be appended to L_E , the start positions of intervals are determined (line 10 to 11). For each child C_i of query node E , the start position is equal to $length(L_{C_i}) + 1$. The end positions of an element can be determined when the element will not have any new descendants coming in the future (line 9). For each child C_i of query node E , the end position is equal to $length(L_{C_i})$. For example in Fig.3(f), the coming of $a3$ indicates $a2$ will not have any new descendants in the future, so the end positions of $a2$ are determined.

Example 2. Consider the data tree and twig pattern shown in Fig.3. The first call of *getNext()* returns a , with $a1$ being the current element (v_{act}) of T_a . Since a is the root of Q , and a is not the leaf, the procedure *SetEndPointers*($a, v_{act}.start$) is called but it does nothing since $a.tail = 0$. Now the start positions of $a1$'s descendant intervals are set to 1, and $a1.cancezor = 0$, $a.tail = 1$, and $a1$ is appended to list L_a , and current element of T_a is set to $a2$ (Fig.3 (c)). The second call of *getNext* also returns a , and *SetEndPointers*($a, a2.start$) is called. Since $a.tail \neq 0$, and $a1.end \geq a2.start$ (i.e., $a2$ is a descendant of $a1$), the procedure finishes with nothing done. Now lines 10 to 15 sets the start positions of $a2$'s descendant intervals as 1, and $a2.cancezor = 1$, $a.tail = 2$, appends $a2$ to L_a ((Fig.3 (d))), and advances T_a to $a3$. The next call of *getNext()* returns b , which is a leaf node. The current element of T_b is $b1$. Therefore, *SetEndPointers*($a, b1.start$) is called. Since $a.tail \neq 0$, and $a2.end > b1.start$, the procedure returns to line 7. Since $a.tail > 0$, $b1$ is appended to L_b (line 14), and $PT(T_b)$ points to $b2$. Similarly, the next call of *getNext()* returns c , and we append $c1$ to L_c , and make $PT(T_c)$ point to $c2$ (Fig.3 (e)). The next call of *getNext()* returns a with $v_{act} = a3$. *SetEndPointers*($a, a3.start$) is called. Since $a2.end < a3.start$, i.e., $a2$ no longer has b -descendants or c -descendants, we set the end positions of $a2$ as 1 and 1 for b and c . We then set the start positions of $a3$ as 2 and 2, $a3.cancezor = 1$ (pointing to $a1$), $a.tail = 3$, append $a3$ to L_a and advance T_a (Fig.3 (f)). The next two calls of *getNext()* return b and c respectively, so we append $b2$ and $c2$ to L_b and L_c respectively, and advance T_b and T_c (Fig.3 (f)). Now we use the infinite value to set the remaining end positions. That is, the end positions of $a3$ to 2. The final lists are shown in Fig.3 (h).

Correctness and complexity. Both the correctness of *TwigFast* and the linear time and space complexity of Algorithm 3 can be established, in a way similar to *TwigMix*.

Considerations of /-edges. For *TwigFast*, the strategy of adding sibling links [10] can also be applied. But one thing should be noted. *TwigFast* directly builds

up the final solutions into result lists, so ancestors are always appended to result lists before their descendants. Therefore, when we set end pointers for an element, if it can not find its children, it should be marked as useless. The enumeration algorithm will skip this element.

5 Experiments

In this section, we present the experiment results on the performance of **TwigMix** and **TwigFast** against **TwigList** [10] and **HolisticTwigStack** [8], with both real-world and synthetic data sets. **TwigList** is the most up-to-date one-phase twig pattern matching algorithm that applies the bottom-up approach. It is claimed to significantly outperform **Twig²Stack** [5] which, in turn, is claimed to be faster than **TwigStack**. **HolisticTwigStack** is also a one-phase holistic twig pattern matching algorithm, but the data structure used is complicated and expensive to maintain.

The algorithms are evaluated with the following metrics: (1) number of elements pushed into the S stack and result lists, (2) processing time.

5.1 Experiment Set-Up

The XML document parser we used is Libxml2 [2]. We implemented a generator in C to generate element encodings (*start, end, level*) for each element in an XML document. A simple XPath parser is also implemented, which generates the twig tree from an XPath expression.

We implemented **TwigMix**, **TwigFast**, **TwigList** and **HolisticTwigStack** in C++. All the experiments were performed on 1.6GHz Intel Centrino Duo processor with 1G RAM. The operating system is Windows XP. We used the following three data sets for evaluation:

TreeBank: We obtained TreeBank XML document from the University of Washington XML repository [1]. The data is deep and has many recursive elements with the same label. The maximal depth is 36 and there are more than 2.4 million elements.

DBLP: DBLP XML document is also obtained from the University of Washington XML repository [1]. This data set is wide and shallow. There are more than 3.3 million elements.

XMark: XMark is a synthetic data set, which is generated by the XML Benchmark Project [11]. We set the scaling factor as 2. The generated document is 226M with more than 3.33 million elements.

5.2 Experiment Results

We compared the algorithms **TwigMix**, **TwigFast** against **TwigList** and **HolisticTwigStack** with different twig pattern queries over the three data sets above. The queries are listed in Table 2.

Table 2. Queries over TreeBank, DBLP and XMark

Data set	Query	XPath expression
TreeBank	TQ1	//S//MD//ADJ
TreeBank	TQ2	//S//VP//IN//NP
TreeBank	TQ3	//S//VP//PP//NP//VBN//IN
TreeBank	TQ4	//S//VP//PP//NN//NP//CD//VBN//IN
TreeBank	TQ5	//S//VP//NP//VP//PP//IN//NP//VBN
DBLP	DQ1	//dblp//inproceedings//title//author
DBLP	DQ2	//dblp//article//author//title//year
DBLP	DQ3	//dblp//inproceedings//cite//title//author
DBLP	DQ4	//dblp//article//author//url//ee
DBLP	DQ5	//article//volume//cite//journal
XMark	XQ1	//item//location//description//keyword
XMark	XQ2	//people//person//address//zipcode//profile//education
XMark	XQ3	//item//location//mailbox//mail//emph//description//keyword
XMark	XQ4	//open_auction//parlist//bidder
XMark	XQ5	//people//person//address//zipcode//profile

Number of elements moved to S and result lists. We compared the number of elements pushed into stack S and appended to the result lists during processing. **TwigFast** and **HolisticTwigStack** do not push an element into the stack S , so we compared **TwigMix** with **TwigList**. The comparison results are presented in Table 3 and 4. Apart from the number of elements, we also calculated the reduction percentage made by **TwigMix**.

As shown in the tables, **TwigMix** reduces a large percentage (up to 99.9%) of elements moved to stack S and result lists. In some queries, the number of elements reduced is over 1 million. Even though one operation on stack or list is minor, such a large percentage of reduction is enough to significantly reduce the overall time. Additionally, the reduction is significant over all of the three data sets regardless of the structural characteristics of the data, which means the performance improvements brought by **TwigMix** are consistent.

The reduction of elements appended to result lists shows the advantage of **TwigMix** in *memory consumption*. Since the elements appended to result lists will not be released until the results enumeration finishes, they will waste memory space if they do not contribute to the final solutions. Therefore, the useless elements eliminated by **TwigMix** can significantly reduce the usage of memory.

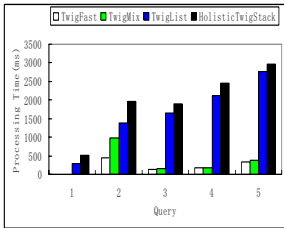
Processing time. The comparison of processing time is illustrated in Fig.4. As shown, both **TwigMix** and **TwigFast** significantly outperform **TwigList** and **HolisticTwigStack**. **TwigFast** shows better performance than **TwigMix** because it does not need to push elements into stack. This demonstrates that the overhead of maintaining the *cancestor* and *tail* pointers in **TwigFast** is well worthwhile. If we observe the figure together with Table 3 and Table 4, we can see that the processing time is closely related to the number of elements moved to S and result lists. In other words, the reduction of elements for processing directly brings the improvement of performance. For example, for query TQ4, the percentage of reduction is up to 99.1% such that the gap of processing time is huge. For query DQ1, against **TwigMix**, **TwigFast** saves 915,856 elements from being pushed into the stack, so the processing time nearly decreases by 2 times.

Table 3. Number of elements pushed into S

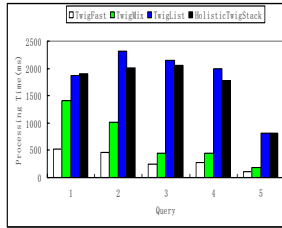
Query	TwigMix Elements	TwigList Elements	Reduction percentage	Useful Elements
TQ1	34	166,940	99.9%	34
TQ2	608,683	883,479	31.1%	608,683
TQ3	40,058	1,047,564	96.1%	40,058
TQ4	11,728	1,283,194	99.1%	11,728
TQ5	64,745	1,637,551	96.0%	64,745
DQ1	915,856	1,257,621	27.2%	915,856
DQ2	553,062	1,485,788	62.8%	553,062
DQ3	149,015	1,428,692	89.6%	149,015
DQ4	126,490	1,270,476	90.0%	126,490
DQ5	52,783	508,499	89.6%	52,783
XQ1	124,066	316,594	60.8%	124,066
XQ2	31,861	140,254	77.3%	31,861
XQ3	63,124	541,558	88.3%	63,124
XQ4	52,941	184,874	71.4%	52,941
XQ5	51,325	127,410	59.7%	51,325

Table 4. Number of elements appended to *result lists*

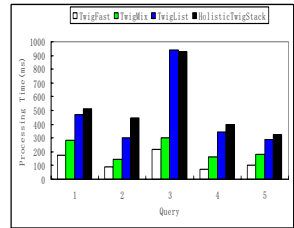
Query	TwigMix Elements	TwigList Elements	Reduction percentage	Useful Elements
TQ1	34	13,686	99.8%	34
TQ2	608,683	770,052	21.0%	608,683
TQ3	40,058	207,930	80.7%	40,058
TQ4	11,728	414,380	97.2%	11,728
TQ5	64,745	797,917	91.9%	64,745
DQ1	915,856	1,257,384	27.2%	915,856
DQ2	553,062	1,484,711	62.7%	553,062
DQ3	149,015	1,222,789	87.8%	149,015
DQ4	126,490	1,183,417	89.3%	126,490
DQ5	52,783	398,708	86.8%	52,783
XQ1	124,066	255,278	51.4%	124,066
XQ2	31,861	82,829	61.5%	31,861
XQ3	63,124	410,540	84.6%	63,124
XQ4	52,941	167,433	68.4%	52,941
XQ5	51,325	89,241	42.5%	51,325



(a) TreeBank



(b) DBLP



(c) XMark

Fig. 4. Processing Time(ms)

6 Conclusion

We presented two novel one-phase twig pattern matching algorithms that efficiently find twig pattern occurrences. **TwigMix** introduces holistic ideas into the

original bottom-up approach, such that the elements that do not contribute to final solutions are not moved into the stack and result lists. **TwigFast** directly builds up final solutions without pushing/popping-up elements into/from the stack. The better overall performance of our algorithms has been substantiated in our experiments. Since the result lists built by our algorithms are far shorter than those built by **TwigList**, our algorithms relieve the problem of memory consumption.

Acknowledgement. This work is partially supported by Griffith University New Researcher's Grant (GUNRG36621).

References

1. <http://www.cs.washington.edu/research/xmldatasets/>
2. <http://xmlsoft.org/>
3. Al-Khalifa, S., Jagadish, H.V., Patel, J.M., Wu, Y., Koudas, N., Srivastava, D.: Structural joins: A primitive for efficient XML query pattern matching. In: ICDE, p. 141 (2002)
4. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins: optimal XML pattern matching. In: SIGMOD Conference, pp. 310–321 (2002)
5. Chen, S., Li, H.-G., Tatemura, J., Hsiung, W.-P., Agrawal, D., Candan, K.S.: Twig²stack: Bottom-up processing of generalized-tree-pattern queries over XML documents. In: VLDB, pp. 283–294 (2006)
6. Chen, T., Lu, J., Ling, T.W.: On boosting holism in XML twig pattern matching using structural indexing techniques. In: SIGMOD Conference, pp. 455–466 (2005)
7. Jiang, H., Wang, W., Lu, H., Yu, J.X.: Holistic twig joins on indexed XML documents. In: VLDB, pp. 273–284 (2003)
8. Jiang, Z., Luo, C., Hou, W.-C., Zhu, Q., Che, D.: Efficient processing of XML twig pattern: A novel one-phase holistic solution. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 87–97. Springer, Heidelberg (2007)
9. Lu, J., Chen, T., Ling, T.W.: Efficient processing of XML twig patterns with parent child edges: a look-ahead approach. In: CIKM, pp. 533–542 (2004)
10. Qin, L., Yu, J.X., Ding, B.: TwigList: Make twig pattern matching fast. In: Kottagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 850–862. Springer, Heidelberg (2007)
11. Schmidt, A., Waas, F., Kersten, M., Florescu, D., Manolescu, I., Carey, M., Busse, R.: The XML benchmark project. Technical Report INS-R0103, CWI (April 2001)

XML Filtering Using Dynamic Hierarchical Clustering of User Profiles

Panagiotis Antonellis and Christos Makris

Computer Engineering and Informatics Department, University of Patras,
Rio 26500, Greece

adonel@ceid.upatras.gr, makri@ceid.upatras.gr

Abstract. Information filtering systems constitute a critical component in modern information seeking applications. As the number of users grows and the information available becomes even bigger it is crucial to employ scalable and efficient representation and filtering techniques. In this paper we propose an innovative XML filtering system that utilizes clustering of user profiles in order to reduce the filtering space and achieves sub-linear filtering time. The proposed system employs a unique sequence representation for user profiles and XML documents based on the depth-first traversal of the XML tree and an appropriate distance metric in order to compare and cluster the user profiles and filter the incoming XML documents. Experimental results depict that the proposed system outperforms the previous approaches in XML filtering and achieves sub-linear filtering time.

1 Introduction

Information filtering systems [1] are systems that provide two main services: document selection (i.e., determining which documents match which users) and document delivery (i.e., routing matching documents from data sources to users). In order to implement efficiently these services, information filtering systems rely upon representations of user profiles, that are generated either explicitly by asking the users to state their interests, or implicitly by mechanisms that track the user behaviour and use it as a guide to construct his/her profile. Initial attempts to construct such profiles typically used "bag of words" representations and keyword similarity techniques to represent user profiles and match them against new data items. These techniques, however, often suffer from limited ability to express user interests, being unable to fully capture the semantics of the user behaviour and user interests. As an attempt to face this lack of expressibility, there have appeared lately a number of systems that use XML representations for both documents and user profiles and that employ various filtering techniques to match the XML representations of user documents with the provided profiles. The process of filtering XML documents is the reverse of searching XML documents for specific structural and value information. An XML document filtering system stores user profiles along with additional information (e.g. personal information of the user, email address). When an XML document

arrives, the system filters it through the stored profiles to identify with which of them the document fits. After the filtering process has finished, the document can be sent to the corresponding users with matching profiles.

1.1 Existing Approaches

The existing XML filtering systems can be categorized as follows:

Automata-based Systems. The prominent examples of automata-based systems are XFilter [2] and YFilter[7]. Systems in this category incorporate Finite State Automata (FSA) to quickly match the document with the user profiles. In these systems, each data node causes a state transition in the underlying finite state automata representation of the filters.

Sequence-based Systems. Systems in this category represent both the user profiles and the XML documents as string sequences and then perform subsequence matching between the document and profile sequences. FiST [12] employs a novel holistic matching approach, that instead of breaking the twig pattern into separate root to leaf paths, transforms (through the use of the Prüfer sequence representation) the matching problem into a subsequence matching problem. In order to provide more efficient filtering, user profiles sequences are indexed using hash structures. XFIS [4] represents XML documents and user profiles using a novel sequence representation based on post order traversal and Prüfer sequences. XFIS supports on-line filtering of XML documents in only one pass, thus it is ideal for on-line applications and filtering systems.

Stack-based Systems. The representative system of this category is AFilter [5]. AFilter utilizes a stack structure while filtering the XML document against user profiles. Its novel filtering mechanism leverages both prefix and suffix commonalities across filter statements, avoids unnecessarily eager result/state enumerations (such as NFA enumerations of active states) and decouples the memory management task from result enumeration to ensure correct results even when the memory is tight.

Push Down Approaches. XPush [9] translates the collection of filter statements into a single deterministic pushdown automaton. The XPush machine uses a SAX parser that simulates a bottom up computation and hence doesn't require the main memory representation of the document. XSQ [13] utilizes a hierarchical arrangement of pushdown transducers augmented with buffers.

Suitable clustering algorithms for semistructured documents were extensively studied in [11]. XML document clustering was based in modeling the XML documents as trees, calculating the tree edit distance between them and applying a modified hierarchical clustering algorithm [8]. The tree edit distance is computed as the minimum-cost sequence of operations required to convert one given tree to another [10] [14]. In [6] the authors suggest the usage of tree structural summaries to improve the performance of the distance calculation and at the same time to maintain or even improve its quality. In [3] the authors introduce a

novel compact representation of XML documents based on edge summaries. The proposed structure is utilized along with a suitable distance metric to efficiently cluster homogeneous and heterogeneous XML documents.

1.2 Motivation and Contribution

Existing XML filtering approaches are not always effective in filtering XML documents against a rapidly growing number of stored user profiles for the following reasons:

- Usually, filtering systems cover a wide range of user interests and topics, thus each incoming XML document is relevant to a small portion of stored user profiles. However this fact is ignored by most filtering systems and the XML documents are checked against all user profiles.
- Systems considering similarities between stored user profiles, e.g. AFilter, utilize those similarities only to reduce extra checks, thus they keep checking every XML document against all user profiles.

In this research work we propose a filtering system that:

- Utilizes a unique sequence representation for both user profiles and XML documents based on the preorder traversal of XML tree.
- Measures similarity between two given user profiles or between an XML document and a user profile based on an innovative metric that utilizes a modification of the Levenshtein distance between the corresponding string representations.
- Creates an hierarchical structure of clusters using a hybrid hierarchical clustering algorithm based on the above mentioned metric.
- Applies a dynamic hierarchical filtering approach for each incoming XML document, based on the formed structure of clusters. The number of different levels of filtering depends on the number of previous matches in each level of clusters.

The rest of the paper is structured as follows. Section 2 introduces the utilized sequence representation and describes the distance metrics adopted; section 3 discusses analytically the clustering and filtering processes; section 4 discusses the experimental results and section 5 presents our conclusions.

2 Sequence Representation and Distance Metrics

2.1 Sequence Representation of XML Trees

In this work, we use a unique sequence representation of XML documents and user profiles, based on the preorder traversal of XML trees.

Every XML document can be easily modeled as an XML tree, where every enclosed element or attribute is modeled as a child in the XML tree. On the other

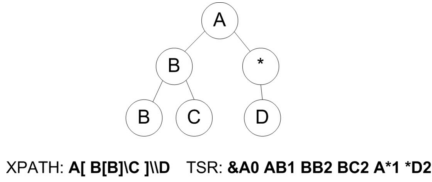


Fig. 1. Modeling a user profile as XML tree

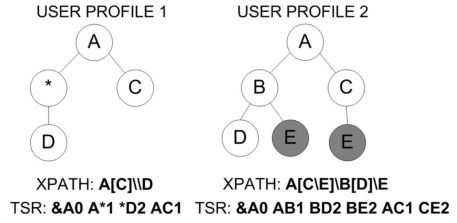


Fig. 2. Modified edit distance between two user profiles

hand, tree modeling of user profiles (expressed in XPath[19]) is not straightforward, as they may contain special relations (such as //, etc). In this paper we consider only parent/child (/) and ancestor/ descendant relations (//), which is the most used relation in user profiles. In order to model such a relation, we add an extra node in the XML tree, labeled with *. Figure 1 depicts an example of modeling a user profile (expressed in XPath) as an XML tree.

In order to construct the sequence representation of an XML tree, we need to replace each distinct tag label with a single unique char. For this reason, we utilize a dictionary structure that assigns each distinct tag with a single char label and keeps track of the correspondence between tag labels and char labels. In the rest of this paper, for simplicity reasons, we will refer to the XML nodes directly with their char labels.

Based on the above observations, we introduce a novel tree sequence representation (TSR) of XML trees, based on preorder traversal, with the property that every XML tree is represented by a unique sequence representation. Each node of the XML tree is encoded by the pair `<Parent><Node><Depth>`, where `<Parent>` represents the parent's char label, `<Node>` represents the current node's char label and `<Depth>` represents the current's node depth. If the current node is the root node, we replace `<Parent>` with `&`. For example, let us consider the XML tree in Figure 1. The encoding of the C node is `BC2`, where B is the char label of node's parent, C is the node's char label and 2 is its depth in the XML tree. The TSR of an XML tree is calculated by preorder traversing the XML tree and appending in each step the string encoding of every node reached. For example, the TSR of the XML tree presented in Figure 1 is `&A0 AB1 BB2 BC2 A*1 *D2`. The depth information is stored in order to avoid ambiguity in cases of nested nodes with the same label. For example, in Figure 1, if the TSR didn't store the depth for every node, the node D could be child of either the two B nodes.

2.2 Distance between User Profiles

In order to construct clusters of similar user profiles, we need to define a measure of the distance between two given user profiles. Previous works in this area propose the tree edit distance as a measure of the distance between any two labeled trees. The tree edit distance counts the cost of the total number of simple

edit operations required to transform one tree to another. Initially, each edit operation costs 1, but someone can assign different costs in every edit operation.

The following simple edit operations are allowed:

- *Delete*. Deletion of a single node.
- *Insert*. Insertion of a single node.
- *Replace*. Replacement of an existing node with another one.

The semantics of user profiles require some modification to the above measure. For this reason, we make the following assumptions:

- Delete operations are allowed only in leaf nodes or in ancestor/descendant cases and cost 1.
- Insert operations are allowed only in leaf nodes or in ancestor/descendant cases and cost 1.
- Replace operations are allowed everywhere, but cost as much as the minimum weight of the two corresponding nodes (replaced and replacement node). The intuition behind this is that the more descendants a tree node has, the more important is for the corresponding user profile semantics. However, if the replaced node has a $*$ label, then the replacement cost is 0.
- The weight of a node v , denoted as $w(v)$ is the total number of nodes in the subtree rooted at v .

Ancestor/descendant cases correspond to the presence of a $*$ -label. In those cases, it is allowed to delete/insert a parent/child node of a $*$ -labeled node. Figure 2 depicts an example of the modified edit distance between two user profiles. In this case, in order to transform User Profile 1 into User Profile 2, the following edit operations are required:

- Insertion of the E-labeled node under the $*$ -labeled node.
- Insertion of the E-labeled node under the C-labeled node.
- Replacement of the $*$ -labeled node with the B-labeled node.

Following the previously mentioned assumptions, each of the first two edit operations cost 1, while the third edit operation costs 0. Hence the modified edit distance between the two user profiles is 2.

In order to calculate our modified tree edit distance between two user profiles, we need a distance metric that reduces the problem of calculating tree edit operations into that of calculating the sequence edit distance between user profiles TSRs. For this reason we employ a modification of the Levenshtein distance between the TSRs of user profiles. The original Levenshtein algorithm [16] (also called Edit-Distance) calculates the least number of edit operations that are necessary to modify one string to obtain another string. The most common way of calculating this is by the dynamic programming approach. A tableau is initialized measuring in the (m, n) -cell the Levenshtein distance between the m -character prefix of one with the n -prefix of the other word [20]. The tableau can be filled from the upper left to the lower right corner. Each jump horizontally or vertically corresponds to an insert or a delete, respectively. The cost is normally

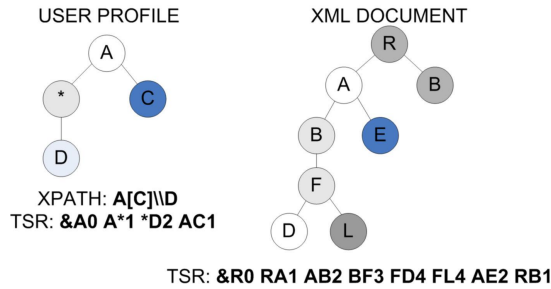


Fig. 3. Modified edit distance between a user profile and an XML document

set to 1 for each of the operations. The diagonal jump can cost either one, if the two characters in the row and column do not match, or 0, if they do. Each cell always minimizes the cost locally. This way the number in the lower right corner is the Levenshtein distance between the words.

However, in TSR, every node is represented as a pair of char labels and an integer representing its depth, thus we modify the Levenshtein distance to consider pair of chars instead of single chars. The depth information is used only in the case of nested nodes with the same tag, in order to distinguish between them. Moreover, in order for the user profiles semantics to be fulfilled, we apply the previously mentioned assumptions to the modified Levenshtein distance algorithm. Table 1 presents the modified Levenshtein algorithm applied for the two user profiles in Figure 2.

Table 1. Modified Levenshtein Algorithm

		&A0	AB1	BD2	BE2	AC1	CE2
	0	1	2	3	4	5	6
&A0	1	0	1	2	3	4	5
A*1	2	1	0	1	2	3	4
*D2	3	2	1	0	1	2	3
AC1	4	3	2	1	2	1	2

2.3 Distance between an XML Document and a User Profile

The distance between an XML document and a user profile is measured in a similar manner with the distance between two user profiles. This fact is critical, as the filtering algorithm should be able to compare the distance between two user profiles and the distance between an XML document and a user profile. The only differences are the following:

- Delete operations in the side of the XML document cost 0.
- Replace operations cost as much as the weight of the user profile’s corresponding node.

The above rules ensure that a user profile's distance from an XML document is 0 iff its tree representation is a subtree of the XML document's representation. Figure 3 depicts an example of the modified edit distance between a user profile and an XML document. In this case, in order to transform the XML document into the user profile, the following edit operations are required:

- Deletion of the R-labeled node.
- Deletion of the B-labeled node (under the R-labeled node).
- Deletion of the L-labeled node.
- Replacement of the E-labeled node with the C-labeled node.
- Replacement of the B-labeled node and the F-labeled node with the *-labeled node.

Following the assumptions made before, the first three edit operations cost 0. The fourth edit operation costs as much as the weight of the C-labeled node, e.g. 1. Finally the last edit operation costs 0. Hence the modified edit distance between the user profile and the XML document is 1.

The distance is calculated again utilizing a modified Levenshtein distance algorithm based on the previously presented assumptions.

One crucial property of the use of the two previously described metrics is expressed in the following lemma:

Lemma 1. *Given two user profiles P_1 , P_2 and an XML document D , suppose that $distance(D, P_2) = 0$. Then $distance(D, P_1) \leq distance(P_2, P_1)$*

Proof. Since $distance(D, P_2) = 0$, then a segment of the XML document matches exactly with P_2 . Let us denote by S that segment and S' the rest of the XML document (excluding S). Let us consider $distance(S', P_1)$. There are two cases:

- $distance(S', P_1) \leq distance(P_2, P_1)$
- $distance(S', P_1) > distance(P_2, P_1)$

In the first case, we can delete S from D (deletion costs 0), and thus:

$$distance(D, P_1) = distance(S', P_1) \Leftrightarrow distance(D, P_1) \leq distance(P_2, P_1)$$

In the second case, we can delete S' from D (deletion costs 0). Thus, $distance(D, P_1) = distance(S, P_1)$.

However, because S matches with P_2 , we have:

$$distance(S, P_1) = distance(P_2, P_1) \Leftrightarrow distance(D, P_1) = distance(P_2, P_1).$$

Thus in every case we have proved that: $distance(D, P_1) \leq distance(P_2, P_1)$. $\square$

The above lemma allows us to apply a clustering technique in order to reduce the filtering space and thus create an hierarchical filtering scheme as explained in the next sections. In particular, consider a cluster of user profiles, C , and its centroid profile P . The centroid profile is the profile that has the minimum average distance from the rest of the user profiles. In addition, consider that the most distant profile from the centroid is the profile O and its distance from P is d . Finally, consider an XML document D whose distance from P is r . If $r \geq d$, then based on Lemma 1, we can assume that there is no profile in the cluster

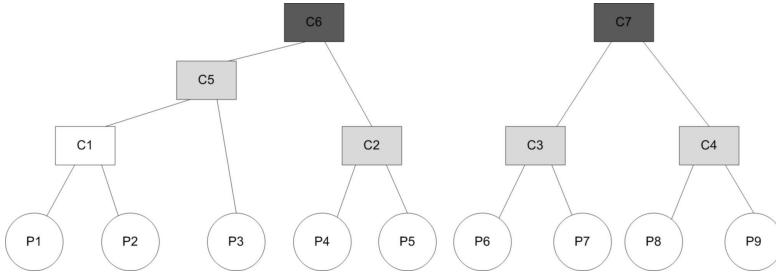


Fig. 4. Example of a cluster hierarchy forest

C whose distance from D is 0. If there was such a profile Z , then its distance from P should be greater than r , based on Lemma 1. However, the most distant profile's distance from P is $d \leq r$, thus there is no such a profile as Z in the cluster C .

3 Filtering System

Our filtering system consists of two subcomponents: *User profile clustering* and *Filtering algorithm*. The user profile clustering process is activated once, when the system is initialized. When an XML document arrives, the filtering algorithm is invoked to find those user profiles that match with the XML document. The filtering algorithm utilizes the hierarchical structure of clusters formed at the clustering phase in order to find those user profiles that match with each incoming XML document.

3.1 User Profile Clustering

Our XML filtering system utilizes a modified hierarchical clustering algorithm, in order to form a cluster hierarchy. The proposed clustering algorithm is a classical hierarchical clustering algorithm which utilizes the previous described distance metric between two user profiles. Our clustering algorithm works as follows:

At first, every user profile is considered as a single cluster. In every step, the algorithm finds the two closest clusters and merges them in one cluster. For every newly formed cluster, the algorithm calculates the cluster centroid, which is that user profile which minimizes the average distance with the rest user profiles in that cluster. In addition, the algorithm calculates the **max distance**, which is the distance between the cluster's centroid and the most distant user profile inside the cluster. The **max distance** will be utilized during the filtering process, described in Section 3.3. Finally, the clustering algorithm keeps track of the cluster hierarchy, thus every formed cluster points to the two clusters it was formed by. The clustering algorithm stops until only two top-level clusters have remained. The result of our clustering algorithm is a cluster hierarchy forest (with two root nodes) in which every node has exactly two children nodes (except

of the leaf nodes). Every node u of that tree stores pointers to every-leaf node (e.g. user profile) contained in the subtree rooted at node u , a pointer to its centroid profile and its **max distance**.

Figure 4 shows an example of a such a cluster hierarchy forest. The clusters C6 and C7 are the two root nodes of the forest. As it can be seen, every cluster in the forest has exactly two children, except of the low-level clusters (P1 through P9) which represent the stored user profiles. Every cluster in the forest stores pointers to its low-level user profiles, thus, for example, cluster C5 has pointers to user profiles P1, P2, and P3. Due to space limitations, the **max distance** and centroid profile for every cluster are not shown into the figure.

3.2 Filtering Algorithm

The filtering algorithm is used in order to filter an incoming XML document through the previously described cluster hierarchy forest. The result of the filtering process is a list of user profiles that match with the incoming XML document. The incoming XML document is then forwarded to the corresponding users of the matched profiles.

Before describing the filtering algorithm, we give two important definitions which will be used through out the filtering algorithm.

Definition 1. *A user profile p matches with an incoming XML document D iff $distance(p, D) = 0$.*

The distance between a user profile and an XML document is measured as described in Section 2.3, thus the above definition ensures that a user profile matches an XML document iff its corresponding tree representation is a subtree of the XML document's tree representation.

Definition 2. *An XML document D matches with a cluster of user profiles C iff $distance(D, c) \leq m$, where c is the cluster's centroid and m is the cluster's **max distance**.*

The above definition is based on Lemma 1. So, if $distance(D, c) \leq m$, then it is possible that the cluster C contains a user profile that matches with D . On the other hand, if $distance(D, c) > m$, it is not possible that the cluster C contains a user profile that matches with D . Thus, if an XML document D matches with a cluster C , then it should be filtered through all the user profiles contained in C . Otherwise, we can ignore the unmatched cluster and all its user profiles. This notion is exploited by our filtering algorithm in order to dramatically decrease the filtering space, thus achieving much better filtering time than the other filtering algorithms.

The proposed algorithm utilizes a list of active clusters, called **activeList**, which at any moment contains all the clusters that should be checked against the next incoming XML document. This list is updated after an XML document has been filtered as described later. In addition, the filtering algorithm adds a counter called **matchCnt** in every cluster of the hierarchy tree. This counter counts

how many XML documents have been matched with the corresponding cluster. Finally, the filtering algorithm initializes an extra global counter, called `totCnt`, that counts the total number of XML documents that have been filtered. The filtering process is as follows:

When first initialized, the `activeList` contains only the two top clusters in the hierarchy forest. As a result, the first incoming XML document will be checked against the two clusters in the `activeList`. At any step of the filtering process, every incoming XML document is checked only against the clusters of the `activeList`. For every matched cluster, the filtering algorithm increases its `matchCnt` and then filters the XML document against all the user profiles contained in that cluster, by simply calculating the distance between the XML document and every user profile as described in Section 2.3. Every profile that matches with the XML document is added to the output resultset of profiles. However, the process of filtering an XML document with all the user profiles within a cluster is the bottleneck of the filtering algorithm because it requires checking the XML document with all the user profiles. Based on this notion, we propose a dynamic filtering process which takes into consideration the number of matchings per cluster and updates accordingly the `activeList`. The intuition is that if a cluster has a lot of matchings, then the matched XML documents are always checked against all its user profiles, thus if we want to reduce that cost, we should split that cluster. In the same manner, if a cluster has very few matchings, then we can eliminate the cost of checking every incoming XML document with its centroid by merging that cluster with its sibling cluster. Thus, after an XML document has been filtered, the filtering algorithm checks the value of $\frac{\text{matchCnt}}{\text{totCnt}}$ of all the clusters contained in the `activeList` and compares them with two thresholds: `topThr` and `bottomThr`. If the $\frac{\text{matchCnt}}{\text{totCnt}}$ for a cluster C is greater than `topThr`, then we remove C from the `activeList` and insert into the `activeList` the two children of C . On the other hand, if the $\frac{\text{matchCnt}}{\text{totCnt}}$ for a cluster C is less than `bottomThr`, we remove C and its sibling from the `activeList` and insert into the `activeList` the parent cluster of C . Thus, in every step of the filtering process, we try to eliminate the cost of checking an XML document with the centroids of the clusters in the `activeList` and the cost of filtering an XML document with all the user profiles within a matched cluster.

For example, consider the cluster hierarchy tree presented in Figure 4. Initially, the `activeList` contains the clusters C6 and C7. Thus every incoming XML document is checked against those clusters and if it matches with one or both of them, it is filtered through the user profiles of the matched cluster(s). After a few XML documents have been filtered, suppose that the value of $\frac{\text{matchCnt}}{\text{totCnt}}$ for the cluster C6 has exceeded the `topThr`. In such a case, the filtering algorithm removes C6 from the `activeList` and inserts the clusters C5 and C2 into the `activeList`. Thus, every incoming XML document is now checked against clusters C5, C2 and C7.

The values of **topThr** and **bottomThr** vary between 0 and 1 and are not strictly defined and can be adjusted accordingly to the needs of every system. We propose the following indicative values: **topThr** = 0.3 and **bottomThr** = 0.05.

4 Experiments

We tested our filtering system against FiST[12], which is one of the state-of-the-art algorithms for filtering XML documents against twig pattern user profiles. We chose FiST because it supports twig pattern user profiles, unlike other systems (e.g. AFilter, XFilter) which support only linear path expressions. Our filtering system was implemented in Java using the freeware Eclipse IDE[18]. In order to obtain comparable and reliable results, we also implemented a FiST-like algorithm in Java using Eclipse.

In our experiments we used three different datasets: the DBLP dataset[15], the Shakespeare's plays dataset[17] and the Sigmod Record dataset[21]. For each of those datasets, we also generated a random number of user profiles with arbitrary depth and fan-out.

Our first experiment was to investigate the influence of the **topThr** threshold in the performance of our algorithm. For that purpose, we disabled the checking for the **bottomThr** threshold and we used our algorithm to filter 100 documents through 1000 user profiles. Both utilized documents and user profiles were arbitrarily selected from the 3 aforementioned datasets. The initial value of **topThr** was 0.1 and in each step of this experiment we increased **topThr** by 0.1 until it became 0.9. We measured the total filtering time of 100 documents required by our algorithm in each step and we present the results in Figure 5. As we can see, the total filtering time for the 100 XML documents decreases as the value of **topThr** increases, until **topThr** reaches 0.4. At that point, the filtering time has its global minimum value (approximately 115000ms). After that point, the filtering time starts to increase again as the value of **topThr** increases. This trend of the filtering time can be explained as follows: at first, when **topThr**

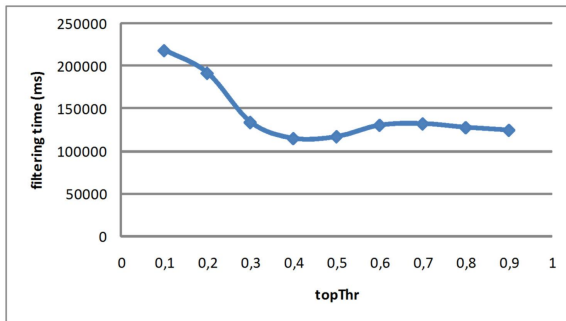


Fig. 5. Filtering time in relation with **topThr** threshold

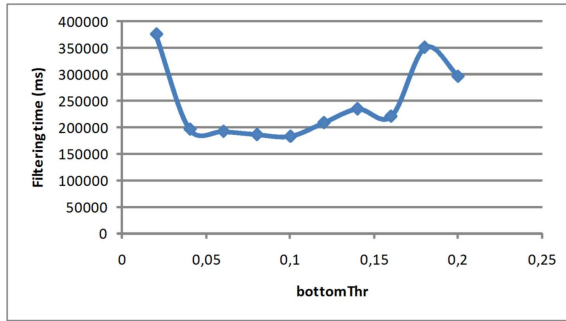


Fig. 6. Filtering time in relation with bottomThr threshold

has a low value (e.g 0.1, 0.2), the filtering algorithm acts aggressively and splits very often the clusters belonging to **activeList**, thus it moves deep in the cluster hierarchy. As a result, the size of the **activeList** becomes very large and each incoming document is always checked against a large number of low-level clusters. However, as the value of **topThr** increases (e.g 0.3, 0.4), the filtering algorithm becomes less aggressive and splits fewer clusters, resulting in a smaller **activeList**. Every incoming XML document is now checked against a moderate number of top-level clusters, thus the total filtering time is reduced and reaches its minimum value when **topThr** becomes 0.4. However, after that point and as the value of **topThr** continues to increase, the filtering algorithm acts more conservatively and rarely splits the clusters contained in **activeList**. Although, the size of **activeList** remains very small, the clusters contained in **activeList** are very large (as they are not splitted easily) and whenever an incoming XML document matches with a cluster in **activeList** it is filtered out through all the profiles belonging to the corresponding cluster, thus requiring more time to be filtered.

Our second experiment was to investigate the influence of the **bottomThr** threshold in the performance of our algorithm. For that purpose, we standardized **topThr** to 0.4, while in each step of that experiment we incremented **bottomThr** by 0.02 (starting from 0.02) until it reached 0.2 (half value of **topThr**). We measured the total filtering time of 100 documents required by our algorithm in each step and we present the results in Figure 6. As we can see, the filtering time behaves in a similar manner with the first experiment: it decreases until **bottomThr** becomes 0.1 and increases after that point. The explanation behind this is that low values of **bottomThr** result in a very conservative filtering algorithm which rarely merges two clusters of the **activeList**, thus the size of **activeList** never decreases even if some of its clusters are rarely matched with an incoming XML documents. However, as the value of **bottomThr** increases, the filtering algorithm starts to merge more easily rarely matched clusters, thus the **activeList** contains less but more popular clusters, thus the filtering time decreases. Further increase of **bottomThr** (> 0.1) results in an aggressive merging of clusters contained in **activeList**, thus the **activeList** contains only some

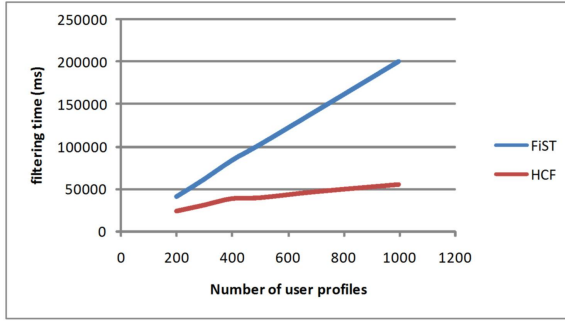


Fig. 7. Filtering time required for 200 documents by FiST and our algorithm (HCF)

few top-level clusters, which in turn results in a lot of cluster matchings for every incoming XML document. Those matchings increase the filtering time, as the XML document has to be filtered out through all the user profiles contained in every matched top-level cluster.

Our third experiment was to compare our proposed algorithm with FIST, one of the state-of-the-art XML filtering algorithms. During this experiment we measured the total time required by the two algorithms for filtering a set of 200 XML documents. We varied the number of stored user profiles between 200 and 1000 in order to investigate the relation between the number of user profiles and the required filtering time. The results of this experiment are presented in Figure 7.

According to Figure 7, our algorithm (referred as HCF) outperforms FIST in every step of that experiment and the difference in filtering time increases dramatically as the number of user profiles grows. In particular, in the case of 200 user profiles, our algorithm is 41% faster while in the case of 1000 user profiles, our algorithm is 72% faster. Another important notion is that FIST requires linear filtering time, while our algorithm requires sub-linear time. The effectiveness of our algorithm is due to the reduction of the filtering space achieved by employing the cluster hierarchy forest. As a result, the number of user profiles needed to be checked against every incoming XML document is very small related to the total number of stored user profiles.

5 Conclusions and Future Work

In this paper we have presented a new XML filtering system that uses clustering of user profiles in order to scale well as the number of user profiles grows. The proposed system utilizes a unique sequence representation for user profiles and XML documents, based on the preorder traversal. Based on this sequence representation, we proposed a modification of the Levenshtein distance metric in order to calculate the distance between two user profiles or between an XML document and a user profile. The proposed metric reduces the problem of calculating the tree edit distance into that of calculating the modified Levenshtein

distance between the sequence representations. Our system applies hierarchical user profile clustering in order to succeed sub-linear filtering time, based on the number of matchings per cluster. Our experimental results showed that the proposed system outperforms the previous algorithms in XML filtering and requires sub-linear time to filter the incoming XML documents.

As future work, we intend to compare our filtering algorithm with more approaches (such as AFilter, YFilter etc) as well as to utilize alternative clustering techniques such as k-Means; moreover, we aim to extend our filtering algorithm in order to additionally support value-predicate user profiles instead of only structural user profiles.

Acknowledgements

Panagiotis Antonellis' work was supported in part by the Hellenic State Scholarships Foundation (IKY).

References

1. Aguilera, M.K., Strom, R.E., Stunnen, D.C., AsHey, M., Chandra, T.D.: Matching Events in a Content-based Subscription System. In: PODC 1999, pp. 53–61 (1999)
2. Altinel, M., Franklin, M.I.J.: Efficient Filtering of XML Documents for Selective Dissemination of Information. In: VLDB 2000, pp. 53–64 (2000)
3. Antonellis, P., Makris, C., Tsirakis, N.: XEdge: Clustering Homogeneous and Heterogeneous XML Documents Using Edge Summaries. In: ACM SAC 2008 (to appear, 2008)
4. Antonellis, P., Makris, C.: XFIS: An XML Filtering System based on String Representation and Matching. *International Journal on Web Engineering and Technology (IJWET)* 4(1), 70–94 (2008)
5. Canadan, K., Hsiung, W., Chen, S., Tatemura, J., Agrawal, D.: AFilter: Adaptable XML Filtering with Prefix-Caching and Suffix-Clustering. In: VLDB 2006, pp. 559–570 (2006)
6. Dalamagas, T., Cheng, T., Winkel, K., Sellis, T.: Clustering XML documents using Structural Summaries. In: Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 547–556. Springer, Heidelberg (2004)
7. Diao, Y., Altinel, M., Franklin, M.: Path sharing and predicate evaluation for high-performance XML filtering. *TODS* 28(4), 467–516 (2003)
8. Francesca, F., Gordano, G., Ortale, R., Tagarelli, A.: Distance-based Clustering of XML Documents. In: MGTS 2003, pp. 75–78 (2003)
9. Gupta, A.K., Suci, D.: Stream processing of XPath queries with predicates. In: SIGMOD 2003, pp. 419–430 (2003)
10. Isert, C.: The editing distance between trees. Technical Report, Ferienakademie, for course 2: Bume: Algorithmik Und Kombinatorik, Italy (1999)
11. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
12. Kwon, J., Rao, P., Moon, B., Lee, S.: FiST: Scalable XML Document Filtering by Sequencing Twig Patterns. In: VLDB 2005, pp. 217–228 (2005)

13. Peng, F., Chawathe, S.: XSQ: A streaming XPath Queries. In: TODS 2005, pp. 577–623 (2005)
14. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing, 1245–1262 (1989)
15. <http://kdl.cs.umass.edu/data/dblp/dblp-info.html>
16. <http://www.levenshtein.net/>
17. <http://xml.coverpages.org/bosakShakespeare200.html>
18. <http://www.eclipse.org>
19. <http://www.w3.org/TR/xpath>
20. <http://www.levenshtein.net>
21. <http://www.dia.uniroma3.it/Araneus/Sigmod/Record/DTD/index.html>

Person Retrieval on XML Documents by Coreference Analysis Utilizing Structural Features

Yumi Yonei, Mizuho Iwaihara, and Masatoshi Yoshikawa

Department of Social Informatics, Graduate School of Informatics, Kyoto University
Yoshidahonmachi, Sakyo-ku, Kyoto, 606-8501 Japan

Abstract. Keyword retrieval of the present day exploits frequencies and positions of search keywords in target documents. As for retrieval by two or more keywords, semantic relation between keywords is important. For retrieving information about a person, it is common to search by a pair of keywords consisting of person's name and his/her attribute of the interest. By using dependency analysis and coreference analysis, correct occurrences of pairs of person and his/her attributes can be retrieved. However, existing natural language analysis does not consider the factor that logical structures of the documents strongly influence probabilistic patterns of coreference. In this paper, we propose a new way of person retrieval by computing a maximum entropy model from linguistic features and structural features, where structural features are learned from probabilistic distribution of coreference over XML document structures. Our method can utilize strong correlation between XML document structures and coreference, thus having superior accuracy than existing methods.

1 Introduction

Keyword retrieval of the present day exploits frequencies and positions of search keywords in target documents. As for retrieval by two or more keywords, semantic relation between keywords is important. For retrieving information about a person, it is common to search by a pair of keywords consisting of person's name and his/her attribute of the interest. For example, when we want to know George Walker Bush's birthplace, we can set a pair of search keywords; "George Walker Bush" and "birthplace". However if semantic relation between the keywords is not considered, documents that describe a different person's birthplace may be retrieved. In this paper, we aim at improving accuracy of person retrieval by considering coreference relation between keywords consisting of person's name and his/her attribute. Coreference analysis is aimed at resolving references of expressions such as "his birthplace", where the word "his" is called anaphor, and the problem is to find antecedent, which is the word that appears before the anaphor and represents the person or object the anaphor is referring to.

By using dependency analysis and coreference analysis which have been studied in the field of natural language processing, it is possible to retrieve contents in

```
<article>
  <name> George W. Bush </name>
  <body>
    <p> George Walker Bush (born July 6, 1946) is the forty-third
    and current President of the United States of America.
    He previously served as the forty-sixth Governor of Texas
    from 1995 to 2000 and is the eldest son of former United
    States President George Herbert Walker Bush.
    He was inaugurated as President on January 20, 2001 and his
    current term is scheduled to end at noon on January 20, 2009.
  </p> ...
</body>
</article>
```

Fig. 1. XML document and coreference

which query keywords have semantic dependencies so that search precision can be improved. Coreference analysis that identifies objects that refer to the same entity in documents is performed by machine learning using probabilistic knowledge such as grammatical relationships and string matching. These clues are called features. However, applying dependency analysis and coreference analysis to document retrieval still has a problem because they need language resources such as large-scale dictionaries and their accuracies are still not enough for web pages.

On the other hand, in structured documents such as XML and HTML, coreference is often influenced by their document structures. For example, in Fig.1, a person's name appears in a name node which holds the title of the article. The possibility that the contents about the person are written all over the article is high. Therefore, it can be concluded that it is mostly certain that "He" in the paragraph of Fig.1 indicates the person of the article name, though "George Hervert Bush" is closer in the text.

In this way, for deciding whether coreference relation exists between two words, we can utilize probabilistic relationship between the relative logical positions of these words and coreference. We call it a *structural feature*. Document structures such as sections and itemizations are usually given by one or more writers at the time the document was authored. Therefore, we can reasonably assume that strong correlation exists between coreference and document structures.

Regarding structural coreference, extracting relationship between attributes and values from tables in HTML documents have been studied[3][16]. Also for improving accuracy of image search, extracting sentences that describe the target image have been studied[5][17]. As for HTML and XML documents, it is thought that the search accuracy of web pages can be improved by using structural information of documents. However, for documents of Web pages, not only the document structure but also linguistic relationships needs to be considered, because a long text may be written in one text node of a document and coreference can exist within the same text node. Therefore, retrieval with high accuracy requires both linguistic features and structural features. To the authors' knowledge, no work has been done in the literature on integrating linguistic features

and structural features of web documents to produce a probabilistic model for coreference analysis.

In this paper, we propose construction of structural features from XML document trees, where structural features represent subtree patterns having positive or negative correlation with coreference between antecedents and anaphors. We also introduce linguistic features from existing work for detecting coreference of person subjects. Then a maximum entropy model is constructed from training data[2]. Experimental results show that our approach of combining structural and linguistic features have superiority over the methods that consider either structural or linguistic features alone. Also, since our abstraction of subtrees patterns is effective, our definition of structural features overperformed support vector machine with tree kernel. Our method has little part that depends on the natural language of the interest; switching from Japanese to other languages can be done by replacing language-specific features and dictionaries.

The rest of the paper is organized as follows: In Section 2, we describe related work. In Section 3, we describe our approach of analyzing coreference for person retrieval that utilizes both linguistic and document-structural features. In Section 4, we present experimental results of coreference analysis on XML documents converted from Wikipedia. Section 5 is a conclusion.

2 Related Work

A number of researches have been done on utilizing natural language analysis for web retrieval. Reputation mining from the Internet extracts pairs of “object” and “evaluation” from a large amount of free sentences regarding commodities of multiple fields. By associating the extracted “object” and its “evaluation” in the same sentence, the structure of the pair is extracted. In [10], pairs of “attribute” and “evaluation value” that appear in different sentences are extracted by applying coreference analysis for opinion extraction. Feature-based learning approach has been extensively used for coreference analysis. Features regarding document structures, such as distance in the number of sentences, are considered in [6][10], but in a simple way that the feature checks whether or not an antecedent and anaphor are apart in more than three sentences. Such a feature fails to capture long-distance coreference occurring in structured documents, such as those between the title and its body. Also, structural constructs such as itemization and section/subsections have never been considered in the coreference analysis research. In this paper we define structural features that effectively capture coreference patterns.

Table analysis and image retrieval also extract relationships between objects by considering the document structure of web pages. Semantic information such as attributes and their values can be extracted from a table by considering relative positioning of cells in the table, as well as the relationship between captions and tables. The method of [3] considers similarity between cells in the table to extract attribute and attribute values. The method of [16] also considers clustering of similar tables by analyzing data contents and its configuration in the tables.

As for image retrieval, there are researches on extracting sentences and/or keywords that are related to an image from its surrounding texts. In [5], three preceding document nodes till the parent node are considered for candidates containing sentences explaining the image. In [17], three types of web contexts regarding usage of the image are defined, by considering sentences appearing around the image, the document structure and the link structure.

Our method is unique in the respect that we utilize the fact that the same author(s) introduces logical structures as well as coreference between sentences, so that document structures and coreference patterns should have probabilistic correlations. By integrating linguistic and structural learning, a great improvement on retrieval accuracy can be expected. Also, unlike the approaches mentioned above, our approach can automatically capture cases where coreference occurs between two distant positions in a document, such as coreference between section titles and their bodies.

Recently, information retrieval on semi-structured data is extensively studied, where IR-style full-text querying is integrated with structural querying on XML documents[1][13]. Query answers are ranked by relevancy scores, where IR-style scoring of term and tag frequencies is extended to find most-relevant subtrees. Relevancy scoring also considers other factors such as exhaustivity and specificity. However, these scoring models do not directly reflect linguistic relationship between keywords as natural-language sentences, rather indirect indicators of linguistic relationships, such as term proximities and term distributions, are used. A finer search accuracy shall be achieved, by combining structural factors and natural-language factors.

3 Linguistic and Structural Features of XML Documents

We describe our approach on person retrieval by structural and linguistic coreference on XML documents.

3.1 Coreference

Machine learning with features has been used for analyzing coreference. In this paper, we describe our method of extracting linguistic and structural features from input XML documents, to obtain pairs of antecedent and anaphor candidates which appear in the documents. Then we apply machine learning over those features for learning coreference relations.

Linguistic features. The following four types of linguistic features are from [6]. These features have little implementation complexity but have good classification accuracies.

1. **Features that use vocabulary information** (10 features)

Scores on string matching of an anaphor candidate and an antecedent candidate are introduced as features. String matching is further classified as “complete matching”, “forward matching”, “backward matching”, “main

word (most right content word) matching”, etc. In general, it is likely that an anaphor candidate and an antecedent candidate are the same objects if they have matching substrings.

2. Features that use morphological and syntactical information (7 features)

Grammatical information of both anaphor and antecedent candidates, such as their parts of speech, demonstratives, particles, tense of attributive modifiers of noun phrases, can be used as features. For example, when instruction attribute “the” lies on top of a noun phrase, the probability of the noun phrase being a fixed noun phrase is high.

3. Features that use semantic information (6 features)

Semantic information from dictionaries and collections of peculiar expressions can be used as features. Peculiar expressions include person names and place names. We use Cabocha[9] for obtaining peculiar expression analysis. If semantic attribute of an anaphor candidate and an antecedent candidate is the same, there is a high possibility that they have coreference.

4. Features that use distance information between noun phrases (3 features)

If there is a relatively long distance between an anaphor candidate and an antecedent candidate, the possibility of not having coreference is high. We can use as features regarding several different distances, such as the sentence distance and noun-phrase distance between anaphor and antecedent candidates.

The above grammatical and semantical features are dependent on the target natural language. In this paper, we are dealing with XML documents containing Japanese texts, but our method can be easily modified to another target language by incorporating grammatical and semantical features of the target language.

Structural features. In structured documents such as XML and HTML, coreference can be strongly influenced by their document structures. Even if two words are separated in a document, there is a high possibility that they are in the coreference relation if they have a parent-child relationship in the XML document tree. On the contrary, even if an anaphor candidate and an antecedent candidate appear in the vicinity of text, coreference might not occur if they are distant in the document tree. Therefore, features regarding structural characteristics of document trees shall be introduced for coreference analysis. We focus on the document path from the node of an antecedent candidate to the node of an anaphor candidate, possibly via their least common ancestor, and call the path a *coreference combination subtree (CCST)*. We utilize structural characteristics of the subtree as features of coreference analysis.

Fig.2 is an example of generating a CCST. Fig.2 (a) shows an entire XML document tree. An antecedent candidate appears in the title node of a section node, and an anaphor candidate appears in a paragraph node of a section node. Its CCST is generated automatically as shown in Fig.2 (b). As CCSTs include essential information that influence coreference, such as distance between antecedent and

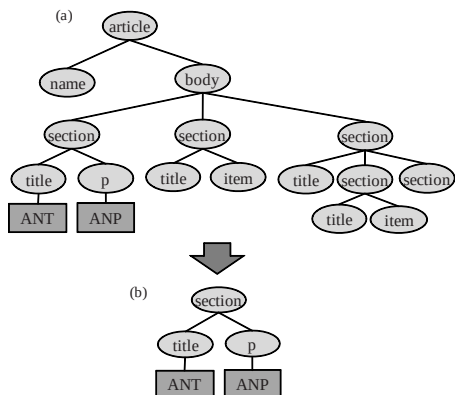


Fig. 2. Generating the CCST

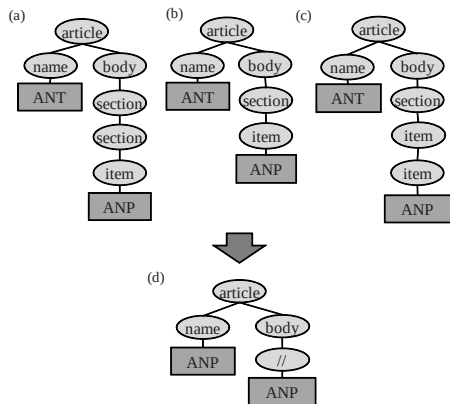


Fig. 3. Generating the k-CCST

anaphor candidates in the document tree and node types (such as section, title, body, etc), we can employ CCSTs as structural features.

If we generate CCSTs for all the combination of antecedent and anaphor candidates within a document, the number of subtrees is too large to be efficient features. Also, there could be many similar subtrees which represent the same coreference pattern. Such similar subtrees need to be grouped. We extract a subtree whose height is k from the root of the CCSTs. In this paper, we call it a *k-coreference combination subtree (k-CCST)*. We focus only on top- k portion of a subtree and replace the subtrees below height k with nodes representing arbitrary subtrees. This generalization is simple but based observation that coreference is frequently occurring either (a) between two nodes connected by a small number of edges or (b) between a node representing a logical region such as section body and a node representing its title.

Fig.3 shows generation of k-CCST at $k=2$. Three CCSTs of Fig.3 (a) (b) and (c) are different from each other, but all of them have the same important characteristics such that all the antecedent candidates appear in “name” nodes, while the anaphor candidates appear in “body” nodes. Fig.3 (d) shows the k-CCST with $k=2$, which generalizes the subtrees of Fig.3 (a)(b) and (c) by substituting subtrees deeper than $k=2$ with the descendant-or-self node “//”. We employ k-CCSTs as structural features. We use the following parenthesized notation of subtrees for structural features:

(article (name ANTECEDENT)(body ANAPHOR))

Here, the paths from the subtree root to both ANTECEDENT and ANAPHOR nodes contain at most k nodes, and they are regarded as having “//” nodes just in front of the ANTECEDENT and ANAPHOR nodes.

Learning machine. As for learning machines, we consider two models: the maximum entropy model[2] has been successfully used for feature-based learning

in natural language processing, and support vector machine (SVM) utilizing tree kernels are known to be effective for learning tree-structured data[7].

1. Maximum entropy model

The maximum entropy model is an algorithm which estimates a probabilistic model for conditional probabilities $p(x, y)$ from given frequencies $C(x, y)$ such that events x and y co-occur.

First, arbitrary functions, called feature functions, that return 1 or 0 are defined over two events x and y , and constraints on feature functions are computed. For example, suppose that event x is a structural feature represented by the k-CCST shown below:

(article (name ANTECEDENT)(body ANAPHOR))

Also suppose that event y represents the coreference relation (the correct answer) between the antecedent and anaphor candidates. Then we have the following feature function:

$$f_i(x, y) = \begin{cases} 1 & \text{if } x \wedge y \\ 0 & \text{otherwise} \end{cases}$$

This function returns 1 when the antecedent candidate appears in the article name, the anaphor candidate appears in the body and they are actually in the coreference relation. The probabilistic model is calculated by the formula (1).

$$p(x, y) = \frac{1}{z(x)} e^{\sum_i \lambda_i f_i(x, y)} \quad (1)$$

$$z(x) = \sum_y e^{\sum_i \lambda_i f_i(x, y)} \quad (2)$$

Here, λ is a parameter which defines weight on each feature. Also, $z(x)$ is a normalization factor. Here, the probabilistic model needs to satisfy the following constraints: $\tilde{P}(f_i)$ is the expected value of features by the training data, and $P(f_i)$ is the expected value of features obtained by the model. The model is required to make these expected values identical. The entropy model is represented by the formula (3):

$$H(P) = - \sum_{x, y} p(x, y) \log p(x, y) \quad (3)$$

2. Support vector machine(SVM) using tree kernel

SVM classifies classes, constructing a separation hyperplane to maximize the margin which is a distance between training points called support vectors located in the class boundary neighborhood and separating plane[4]. If it is not possible to linearly separate, it classifies training data by mapping the input space to a higher-dimension feature space by using kernel trick. The kernel method accesses in the form of the inner product of two data.

The function that gives the product is called a kernel function and SVM can classify higher dimensional data by selecting an appropriate kernel function. Tree kernels for learning tree-structured data are known[7]. When two trees V_1, V_2 have node sets T_1, T_2 and edge sets E_1, E_2 , the tree kernel is defined as follows.

$$K(T_1, T_2) = \sum_{v_1 \in V_1} \sum_{v_2 \in V_2} \sum_{s_1 \in S_{v_1}(T_1)} \sum_{s_2 \in S_{v_2}(T_2)} K^S(s_1, s_2) \quad (4)$$

$$K^S(s_1, s_2) = I(s_1 = s_2) \quad (5)$$

Here, $S_v(T)$ is the set of subtrees having $v \in V$ as roots and K^S is assumed to be a kernel function defined between two subtrees. $I()$ in the formula (5) is a function which returns 1 if and only if the proposition in the parentheses is true and returns 0 otherwise. Also, $s_1 = s_2$ becomes true if and only if two subtrees are completely identical. As mentioned above, SVM can be used to learn document structures by using tree kernel. We use the combination of vectors of linguistic features returning 0 or 1 and tree structures from CCSTs as training data. We note that we do not restrict the height of CCSTs to given k for SVM, since tree kernel has its own generalization mechanism.

In this paper, we adopt both the maximum entropy model and SVM as learning machines and compare their performances. We call the model generated from training data a *coreference model*.

3.2 Applying Coreference Analysis to Person Retrieval

When a coreference exists within a document where the antecedent is a person name, we can reasonably assume that the anaphor part contains descriptions related to the person. As for CCSTs, if the feature

(article (name ANTECEDENT)(body ANAPHOR))

has relatively high weight after learning and thus correlation between this feature and coreference is highly likely, there is a high possibility that the body node containing the anaphor candidate mentions about the person name appearing in the article-name attribute node. In this way, for person retrieval on XML documents by pair keywords of person name and person's attribute, retrieval answers should be improved by reporting subdocuments in which the person name and person's attribute are in the coreference relation. Also, in a more flexible way, we can sort subdocuments by the probabilities of the coreference relation of each subdocument.

4 Experiments

We have conducted a series of experiments to evaluate effectiveness of structural features to coreference analysis of XML documents.

Table 1. Experimental data

	Ex.1	Ex.2	Ex.3	Ex.4
number of ANAPHOR	333	521	6	72
number of ANTECEDENT	75	96	4	4
number of pairs	12,597	22,758	18	33
number of correct answers	240	390	7	14
number of incorrect answers	12,357	22,269	11	19

4.1 Experimental Conditions

Experimental data. XML documents converted from Japanese version of Wikipedia¹ are used for benchmark. We used the following four articles:

- Ex.1: Koki Kameda (professional boxer)
- Ex.2: Yasuo Fukuda (prime minister)
- Ex.3: The gasoline dispute in the Diet
- Ex.4: The Security Council of Japan

Ex.1 and Ex.2 are articles whose article name is a person’s name. Ex.3 and Ex.4 are articles whose article name is other than a person’s name. The characteristics of experimental data are shown in Table 1. Since our focus is person retrieval, we try to extract coreference regarding persons. In this setting, an anaphor candidate should be a noun phrase or a zero-pronoun (Japanese-specific grammar where pronoun is omitted) that represents a person in a document, and an antecedent candidate should be a person’s name. For example, “Bush”, “him”, and “president”, etc. can become anaphor candidates, while “George Walker Bush” and “Barack Hussein Obama, Jr ”, etc. are regarded as antecedent candidates. Antecedent candidates of a specified anaphor candidate need to be retrieved from the entire text preceding the anaphor candidate.

Traditional coreference analysis often searches antecedent candidates within the same paragraph or within a few preceding sentences of an anaphor candidates. On the other hand, for structured documents such as XML, structural relationship can cause coreference between two distant locations in the text. Another difference is that learning of document structures needs to be done from the entire document. As shown in Table 1, the number of incorrect answers is significantly larger than that of correct answers, because every pair of antecedent and anaphor candidates within a document becomes incorrect coreference if it is not correct.

If our method is applied to non-human subjects, linguistic features for detecting human subjects, such as peculiar expressions for humans, need to be replaced by features for detecting subjects of the interest.

Features. We introduced linguistic features from [6]. In our experiments, we used Chasen[12] and Cabocha[9] for morphological analysis, peculiar expression tag allocation, and dependency analysis. These processes are automatic. For semantic features involving dictionaries, we use the EDR electronic dictionary[14].

¹ <http://ja.wikipedia.org/>

Table 2. Distribution of k-CCSTs(k=2) about all examples

k-CCSTs(k=2)	number of right answers
(1) (item ANTECEDENT ANAPHOR)	91
(2) (p ANTECEDENT ANAPHOR)	153
(3) (item ANTECEDENT (normalist ANAPHOR))	4
(4) (article (name ANTECEDENT)(body ANAPHOR))	357
(5) (section (title ANTECEDENT)(section ANAPHOR))	4
(6) (body (p ANTECEDENT)(section ANAPHOR))	42
(7) (section (section ANTECEDENT)(section ANAPHOR))	0
(8) (normalist (item ANTECEDENT)(item ANAPHOR))	0
(9) (section (normalist ANTECEDENT)(normalist ANAPHOR))	0
(10) (section(p ANTECEDENT) (p ANAPHOR))	0
(11) (section (section ANTECEDENT)(section ANAPHOR))	0
sum	651

In the EDR concept dictionary, each entry word has the following binary flags: “Human”, “Human’s attribute”, and “Human and subject having human-like behavior”. We used these flags as features to probabilistically infer that the word is referring a human.

Structural features of XML documents of Wikipedia are automatically generated based on the algorithm illustrated in Fig.2 and Fig.3. Structural features of k-CCSTs at k=2 generated from all examples are shown in Table 2.

In Table 2, subtree (1) and (2) are the cases where the coreference pair appears in the same item node or in the same paragraph. Subtree (3) is the case where an antecedent candidate appears in an item node and its anaphor candidate appears in its child element, meaning that the coreference pair has a parent-child relationship. Subtree (4) is the case that an antecedent candidate appears in a name node that is the title of the article, and (5) is the case where an antecedent candidate appears in the title node of a section node and an anaphor candidate appears in the section body. Subtree (6) is the case where an antecedent candidate and an anaphor candidate appear in different a section and a paragraph. The subtrees (7)–(11) mean that the antecedent candidate and the anaphor candidate have a sibling relationship in the document tree structure. It is observed that the frequency of coreference relations occurring within the same document node of XML is high. However, the frequencies of coreference relations occurring between a parent and a child (3), and between a “name” or “title” node and its body are also high. These results show that certain structural patterns have significantly higher possibilities of coreference than others, so that structural features can be an effective classifier for coreference.

Learning machines. As for learning machines, we compare the maximum entropy model and SVM with tree kernel. We use the tool [11] for the maximum entropy model and *SVM^{light}* [15] for SVM with tree kernel. For the maximum entropy model, linguistic features and structural features from k-CCSTs with varying k are used for input. For the SVM with tree kernel, the same linguistic features and structural features from the entire CCSTs are used. The training data and the classification data are selected from all data at random half.

4.2 Evaluation Method

The data classified by the learning machines are evaluated according to precision and recall. Coreference relations corresponding with the results by a human subject are counted as correct answers. Precision and recall are calculated by the following formulas:

$$precision = \frac{\text{number of coreference relations correctly identified}}{\text{number of coreference relations identified by the system}}$$

$$recall = \frac{\text{number of coreference relations correctly identified}}{\text{total number of correct coreference relations}}$$

Since the recall tends to fall if the precision rises, and the precision falls if the recall rises, we use F-measure that is the harmonic mean of the precision and the recall as an evaluation measure. F-measure is calculated by the following formula:

$$F\text{-measure} = \frac{2 \times precision \times recall}{precision + recall}$$

4.3 Experiment Results and Discussions

To evaluate effectiveness of structural features for coreference analysis of XML documents, we have compared analysis results with and without structural features. In addition, in order to examine the effect of height parameter k of k -CCSTs, we tested for $k=2$, $k=3$ and $k=\infty$. Moreover, we compared results of SVM with tree kernel and the maximum entropy model, in order to investigate which tree generalization is more effective.

Comparing performance of features. To evaluate whether structural features is effective for analyzing coreference in XML documents, we implemented the following three methods using different features: (I) coreference analysis using only linguistic features, (II) coreference analysis using both linguistic and structural features, and (III) coreference analysis using string matching and structural features. We use the maximum entropy model for learning machine. As for structural features, k -CCSTs at $k=2$ is used. The experiment result is shown in Table 3.

From Table 3, the analysis using both linguistic and structural features (II) has better F-measure values for all the test data than the analysis using linguistic features only (I). From this, we can see that structural features can improve coreference analysis of XML documents. By comparing coreference analysis using string matching and structural features (III) with coreference analysis using structural features (II), the former obtained results such that precision is high but recall is low. This can be explained as the analysis of (III) can return correct answers when partial string matches are identifying correct coreference, while this method tends

Table 3. Experiment results about Comparing performance of features (%)

		Ex.1	Ex.2	Ex.3	Ex.4
(I)linguistic	precision	74.3	76.0	51.3	31.2
	recall	40.8	66.8	57.5	48.7
	F-measure	52.7	71.1	54.2	38.0
(II)linguistic & structural	precision	77.0	78.9	69.3	75.0
	recall	48.1	69.2	91.7	54.8
	F-measure	59.2	73.7	74.9	63.3
(III)string matcing & structural	precision	90.6	92.0	82.0	86.0
	recall	38.9	33.5	62.0	54.8
	F-measure	54.4	49.1	70.6	66.7

Table 4. Experiment results about height of CCSTs (%)

		Ex.1	Ex.2	Ex.3	Ex.4
k=2	precision	77.0	78.9	69.3	75.0
	recall	48.1	69.2	91.7	54.8
	F-measure	59.2	73.7	74.9	63.3
k=3	precision	75.4	77.6	69.3	75.0
	recall	46.2	68.6	91.7	54.8
	F-measure	57.3	72.8	74.9	63.3
k= ∞	precision	72.1	78.0	69.3	75.0
	recall	49.3	72.0	91.7	54.8
	F-measure	58.6	74.9	74.9	63.3

Table 5. Experiment results about comparing learning machines (%)

		Ex.1	Ex.2	Ex.3	Ex.4
maxent	precision	72.1	78.0	69.3	75.0
	recall	49.3	72.0	91.7	54.8
	F-measure	58.6	74.9	74.9	63.3
tree kernel	precision	97.4	85.3	63.3	88.0
	recall	30.6	61.2	83.3	30.4
	F-measure	46.6	71.3	719	45.2

to fail for the cases that cannot be judged by partial string matching, like the case where “George Walker Bush” is mentioned as “the president”.

The recalls for all the feature sets are not better than expected. It is because the number of negative instances is extremely larger than positive instances, so that learning models tend to return negative answers. For this problem, we expect improvement by assigning weights to instances and/or ordering instances according to the document structure; tournament model[6] shows ordering on flat texts.

Height of CCSTs. We compared k-CCSTs of varying k, where k=2, 3, and ∞ (no pruning by height). Both linguistic and structural features were used, and the maximum entropy model was used for learning. The results are shown in Table 4.

For Ex.3 and Ex. 4, all subtrees have obtained the same results, because these documents do not have CCSTs of height larger than two. For Ex. 1 and Ex. 2, k=2 has obtained better results for both precision and recall against k=3. For k= ∞ , namely the case where CCSTs not pruned by the height were used as structural has obtained results such that the recall is the highest of three while the precision is low. As for F-measure, one at k=2 is the highest for Ex.1 and one at k= ∞ is the slightly highest for Ex. 2. In total, there was no clear difference for varying tree heights. However, if CCSTs with k= ∞ are used as structural features, the number of features becomes enormous as documents become complex. In addition, generality is lost and structural features from k= ∞

tend to become hard to be applied to other documents. Therefore, by the point that k-CCSTs with k=2 produces more general and less number of features, subtrees with k=2 are producing most desirable results.

Comparing learning machines. We compared performance of coreference analysis by the maximum entropy model and SVM with tree kernel. As for features, we use both linguistic and structural features for this experiment. To compare the learning methods, CCSTs with $k=\infty$ were used as structural features. Table 5 shows the experiment results.

For all the test examples, learning by the maximum entropy model has obtained better F-measure values than that of SVM using tree kernel. Except for Ex. 3, SVM using tree kernel has higher precision and lower recall than those of the maximum entropy model. Namely, SVM using tree kernel returns correct answers with higher probabilities, but the fewness of correct answers impacted the F-measure. Since the expression of feature space is implicit in the kernel model, it is hard to investigate the cause of this tendency. However, the dropped correct answers indicate that SVM with tree kernel is failing to capture the patterns where anaphors are occurring in arbitrary levels of a body subtree.

5 Conclusion

In this paper, we proposed a new way of person retrieval through coreference analysis that utilizes structural features of XML documents. As for structural features, we proposed k-CCSTs, and our experimental results over XML documents of Wikipedia show significant improvement of accuracy to coreference analysis which utilizes only linguistic features. F-measure of the coreference analysis by structural and linguistic features is higher than that of coreference analysis by only linguistic features. We also found that the height of k-CCSTs shows little impact on accuracy for our benchmark, and subtrees with k=2 were most advantageous because of their generality and compactness. Finally, we compared the maximum entropy model and SVM using tree kernel as learning machines for our person retrieval. The results show that F-measure by the maximum entropy model was higher than that by SVM using tree kernel.

As future work, improvement on recall can be expected by resolving the problem of unbalanced positive and negative instances. Also, we would like to apply our method to various collections of XML and HTML documents, and to apply retrieval of non-person entities, such as merchandizes, for reputation mining.

Acknowledgments

This work is in part supported by a Grant-in-Aid for Scientific Research of MEXT Japan (#18300031), and Strategic International Cooperative Program of JST (Japan Science and Technology Agency).

References

1. Amer-Yahia, S., Lalmas, M.: XML search: languages, INEX and scoring. ACM SIGMOD Record 35(4) (2006)
2. Berger, A.L., Pietra, S.D., Pietra, V.J.D.: A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics 22(1), 39–71 (1996)
3. Chen, H., Tsai, S., Tsai, J.: Mining Tables from Large Scale HTML Texts. In: 18th International Conference on Computational Linguistics, pp. 166–172 (2000)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
5. Idehara, H., Fujimoto, N., Takeno, H., Hagihara, K.: A Sentence Extraction Technique Based on HTML Parsing Tree Structures around Images for WWW Image Retrieval. IEICE technical report. Dependable computing 105(340), 19–24 (2005) (in Japanese)
6. Iida, R., Inui, K., Matsumoto, Y., Sekine, S.: Noun Phrase Coreference Resolution in Japanese Based on Most Likely Antecedent Candidates. Transactions of Information Processing Society of Japan 46(3), 831–844 (2005) (in Japanese)
7. Kuboyama, T., Shin, K., Kashima, H.: Flexible Tree Kernels based on Counting the Number of Tree Mappings. In: Workshop on Mining and Learning held with ECML/PKDD (2006)
8. Kehler, A.: Probabilistic Coreference in Information Extraction. Association for Computational Linguistics, 163–173 (1997)
9. Kudo, T., Matsumoto, Y.: Chunking with Support Vector Machines. IPSJ SIG Notes 2000(107), 9–16 (2000) (in Japanese)
10. Kobayashi, N., Clida, R., CInui, K., Matsumoto, Y.: Opinion Extraction Using a Learning-Based Anaphora Resolution Technique. In: The Second International Joint Conference on Natural Language Processing, pp. 175–180 (2005)
11. Le, Z.: Maximum Entropy Modeling Toolkit for Python and C++. http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html
12. Matsumoto, Y., Kitauchi, A., Yamashita, T., Hirano, Y., Matsuda, H., Takaoka, K., Asahara, M.: Morphological Analysis System ChaSen version 2.2.9 Manual. Nara Institute of Science and Technology (2002)
13. Theobald, M., Bast, H., Majumdar, D., Schenkel, R., Weikum, G.: TopX: efficient and versatile top-k query processing for semistructured data. The VLDB Journal 17(1) (2008)
14. Yokoi, T.: The EDR electronic dictionary. Communications of the ACM 38 (1995)
15. SVM^{light}, <http://dit.unitn.it/~moschitt/Tree-Kernel.htm>
16. Yoshida, M., Torisawa, K., Tsujii, J.: Extracting ontologies from World Wide Web via HTML tables. Pacific Association for Computational Linguistics, 332–341 (2001)
17. Zettsu, K., Tanaka, K.: Extraction and Visualization of Image Contexts from Web. In: DEWS, 6-p-05 (2003) (in Japanese)

HFilter: Hybrid Finite Automaton Based Stream Filtering for Deep and Recursive XML Data

Weiwei Sun, Yongrui Qin, Ping Yu, Zhuoyao Zhang, and Zhenying He

School of Computer Science and Technology, Fudan University

220 Handan Road, Shanghai 200433, China

{wwsun, yrqin, yuping, zhangzhuoyao, zhenying}@fudan.edu.cn

Abstract. XML filtering applications are gaining increasing popularity recently. Automata are generally adopted to construct query indexes for evaluating large numbers of XPath queries over XML streams. Usually only shallow data are observed in existing approaches. How to process deep and recursive XML data with low memory limitation efficiently is still a challenging issue. In this paper, we propose HFilter, a Hybrid Finite Automaton (HFA) based stream filtering approach, to solve this problem. We introduce the basic two-tier HFA (lazy DFA tier and NFA tier) first, which realizes data prefix sharing and memory overflow control to improve the filtering throughput. Then an optimized three-tier HFA with an extra pre-expanded DFA tier is put forward, which significantly reduces the restarting cost of HFA after memory overflow. Experiments show that our approaches work more efficiently than existing ones.

Keywords: XML stream filtering, deep and recursive XML data, hybrid finite automaton, data prefix sharing, memory overflow control.

1 Introduction

XML has been widely accepted as the *de facto* standard for data exchange. How to obtain interested information from XML streams has become a problem of great importance. The XML filtering system matches continuously arriving XML documents to large numbers of queries and delivers the matched documents accordingly. Queries in XML filtering systems are expressed in XPath generally. Automata are widely adopted to index XPath to accelerate evaluating XPath queries over XML streams.

Most works focus on improving the throughputs of filtering engines, however, few of them study the problem of filtering deep and recursive XML data efficiently. Usually only shallow XML data are observed in experiments. For example, the average depth of XML streams in most experiments is 5 to 6 in YFilter [2, 4].

However, complicated XML data predominate in some situations. For example, the source data for experiments in [5] illustrate ten different data sets, four of them have the average depth greater than 11, and recursion exists in half of them. The maximum depth of these data sets even extends to 36. How to process stream of deep and recursive XML data efficiently is an essential and imminent problem.

This problem becomes particularly challenging in the context of low memory limitation. NFA based approaches are space efficient. They can deal with any kinds of XML data with low memory requirement. But their throughputs are usually low. Lazy DFA based approaches are time efficient. But for complicated XML data, they require exponential number of states [5] and run out of memory frequently. In this case, their performances are even worse than that of NFA based approaches.

Similar to the query set, paths from the root to other elements of XML data instances also have the same prefixes. If the engine memorizes the active states of these prefixes, it will need to compute them only once and thus improve the filtering performance. The elements at deeper locations of paths are encountered with lower possibility than nodes in their prefixes. Memorizing states of these deep elements will consume a lot of the main memory and have less improvement on the filtering performance.

Based on these observations, we propose HFilter, a Hybrid Finite Automaton (HFA) based filtering approach for deep and recursive XML data with low memory limitation. The basic HFA consists of two tiers (Fig. 1(a)). The core of an HFA is a lazy DFA. The shell of an HFA is an NFA.

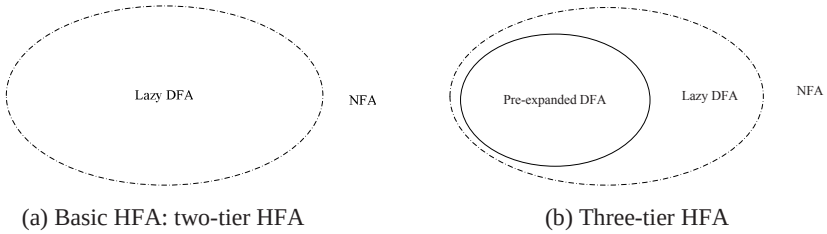


Fig. 1. HFA: Hybrid Finite Automaton

This hybrid structure takes advantage of both NFA and lazy DFA, and realizes data prefix sharing and memory overflow control. First, the core processes the shallow parts of the XML data instances. It memorizes the active states of the data prefixes which are short and encountered very frequently. Elements of these common prefixes can be processed efficiently. Second, the shell processes the deep parts of the XML data instances, which have few common data prefixes. All the active states for these parts are deleted immediately after being processed to reduce the memory consumption.

Since the core of HFilter is a lazy DFA, memory overflow is inevitable after running for a long time generally. To reduce the cost of restarting, we propose a three-tier HFA (Fig. 1 (b)). Its core is divided into two tiers: DFA and lazy DFA. The DFA tier is pre-expanded and memory-resident during the whole running time. When the system runs out of memory, only the lazy DFA tier is cleared and needs to be recomputed. New states in the lazy DFA tier can be re-expanded directly from the DFA tier instead of from the initial state. Therefore, the three-tier HFA restarts more quickly and performs much better in the warming up phase.

The rest of the paper is organized as follows. Section 2 provides background on XML filtering and sketches the NFA based approaches and lazy DFA approaches.

Section 3 and Section 4 propose the HFilter strategy, discuss its two-tier and three-tier implementations. Section 5 shows the performance analysis and comparison. Section 6 surveys related works. Section 7 presents our conclusions and discusses future works.

2 Background

Fig. 2 (a) is an example of a recursive DTD. Fig. 2 (b) shows an instance of this DTD. Fig. 2 (c) gives a sample query set, *Q*.

In this paper, we define *location level* of an element as its position in a path from root. For example, in the path {*a/b/b*}, the *location level* of root element */a* is 1, the *location level* of the first */b* in this path is 2, and that of the second */b* is 3.

In XML stream processing systems, queries, instead of the data, are usually indexed. Two typical finite automaton based filtering engines are widely adopted, which are NFA based [1, 2, 3, 4, 9, 10, 13] and lazy DFA based [5, 6, 7, 8, 12].

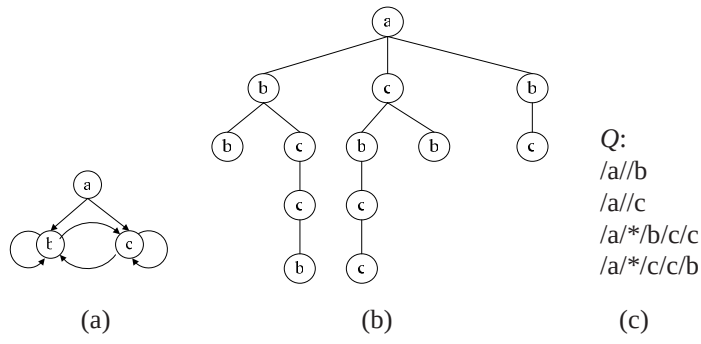


Fig. 2. (a) DTD graph, (b) an XML instance of (a), (c) a query set *Q*

2.1 NFA Based Filtering Engine

Considering the prefix commonality in the query set, an NFA based filtering engine constructs the NFA as the query index from the query set. For each query, an NFA will be constructed, and all these NFAs will be combined into a single NFA [2, 4]. Fig. 3 shows an example of a combined NFA for the query set *Q* shown in Fig. 2(c).

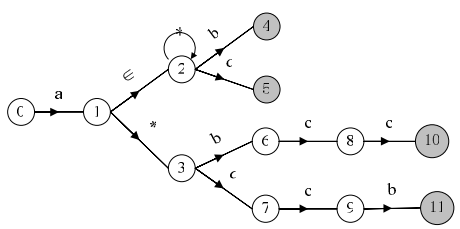


Fig. 3. The combined NFA for *Q* shown in Fig. 2 (c)

For the NFA based filtering engine, a special runtime stack is maintained. Since the NFA may have many active states at the same time, the stack should be able to track the multiple active paths.

Example 1. To process the path $\{/a/b/b\}$ in the XML instance shown in Fig. 2 (b), the initial top state set of the runtime stack is $\{0\}$. After the “start-of-element” event of the root $/a$ has been processed, the top state set is $\{1, 2\}$. Then after the “start-of-element” event of the first $/b$ has been processed, the top state set is $\{2, 3, 4\}$. And after processing the second $/b$, the top state set is $\{2, 4, 6\}$. Then the following events are three “end-of-element” events and the top three state sets are popped.

The characteristics of NFA based approaches can be described as follows:

- **Advantages:** The NFA consumes a very small amount of main memory.
- **Disadvantages:** This mechanism always needs to compute the top set of the active states. Many of the state sets are repeatedly computed, which costs a lot of execution time.

2.2 Lazy DFA Based Filtering Engine

In general, for a DFA based filtering engine, constructing the eager DFA will result in exponential number of states and lead to memory overflow. Therefore, the DFA is constructed in a lazy way in most cases and called the lazy DFA [5].

For the lazy DFA based filtering engine, a special query index is maintained, while a special runtime stack is maintained in NFA. Every DFA state of lazy DFA contains an NFA state table and thus new DFA states can be computed at the runtime other than pre-computed before filtering and save the physical memory. The runtime stack of lazy DFA always pushes or pops only one DFA state.

The filtering engine memorizes all sets of the active NFA states which have been encountered by inserting new DFA states into the lazy DFA. When the same set of active NFA states are needed next time, the engine can get the target DFA state immediately which leads to a high throughput.

Example 2. When processing the path $\{/a/b/b\}$ using lazy DFA three sets of active states will be generated: $\{1, 2\}$, $\{2, 3, 4\}$ and $\{2, 4, 6\}$. Fig. 4 shows an example of a lazy DFA: the lazy DFA after processing the whole XML instance shown in Fig. 2 (b). The NFA state ID sets in the eclipses in Fig. 4 are NFA tables.

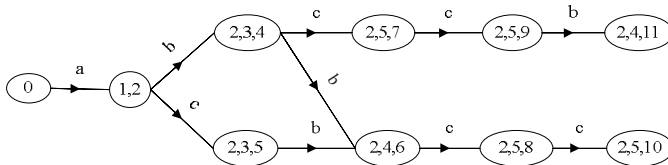


Fig. 4. An example of lazy DFA

The lazy DFA engine has a warming up phase at the beginning of the filtering and enters a stable phase later. During the warming up phase, the engine spends a lot of time in expanding new lazy DFA states. Therefore, the runtime filtering performance is much worse than that during the stable phase.

The characteristics of lazy DFA based approaches can be described as follows:

- **Advantages:** The lazy DFA based filtering engine usually has a much better filtering performance than the NFA.

- **Disadvantages:**

- a) It consumes more runtime memory than the NFA based engine. Frequent memory overflow may happen when processing deep and recursive XML data.
- b) The warming up cost is very high. When the filtering system restarts frequently due to memory overflow, it suffers a lot from the warming up phase and degrades to very low performance.

3 HFilter: A Hybrid Finite Automaton (HFA) Based Filtering Approach

When the main memory is large enough, the lazy DFA based approaches always perform much better than the NFA based approaches. However, when processing deep and recursive XML data, lazy DFA may perform even worse than NFA.

This is because the memory consumption of lazy DFA is very high and that might lead to frequent memory overflow. When overflow happens, the system needs to delete all the expanded lazy DFA states to release the memory and restart the engine from the initial state. Due to the limitation of the main memory and the complexity of the XML data, the restarting may happen very frequently. Thus the lazy DFA based engine suffers a lot from the warming up phase.

To solve this problem, in this section we propose HFilter, a Hybrid Finite Automaton (HFA) based filtering approach. The basic HFA is a two-tier structured query index (shown in Fig. 1 (a)).

3.1 The Structure of Two-Tier HFA

The two-tier HFA is the combination of lazy DFA and NFA. It only memorizes the frequently processed states to improve the filtering performance and deletes the other states immediately after processed to avoid frequent memory overflow.

As mentioned in Section 1, many of the paths from the root to other elements in XML data instances usually have the same prefixes. For example, the paths $\{/a/b/c, /a/b/d\}$ have the same prefix $\{/a/b\}$. Elements at deeper locations of paths, such as the elements $/c$ and $/d$ in this example, are usually encountered much less frequently than shallow nodes in their same prefixes, such as $/a$ and $/b$.

Therefore, to construct a two-tier HFA, a maximum expanding depth D_{exp} of the HFA should be specified. The depth D_{exp} indicates that the parts with depth no greater than D_{exp} of the XML data instances will be processed by the core of two-tier HFA and the rest parts will be processed by the shell. The optimal value of D_{exp} can be specified according to apriori knowledge, such as the average depth of the data set, the frequent label path patterns in the data set, the number of XPath queries and the probability of $*$ and $//$, etc.

This structure can benefit from both of the lazy DFA and the NFA structures. When processing deep and recursive XML data, HFilter can achieve a relatively higher filtering performance and has relatively lower memory consumption.

3.2 The Execution of Two-Tier HFA

The execution of the two-tier HFA is a little more complicated than the lazy DFA and the NFA. For the HFA, it needs to maintain a special query index as in the lazy DFA, and a special runtime stack as in the NFA, at the same time.

- 1) When the *location level* of current processing element is no greater than D_{exp} , the HFA mainly maintains the lazy DFA part of the query index and works like a lazy DFA.
- 2) When the *location level* of current processing element is greater than D_{exp} , the HFA mainly maintains the runtime stack. In this case, the stack needs to process a set of multiple active NFA states each time and keep track of multiple active paths, working like an NFA.

Example 3. The part contained in the largest dotted ellipse shown in Fig. 5 is the first tier of the two-tier HFA. This tier will be cleared when the system runs out of memory. Outside of this ellipse there are four NFA state sets which are processed during the filtering process and form the second tier of HFA. They will be deleted immediately after being processed.

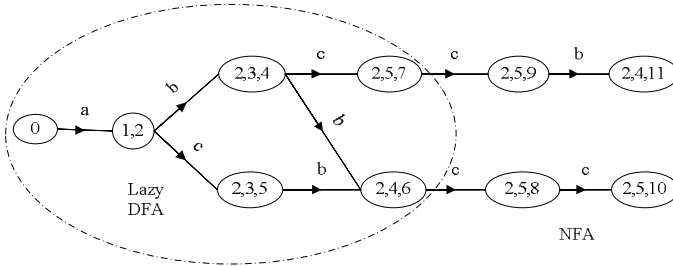


Fig. 5. An example of a two-tier HFA (with $D_{exp}=3$)

When the *location level* of current processing element is $D_{exp}+1$, the two-tier HFA works different from both of the lazy DFA and the NFA.

Case 1. If the event is “start-of-element”,

- 1) HFilter needs to push the set of active NFA states into the runtime stack first. We can accomplish this according to the NFA table of the top lazy DFA state. Because the set we want to push into the stack contains the same NFA states as those in the NFA table. Taking the execution of lazy DFA shown in Fig. 5 for example, if the top lazy DFA state is {2, 4, 6} and the current processing element is /c, then we should push the NFA state set {2, 4, 6} into the stack. Then the HFA can process /c like an NFA, which results in the NFA state set {2, 5, 8}.

- 2) After the above operation, the HFA works like an NFA. The lazy DFA state that was on top remains on the stack. In this way, when processing the “start-of-element” events of the sibling elements of current element, the stack operations can be the same.

Case 2. If the event is “end-of-element”, there are still two sets of NFA states on the top of the runtime stack. One set is for the element with the *location level* equal to $D_{exp}+1$, and the other is for the element with the *location level* equal to D_{exp} . There is also a lazy DFA state on the stack for the element with the *location level* equal to D_{exp} at the same time. The HFilter needs to pop the two sets of NFA states off the stack, but still keep the top lazy DFA state on the stack. After these operations the two-tier HFA works like a lazy DFA again.

3.3 Handling Memory Overflow

Although HFilter decreases the frequency of memory overflow, there are still chances for the system to consume all the main memory. When memory overflow happens, all the lazy DFA states have to be deleted to release the main memory. Then the lazy DFA tier of the HFA will restart from the initial state.

4 HFilter with Three-Tier HFA

When the HFilter with two-tier HFA runs out of memory, it needs to restart the HFA from the initial state which has a very high cost. On the other hand, for the elements of which the *location levels* are small, the number of DFA states for all of them is usually small too. And these DFA states are expected to be processed frequently. Therefore, we can expand the DFA states for the shallow elements in advance and reduce the cost of restarting the HFA.

In this section, we propose a three-tier HFA with an extra pre-expanded DFA tier (shown in Fig. 1 (b)) for HFilter to reduce the cost of restarting when memory overflow happens thus to further improve the filtering performance.

Fig. 6 shows an example of a three-tier HFA. The first tier of the three-tier HFA is the part contained in the largest solid eclipse. It is a pre-expanded DFA. The second tier is the part contained in the largest dotted eclipse excluding the first tier part. It is a lazy DFA. Outside of the largest dotted eclipse is the third tier. It is an NFA.

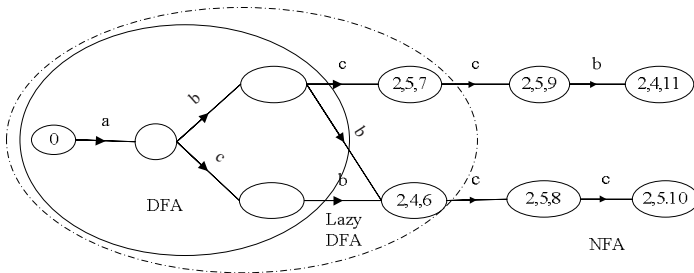


Fig. 6. An example of a three-tier HFA (with $D_{pre}=2$, $D_{exp}=3$)

When memory overflow happens, only the lazy DFA tier needs to be cleared to release the memory. Since there is a pre-expanded DFA in the three-tier HFA, the restarting cost of three-tier HFA is much lower than that of the two-tier HFA.

4.1 Pre-expanding the Lazy DFA

The idea is to expand the lazy DFA states in advance, according to the query set and the DTD of the XML stream. To avoid very high consumption of memory, we only pre-expand those DFA states for the shallow parts of XML data.

We need to specify the pre-expanding depth D_{pre} before pre-expanding. We construct a DFA which contains all related states for the elements with *location levels* no greater than D_{pre} . For example, the D_{pre} of the three-tier HFA shown in Fig.6 is 2.

However, we can not choose a great value of D_{pre} . Because when the D_{pre} is great, the number of DFA states for the pre-expanded DFA may be very large and the DFA consumes a lot of memory. In most cases, the value of D_{pre} is much smaller than D_{exp} .

The pre-expanded DFA construction takes several steps. We describe the process as follows:

- 1) Construct an NFA for the given query set. This process is the same as that described in YFilter [2, 4]. This NFA structure is available all the filtering time.
- 2) Extract all the paths of which the length is not greater than D_{pre} . For example, for the non-simple DTD shown in Fig. 2 (a), there are two paths in all: $\{/a/b, /a/c\}$ when $D_{pre}=2$.
- 3) Take every path as a serial input of elements and expand the DFA states in a similar way just like the lazy DFA.
- 4) After all the paths generated in Step 2 have been processed in Step 3, the construction is completed. Then, delete the NFA tables contained in DFA states except the initial DFA state, for these NFA tables consume a lot of memory. An example is shown in Fig. 7.

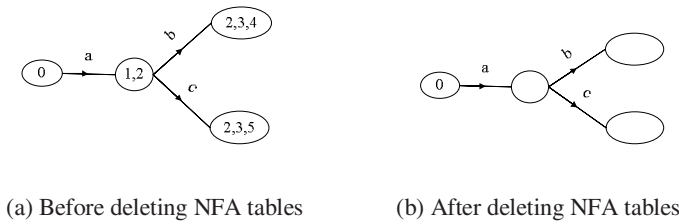


Fig. 7. The pre-expanded DFA for Q shown in Fig. 2(c) (with $D_{pre}=2$)

4.2 The Execution of HFA with Pre-expanded DFA

For the elements of which the *location levels* are greater than $D_{pre}+1$, the pre-processed HFA engine works the same as the two-tier HFA engine. We only discuss the processing of elements with *location levels* no greater than $D_{pre}+1$.

- 1) When processing elements of which *location levels* are smaller than $D_{pre}+1$,
 - a) If the event is “start-of-element”, HFilter pushes the target DFA state of the top DFA state into the runtime stack directly.
 - b) If the event is “end-of-element”, HFilter simply pops the top DFA state off the stack.
- 2) When processing elements of which the *location levels* equals to $D_{pre}+1$,
 - a) If the event is “start-of-element”, the target lazy DFA state does not exist at the first time. In order to compute the new lazy DFA state, the naive way is to compute it from the initial state. A full path from the root to the current element should be kept track of. After the lazy DFA state has been computed, HFilter will push this lazy DFA state into the runtime stack. Next time when HFilter processes the same lazy DFA state, HFilter need not compute it again.
 - b) If the event is “end-of-element”, HFilter simply pops off the top lazy DFA state.

4.3 Optimizing the Computing of New Lazy DFA States

With the pre-expanded DFA, for the elements with *location level* no greater than D_{pre} , we can obtain the target state from the DFA directly without additional computation.

For elements with a greater *location level*, the three-tier HFilter works almost the same as the two-tier HFilter. The only difference is when processing an element with a *location level* equal to $D_{pre}+1$, HFilter needs to compute the new lazy DFA state at the first time (as mentioned, we assume that $D_{pre} < D_{exp}$, therefore the new state must be a lazy DFA state and can not be an NFA state). Otherwise, the three-tier HFA works exactly like the two-tier HFA.

In order to compute the new lazy DFA state, as mentioned in Section 4.2, the naive way is to compute it from the initial state. However, the computing cost is very high.

To reduce the computing cost, we should compute the new lazy DFA state from an intermediate DFA state. For this DFA state, it needs to keep the NFA table. For example, we can keep the NFA tables of the DFA states of which the related *location levels* are $D_{pre}-1$. But we can not keep the NFA tables of the DFA states of which the related *location levels* are D_{pre} , because these DFA states are usually the main part of the DFA that we have pre-expanded. Keeping all the NFA tables of them will lead to a very high memory consumption. On the other hand, the cost of computing the new lazy DFA states will become higher when we choose a smaller value.

Therefore, we need to keep some NFA tables for computing new lazy DFA states. Here we assume that we keep all the NFA tables of the DFA states of which the related *location levels* are $D_{pre}-1$. Besides, we need to keep track of the partial path of which the *location levels* are from $D_{pre}-1$ to D_{pre} . With this partial path and the NFA table contained in the top DFA state on the runtime stack, we can compute those new lazy DFA states at the *location level* that equals to $D_{pre}+1$ and insert them into the HFA.

Example 4. Fig. 8 shows an example of this optimization. For this HFA, to process the path $\{a/b/c\}$ at the first time, it does not need to compute new lazy DFA states for elements $/a$ and $/b$. Because their *location levels* are no greater than D_{pre} . But it needs

to compute a new lazy DFA state when the “start-of-element” event of $/c$ is encountered. Given the partial path $\{b/c\}$ and the initial NFA table $\{1, 2\}$, the computing result is the NFA table $\{2, 5, 7\}$ and then the new lazy DFA state is obtained. This process takes two steps. However, it will take three steps without the optimization.

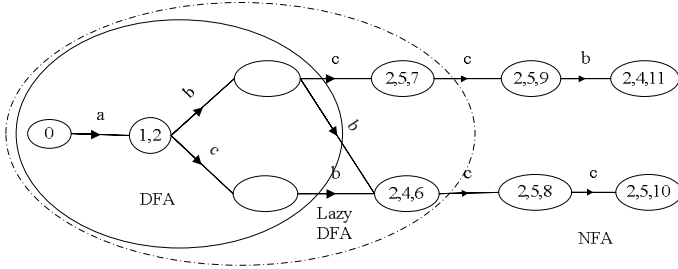


Fig. 8. The optimized three-tier HFA (with $D_{pre}=2$, $D_{exp}=3$)

4.4 Handling Memory Overflow

There are more chances of memory overflow for the three-tier HFA than the two-tier HFA, because the pre-expanding might lead to higher memory consumption during the filtering process. When the system runs out of memory, similar to the two-tier HFA, only the lazy DFA states expanded during the filtering process will be deleted. The pre-expanded DFA is memory-resident during the whole running time. Then, the lazy DFA tier can be re-expanded directly from the pre-expanded DFA tier and it achieves a higher filtering performance during the warming up phase.

5 Experiments

We compare HFilter with two popular XML filtering techniques—YFilter [2, 4] and lazy DFA [5]. Experiments of filtering performance are run on a synthetic data set: News Industry Text Format (NITF) DTD¹, and XML document sets are generated with IBM’s Generator tool [11]. We repeatedly filter each XML document set twice. All the experiments reported here are performed on a Celeron 2.66GHz processor with 1GB main memory running JVM 1.4.2 on Linux 2.6. We take the YFilter’s source code² to implement the NFA-tier of HFilter.

When the filtering system runs out of memory, we need to release the memory consumed by HFilter. During the memory releasing process, we remove all the references of useless objects. Due to the unpredictable garbage collection process of the JVM, we also make some attempts to force the JVM to release most part of the unused memory that are still occupied by it. These efforts may cost some time. However, for different implementations of filtering systems, such as implemented in C++ or in Java, the memory releasing processes are greatly different from each other.

¹ Nitf-2-5.dtd. <http://www.nitf.org/nitfdocumentation/nitf-2-5.dtd>

² YFilter 1.0 release. [http://yfilter.cs.berkeley.edu/code release.htm](http://yfilter.cs.berkeley.edu/code%20release.htm)

On the other hand, some optimizations of the garbage collection process of the JVM will be of much help. Therefore, in our experiments we simply omit the costs of the memory releasing processes.

Workload parameters of our experiments are listed in Table 1. We choose the maximum depth of XML data set as the default value of maximum query depth of our testing query set. The default value of probability of “*” and “/” is set to be 10%. And as mentioned, the values of the maximum expanding depth D_{exp} and the pre-expanding depth D_{pre} of the HFA are specified according to apriori knowledge. In our experiments, we set the default values of D_{exp} and D_{pre} according to the experiment results. We only generate single path queries in our experiments.

Table 1. Workload parameters

Parameter	Range	Default Value	Description
N_Q	10k to 40k	20k	Number of queries
D_{avg}	11 to 15	13	Average depth of an XML data set
D_{exp}	13	13	Maximum expanding depth of HFA
D_{pre}	9	9	Pre-expanding depth of HFA

Note that, we performed all the experiments in a fixed physical memory environment which is 1 GB, but our two-tier HFA and three-tier HFA schemes can easily adapt to other low physical memory environments as we can adjust the values of D_{exp} and D_{pre} according to the runtime memory usage and the input XML stream. When the physical memory is larger, increasing the value of D_{pre} can further improve the performance of the warming up phase. Meanwhile, increasing the value of D_{exp} is beneficial as well due to the increase of the physical memory. In other words, our schemes have very good adaptability in different physical memory environments. In some extreme cases, for example, if the physical memory is too low to use the pure lazy DFA scheme, we can still adjust our HFA schemes with small enough values of D_{exp} and D_{pre} to adapt the special environment. When the values of both D_{exp} and D_{pre} decrease to zero, our HFA schemes will degrade to NFA scheme. In the contrary cases, if the physical memory is extremely large and can use the lazy DFA and HFA schemes without leading to the restarting of the filtering engine, our HFA schemes are still the best choice. This is because the parameter D_{pre} can be set to a particularly large value to ensure all the necessary DFA states can be pre-expanded and thus the filtering engine can reach its maximum filtering speed.

5.1 Results and Analysis

We analyze the performance of HFilter with varying D_{avg} and N_Q . The measures in Fig. 9 and Fig. 10 are the average throughputs and the restarting times, or in other words, the times of memory overflow after 2,000 documents filtered twice.

The effect of varying D_{avg} of different XML data sets on the performance of the four approaches is shown in Fig. 9 (a). The D_{avg} of the three XML data sets is 11, 13 and 15, respectively. The maximum depths of them are 16, 18 and 20 respectively. When the D_{avg} is small, the three-tier HFA(3T-HFA), two-tier HFA(2T-HFA) and lazy DFA based approaches have high throughputs, but the NFA based approach's

throughput is relatively low. As the D_{avg} increases, the performance of lazy DFA decreases quickly. That is because lazy DFA has high memory consumption with regard to deeper XML data, thus memory overflow happens frequently, degrading the filtering performance. On the other hand, as the D_{avg} increases, the performance of the three-tier HFA and two-tier HFA decrease slowly and become closer to the performance of NFA. It is because in our approaches, elements whose depth are greater than D_{exp} are processed by the NFA tier of the two HFA. Thus when the D_{avg} of the data set increases, more elements are processed by the NFA. However, elements with depth no greater than D_{exp} can still be efficiently processed with our approaches. Therefore our hybrid approaches still perform better than the NFA based approaches. On average, the throughputs of our hybrid approaches are more than 30% higher than non-hybrid approaches.

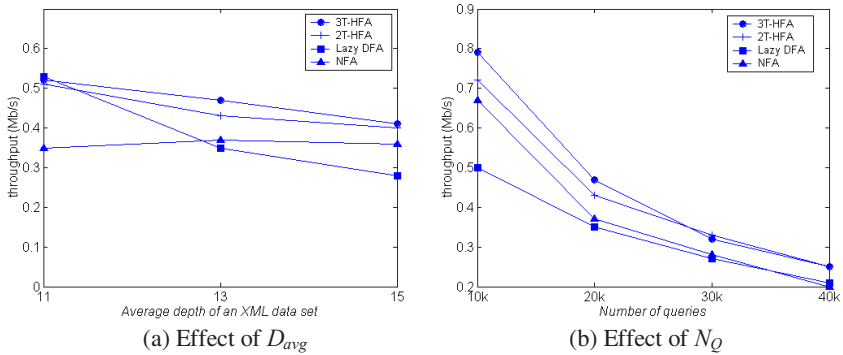


Fig. 9. Throughputs

Fig. 9 (b) depicts the effect of varying N_Q on the throughputs and restarting times. Although the throughputs of all the four schemes become lower as the N_Q increases, our HFA schemes still outperform the lazy DFA and the NFA. It is reasonable that the performance of all methods decreases as N_Q increases due to more frequent memory overflows. The throughputs of our hybrid approaches are about 24% higher than non-hybrid approaches on average. Attributed to the pre-expanding process, three-tier HFA performs better than two-tier HFA in most cases.

Fig. 10 (a) and (b) show the restarting times of the four schemes. The NFA scheme does not need to restart during the whole filtering process as it abandons the active states immediately after processed to keep the occupied memory small. Since the lazy DFA scheme remembers all the history active states in order to reuse those states efficiently, the amount of occupied memory may increase quickly when processing deep and recursive XML data. Therefore, the restarting happens frequently. Meanwhile, the restarting cost of the lazy DFA scheme is very high and it even performs worse than the NFA scheme when D_{avg} becomes larger. The two-tier HFA scheme restarts much fewer times compared with the lazy DFA scheme since its NFA tier saves a large amount of memory during filtering. Due to the pre-expanding process, the restarting of the three-tier DFA scheme happens the most frequently of all. Note that, after restarting, new states of the three-tier DFA that process deeper

elements of the XML data can be computed from the pre-expanded tier directly. Thus the restarting cost of the three-tier DFA scheme is very low and it still outperforms the two-tier DFA and lazy DFA schemes though it restarts more times.

D_{avg}	3T- HFA	2T- HFA	Lazy DFA	NFA
11	9	2	3	0
13	10	2	9	0
15	25	5	21	0

(a) Effect of D_{avg}

N_Q	3T- HFA	2T- HFA	Lazy DFA	NFA
10k	6	2	8	0
20k	10	2	9	0
30k	16	3	10	0
40k	30	4	13	0

(b) Effect of N_Q **Fig. 10.** Restarting times

6 Related Works

Automata based XML stream filtering techniques are widely studied in recent years. They can be classified into NFA based approaches and lazy DFA based approaches.

NFA based approaches. XFilter [1] is Finite State Machine (FSM) based approach. It builds a FSM for each path query. XQRL [9] and XScan [10] are FSM-based approaches too. YFilter [2, 4] is Nondeterministic Finite Automaton (NFA) based approach which constructs a single NFA to represent all XPath queries by sharing the common prefixes of the paths. [13] develops a filtering engine called YFilter* to enhance YFilter’s ability to handle nested path queries. It works efficiently with precision loss. Xtrie [3] divides queries into sub-strings that only contain parent-child (“/”) axis. All sub-strings are indexed by a trie-based data structure which takes advantage of the sharing of common sub-strings.

Lazy DFA based approaches. In [5], the filtering engine is Deterministic Finite Automaton (DFA) based and expanded lazily during the filtering process. It is more efficient than YFilter. Some optimizations of [5] are proposed in [7] and [8]. XPush Machine [6] is another similar system, which uses a single deterministic pushdown automaton to index all path queries. [12] studies cache-conscious technique to improve filtering performance, and applies this technique to DFA based approaches.

Above approaches are not suitable for processing deep and recursive XML data in low memory limitation environment.

7 Conclusions and Future Works

In this paper, we have proposed an XML stream filtering system, called HFilter, for processing deep and recursive XML data in low memory limitation environment. The basic HFilter is a two-tier Hybrid Finite Automaton (HFA) based approach, which combines the lazy DFA based approaches and the NFA based approaches. It processes the shallow parts of XML data using a lazy DFA and processes the deep

parts using an NFA. Therefore, the filtering process can achieve a higher throughput due to the usage of lazy DFA and data prefix sharing. On the other hand, the memory consumption decreases due to the usage of NFA.

We have also proposed a three-tier HFA approach with an extra pre-expanded DFA tier. It can accelerate the restarting process of HFilter when memory overflow happens, since the lazy DFA tier can be re-expanded directly from the DFA tier other than from the initial state.

Our experiments show that with multi-tier HFA, HFilter provides significant throughput improvement above 25% on average compared with existing approaches. The two-tier HFA reduces the frequency of memory overflow drastically and it outperforms the lazy DFA and the NFA. The three-tier HFA reduces the cost of restarting greatly. Although it also increase the frequency of memory overflow, it still performs better than the two-tier HFA in most cases due to the pre-expanded tier. The three-tier HFA can be re-expanded efficiently from the pre-expanded tier directly after restarting and has very low restarting cost. Therefore, the three-tier HFA schemes is the best one in our experiments.

In future, to enhance the flexibility of HFilter, we will focus on developing a self-adaptable mechanism which adjusts values of D_{exp} and D_{pre} dynamically with the change of data and queries.

Acknowledgments. This research is supported in part by the National Natural Science Foundation of China (NSFC) under grant 60503035, 60703093, the National High-Tech Research and Development Plan of China under Grant 2006AA01Z234 and SRF for ROCS, SEM.

References

1. Altinel, M., Franklin, M.: Efficient filtering of XML documents for selective dissemination of information. In: VLDB (2000)
2. Diao, Y., Fischer, P., Franklin, M., To, R.: YFilter: Efficient and Scalable Filtering of XML Documents. In: ICDE (2002)
3. Chan, C., Felber, P., Garofalakis, M.N., Rastogi, R.: Efficient filtering of XML documents with XPath expressions. In: ICDE (2002)
4. Diao, Y., Altinel, M., Franklin, M.J., Zhang, H., Fischer, P.: Path sharing and predicate evaluation for high-performance XML filtering. *ACM Trans. on Database Systems (TODS)* 28(4) (2003)
5. Green, T., Gupta, A., Miklau, G., Onizuka, M., Suciu, D.: Processing XML streams with deterministic automata and stream index. *ACM Trans. on Database Systems (TODS)* 29(4) (2004)
6. Gupta, A., Suciu, D.: Stream processing of XPath queries with predicates. In: SIGMOD (2003)
7. Onizuka, M.: Light-weight XPath processing of XML stream with deterministic automata. In: CIKM 2003 (2003)
8. Chen, D., Wong, R.: Optimizing The lazy DFA approach for XML stream processing. In: The Fifteenth Australasian Database Conference (ADC) (2004)
9. Florescu, D., Hillery, C., Kossmann, D., Lucas, P.: The BEA/XQRL streaming XQuery processor. In: VLDB 2003 (2003)

10. Ives, Z., Halevy, A., Weld, D.: An XML query engine for network-bound data. *VLDB Journal* 11(4) (2002)
11. Diaz, A.L., Lovell, D.: XML Generator, <http://www.alphaworks.ibm.com/tech/xmlgenerator>
12. He, B., Luo, Q., Choi, B.: Cache-conscious automata for XML filtering. In: *ICDE 2005* (2005)
13. Zhang, X., Yang, L., Lee, M., Hsu, W.: Scaling SDI systems via query clustering and aggregation. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) *DASFAA 2004*. LNCS, vol. 2973, pp. 208–219. Springer, Heidelberg (2004)

Read-Optimized, Cache-Conscious, Page Layouts for Temporal Relational Data

Khaled Jouini¹, Geneviève Jomier¹, and Patrick Kabore^{2,*}

¹ Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny 75775 Paris, France
`khaled.jouini@dauphine.fr`, `genevieve.jomier@dauphine.fr`

² ESI - Université Polytechnique, BP 1091 Bobo-Dioulasso, Burkina Faso
`kaborepatric@yahoo.fr`

Abstract. The efficient management of temporal data is crucial for many traditional and emerging database applications. A major performance bottleneck for database systems is the memory hierarchy. The performance of the memory hierarchy is directly related to how the content of disk pages maps to the L2 cache lines, *i.e.* to the organization of data within a page or the *page layout*. The prevalent page layout in database systems is the N-ary Storage Model (NSM). As demonstrated in this paper, using NSM for temporal data deteriorates memory hierarchy performance for query-intensive workloads. This paper proposes, new cache-conscious, read-optimized, page layouts specifically tailored for temporal data. The proposed page layouts optimize accesses to all levels of the memory hierarchy by avoiding fetching the same data several times (as opposed to NSM). Experiments show that the proposed page layouts are substantially faster than NSM.

1 Introduction

The earlier Relational DataBase Management Systems (RDBMS) were designed and optimized for On-Line Transaction Processing (OLTP) applications, running on platforms much different from those of today [1]. Despite the evolution over the last quarter century of application requirements and of hardware characteristics, the architecture of most current RDBMS remains essentially the same as that of earlier ones: *e.g.* tuple-oriented execution, B-tree indexing, etc. [2]. In recent papers [3,1,2], Stonebraker & al. argue that time has come to discard the "One Size Fits All" strategy (*i.e.* one system for all applications) followed by almost all leading RDBMS manufacturers and design systems more specialized and more adapted to current hardware characteristics. This especially holds for query-intensive applications that use ad-hoc queries to access large amounts of data. In contrast with OLTP-style applications, query-intensive applications require faster reads, while tolerating slower writes. Hence, they should be *read-optimized* [4]. Typical examples are decision support systems and electronic library card catalogs, where relatively long periods of ad-hoc queries are interspersed with periodic bulk-loading of new data [3,4].

* We would like to thank Claudia Bauzer Medeiros for many helpful discussions and reviews that improved this paper.

Database systems fetch data from non-volatile storage (*e.g.* disk) to processor in order to execute queries. Data goes through the memory hierarchy which consists of disk, main memory, L2 cache, L1 cache [5]. The communication between the main memory and the disk has been traditionally recognized as the dominant database performance bottleneck. However, architectural research on modern platforms has pointed out that the L2 cache miss penalty has an increasing impact on response times [6]. This *bottleneck shift* [7] is due to: (i) the increasing gap between processor and main memory speeds; (ii) the increase in main memory and in buffer pool sizes; and (iii) the development of hardware and software techniques hiding disk latency such as multithreading and prefetching [7, 8]. As a result, database systems should be designed to be sensitive, not only to disk and main memory performance, but also to L2 cache performance.

The basic unit of data transfer between disk and main memory is a *page*, typically 4 to 64 KB [9]. The L2 cache size is typically 2 to 4 MB [10]. The basic unit of data transfer between main memory and L2 cache is a *cache line*, typically 32 to 128 bytes [9]. The mapping of page content to cache lines is determined by the organization of data within the page, called the *page layout* [7]. Thus, the page layout highly impacts the memory hierarchy utilization of database systems [9]. The prevalent page layout in commercial RDBMS is the *N-ary Storage Model* (NSM), also called *row-store architecture* [4, 7, 9]. NSM stores tuples sequentially starting from the beginning of each disk page. While NSM provides a generic platform for a wide range of data storage needs, recent studies demonstrate that it exhibits poor memory hierarchy performance for query-intensive workloads [4]. Thus, alternative page layouts have been recently implemented in a number of academic and commercial read-optimized systems: *e.g.* Sybase IQ [11], senSage [12], VectorNova [13], Monet [14], Google BigTable [15], Vertica [4, 16], etc.. However, although temporal data constitute the core of many traditional (*e.g.* financial) and emerging (*e.g.* biodiversity) applications, in these systems no special attention was given to temporal data. This paper proposes two new read-optimized, cache-conscious, page layouts for temporal data. The aim of the proposed page layouts is to take into account the characteristics of modern platforms and to simultaneously achieve: (i) reasonable performance for writes; and (ii) high-performance reads.

The remainder of this paper is organized as follows. Section 2 defines the main concepts and illustrates the use of standard storage models for temporal data. Section 3.2 introduces our read-optimized storage policies. Section 4 reviews related work. Section 5 compares the performance of the different storage models. Section 6 concludes the paper.

2 Context and Problem Specification

2.1 Conceptual Model of a Temporal Relation

In temporal databases, in order to keep past, whenever a modeled entity e is modified, its old version is retained and a new version of e is created. Thus, an entity e may have in a single temporal relation, a set of associated tuples;

e	t	ssnum	name	salary	dept
e1	t1	323	John	10k	Math
e1	t3	323	John	10k	Info
e1	t5	323	John	12k	Info
e2	t2	854	Paul	20k	Eco
e2	t4	854	Paul	21k	Eco

ts	e	ts	t	ts	ssnum	ts	name	ts	salary	ts	dept
1	e1	1	t1	1	323	1	John	1	10k	1	Math
2	e1	2	t3	2	323	2	John	2	10k	2	Info
3	e1	3	t5	3	323	3	John	3	12k	3	Info
4	e2	4	t2	4	854	4	Paul	4	20k	4	Eco
5	e2	5	t4	5	854	5	Paul	5	21k	5	Eco

e	t	ssnum
e1	t1	323
e2	t2	854

e	t	name
e1	t1	John
e2	t2	Paul

e	t	salary
e1	t1	10k
e1	t5	12k
e2	t2	20k
e2	t4	21k

e	t	dept
e1	t1	Math
e1	t3	Info
e2	t2	Eco
e2	t4	Eco

e	t	ssnum	name	salary	Dept
e1	t1	323	John	10k	Math
e1	t3	323	John	10k	Info
e1	t5	323	John	12k	Info
e2	t2	854	Paul	20k	Eco
e2	t4	854	Paul	21k	Eco

Fig. 1. (a) A sample temporal relation *employee* in NSM-style representation. (b) Straight-forward DSM; *ts*: tuple surrogate. (c) Temporal DSM. (d) PSP only stores values written in black. Values written in grey are implicit.

each one is associated to a timestamp t and records the state (or the version) of e at t . Figure 1.a depicts a sample temporal relation *employee* (*entity surrogate*, *timestamp*, *ssnum*, *name*, *salary*, *department*). In this example, as in the remainder of this paper, time is assumed to be linear and totally ordered: $t_i < t_{i+1}$.

Let t_i and t_j be two timestamps such that: (i) $t_i < t_j$; and (ii) the state of an entity e is modified at t_i and at t_j , but remains unchanged between them. As the state of e remains unchanged between t_i and t_j , it is recorded only once by the tuple identified by (e, t_i) . The tuple (e, t_i) is said to be *alive* or *valid* for each $t \in [t_i, t_j)$. In Fig. 1.a. John’s tuple (e_1, t_3) is alive for each $t \in [t_3, t_5)$.

2.2 N-ary Storage Model

NSM packs all the attributes of a tuple in consecutive memory addresses and stores tuples within a page sequentially [17]. Each tuple in an NSM page is preceded by a tuple header (TH) providing metadata about the tuple, such as the length of the tuple, offsets of variable-length values and other implementation-specific information [18]. Figure 2 depicts a typical tuple format. As illustrated in Fig. 3.a, each NSM page has a page header (PH) containing information such

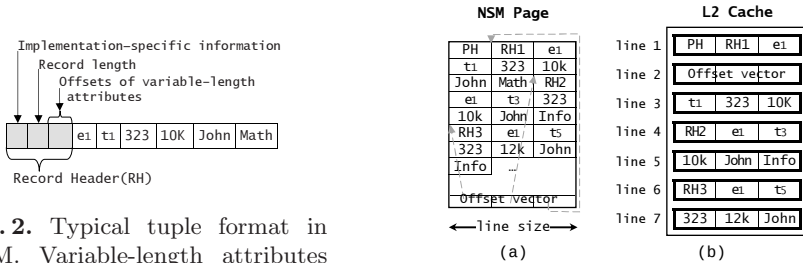


Fig. 2. Typical tuple format in NSM. Variable-length attributes are commonly placed at the end of the fixed-length attributes.

Fig. 3. (a) NSM page layout. (b) Cache behavior for the query "find John’s salary history".

as the page identifier and the total remaining free space. To locate tuples within a page, the starting offsets of tuples are kept in an *offset vector* [18]. Typically, the tuple space grows downwards while the offset vector grows upwards.

In most cases: (i) only a small fraction of the attributes of an entity e are time-varying; and (ii) time-varying attributes vary independently over time. With NSM, even if only one attribute is updated, all the other attributes are duplicated in the new tuple recording the new version of e . For example, in Fig. 1.a the update of John's department at t_3 leads to the replication of his ssnun, name and salary. We call this type of replication *version redundancy*. The important issue here is not the disk space consumed by version redundancy, as disk space costs virtually nothing nowadays [19]. The issue is that loading the memory hierarchy several times with the same data: (i) wastes disk and main memory bandwidths; (ii) pollutes the main memory and the L2 cache; (iii) "possibly forces replacement of information that may be needed in the future, incurring even more delays" [7,8]; and (iv) increases the amount of CPU cycles wasted in waiting for data loading.

Consider for instance the following query: "find John's salary history" and assume that the NSM page of Fig. 3.a is already in main memory and that the cache line size is smaller than the tuple size. As shown in Fig. 3.b, to execute the query, the page header and the offset vector are first loaded in the cache in order to locate John's tuples (cache lines 1 and 2). Next, each John tuple is loaded in the cache (cache lines 3 to 7). The tuple header and John's name and ssnun, which are useless for the query, are brought in the cache with each tuple, leading to the waste of main memory bandwidth, L2 cache space and CPU cycles.

2.3 Decomposition Storage Model

An alternative storage model to NSM is the *Decomposition Storage Model* (DSM) [17], also called *column-store architecture* [4]. As illustrated in Fig. 1.b, DSM partitions vertically a relation R with arity n , into n sub-relations. Each sub-relation holds: (1) the values of an attribute of R ; and (2) the tuple surrogates identifying the original tuples that the values came from. Each sub-relation is stored as a regular relation in an NSM page.

The trade-offs between DSM and NSM are still being explored [20,21,22]. The most cited strengths of DSM are: (i) *improved memory hierarchy utilization*: with DSM, a DBMS needs only read the values of attributes involved in a given query [23]; (ii) *improved data compression* [24]: as the values of an attribute are stored contiguously, DSM enables further compression opportunities. The most cited drawbacks of DSM are: (i) *increased seek time*: "Disk seeks between each read might be needed as multiple attributes are read in parallel" [21]; (ii) *increased cost of insertions*: DSM performs poorly for insertions because multiple distinct pages have to be updated for each inserted tuple [21]; (iii) *increased storage cost*: DSM stores a tuple header and a surrogate for each value; and (iv) *increased tuple reconstruction costs*: for queries involving several attributes, DSM needs to join the participating sub-relations together. In addition to the drawbacks aforementioned, using DSM for temporal data does not avoid version redundancy.

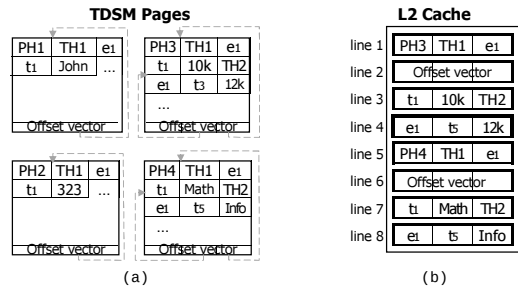


Fig. 4. (a) TDSM pages. (b) TDSM cache behavior for the query "find John's salary and department at t_4 ".

3 Read-Optimized, Cache-Conscious, Page Layouts

This section presents two new, cache-conscious, read-optimized page layouts, specifically tailored for temporal data: the Temporal Decomposition Storage Model (TDSM) and the Per Surrogate Partitioning Model (PSP). In the sequel, subsections 3.1 and 3.2 present TDSM and PSP. Subsection 3.3 discusses some performance issues.

3.1 Temporal Decomposition Storage Model (TDSM)

TDSM is a temporal extension of DSM. As illustrated in Fig. 1.c, TDSM does not store the timestamp attribute in a separate sub-relation as in DSM. Rather, the timestamp attribute is stored with each of the other attributes. With this approach, TDSM has the following advantages when compared to DSM: (i) it avoids version redundancy and hence improves memory hierarchy utilization; and (ii) it reduces the update cost when the attributes of an entity are updated, since only the pages storing the updated values are modified.

Although TDSM avoids version redundancy, it inherits some drawbacks from DSM. In particular, for queries involving several attributes, TDSM needs to join the participating sub-relations, not only on entity surrogate, but also on timestamp (*i.e.* temporal join [25]). Such a temporal join can be costly, especially as time-varying attributes generally vary independently [25].

Consider for instance the following query: "find John's salary and department at t_4 " and assume that the TDSM pages of Fig. 4.a are already in main memory. To execute the query, TDSM needs to load John's departments and salaries and then to join the loaded values on timestamps. As shown in Fig. 4.b, each value is loaded with a tuple header, an entity surrogate and a timestamp, leading to the waste of useful cache space and main memory bandwidth. Joining the loaded values on timestamp consumes additional CPU cycles.

3.2 Per Surrogate Partitioning Model (PSP)

As illustrated in Fig. 1.d, the goal of PSP is: (i) to store each information only once; and (ii) to allow easy tuple reconstructions. In order to allow easy tuple

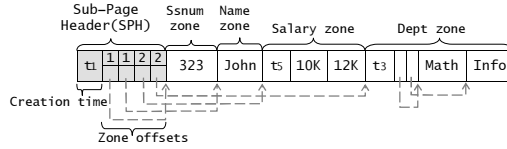


Fig. 5. Sub-page layout

reconstructions, PSP keeps all the attribute values of a tuple in the same page, as in NSM. However, unlike NSM, PSP organizes attribute values within a page, so that, redundant values are stored only once.

Within a page, PSP packs tuples into *sub-pages*, so that tuples of distinct sub-pages have distinct entity surrogates and tuples of any single sub-page have the same entity surrogate. Thus, a sub-page records the *history* of an entity. Within a sub-page, PSP packs the values of an attribute contiguously in an *attribute zone*. Thus, each sub-page is partitioned into n attribute zones, where n is the arity of the temporal relation. To sum up, PSP partitions a page horizontally in sub-pages and each sub-page vertically in attribute zones. The remainder of this section details the design of PSP and discusses the case where the whole history of an entity does not fit in a single page.

Attribute Zone. Let s be a sub-page recording the history of an entity e . An attribute zone is an area within s , storing the history of an attribute a of e : the values taken by a over time. For instance, in Fig. 5, the zone of John’s salary, stores together its successive amounts. Each attribute zone of an attribute a is prefaced by a timestamp vector holding the timestamps of updates on a . The values of a are put at the end of the timestamp vector in the same order as timestamps. When a variable-length attribute is also time-varying, its values are preceded by an offset vector as in the dept zone in Fig. 5.

Let t_c be the lowest timestamp identifying a tuple recording a state of an entity e ; t_c is called the *creation time* of e : e.g. in Fig. 1.a the creation time of entity John is t_1 . To avoid redundancy, t_c is not stored in the timestamp vector of each attribute zone; rather, t_c is stored only once at the sub-page level. Thus, if an attribute zone stores a single value, its timestamp vector is empty. For example, in Fig. 5, the timestamp vector of John’s ssnun zone is empty.

Searching for the current value of an attribute (*i.e.* the most recent one) within a zone is done by a simple offset computation: the current value is always stored at the last position. To find an historical value (*i.e.* non current) valid at a timestamp t , the timestamp vector (and if necessary the offsets of variable length values) is first loaded. Within the timestamp vector, PSP looks for the position i of the greatest timestamp t' lower than t . Finally, the value stored at the position i within the zone is read.

Sub-Page and Page Layouts. As indicated before, a sub-page stores the whole history of an entity e . As shown in Fig. 5, each sub-page is preceded by a sub-page header (SPH) containing: the creation time of e and a vector of pairs (z, v) ,

where z is the starting offset of the zone corresponding to an attribute a of e and v is the number of a distinct values.

As an entity may have tens or even hundreds of versions, a vector of sub-page offsets in PSP is expected to be much smaller than a vector of tuple offsets in NSM. Thus, for PSP it makes sense to store the page header and the offset vector contiguously, so that, loading the page header also loads the offset vector or a large part of it. Such a design avoids an indirection when searching for sub-pages within a page. As illustrated in Fig. 6.a, in a PSP page, the sub-page space grows upwards while the offset vector grows downwards.

Large Sub-Pages. Situations occur where the whole history of an entity does not fit in a single page, *i.e.* a sub-page is too large to fit in a single page. PSP copes with this large sub-page problem, as follows. Let s be a large sub-page storing the history of an entity e and t be the time of the last update of e . The solution consists in creating a new sub-page s' and initializing it with the version of e valid at t . This solution introduces some (limited) redundancy but has an important advantage: the search for e tuples alive at a timestamp t_i , such that $t_i \geq t$, is only performed within s' and the search for e tuples alive at a timestamp t_j , such that $t_j < t$, is only performed within s .

Timestamp Compression. Data compression allows to "trade CPU cycles, which are abundant, for disk and main memory bandwidths, which are not" [4]. By clustering timestamps and values for each attribute together, PSP enables further compression opportunities than NSM. At the current state of PSP implementation, we mainly focus on timestamp compression, using two common lightweight compression schemes: *Dictionary* and *Delta* [26, 27]. The dictionary scheme creates an array storing all distinct timestamps. Then, within each attribute zone, each timestamp is replaced by its position within the array. The delta scheme is used to exploit the fact that references to timestamps in an attribute zone are sorted and stored contiguously. With the delta scheme, each reference is stored as a delta from the previous one.

3.3 Discussion

Consider the query "find John's salary history". As shown in Fig. 6.b, PSP improves the cache space and the main memory bandwidth consumed by this query, because it avoids fetching the same value several times (as opposed to DSM and NSM). In addition, PSP improves the data spatial locality, because the requested values are stored contiguously. PSP also requires less storage space than NSM, because: (i) it stores unchanged values only once; and (ii) it factorizes common entity metadata, whereas NSM stores a header for each tuple. Consequently, a PSP page is expected to contain more data than the corresponding NSM page. Thus, PSP also improves disk bandwidth and main memory space utilization. The side effect of the compactness of PSP is the additional effort needed for data replacement during modification operations.

In case of time-invariant data, each entity has a single version. Thus, each attribute zone within a PSP page stores a single value and has an empty timestamp

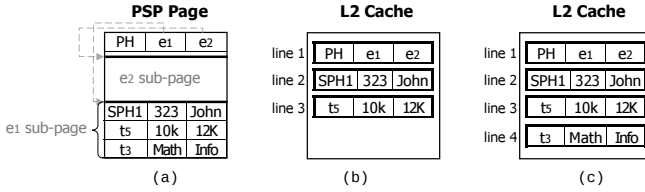


Fig. 6. (a) PSP page layout. (b) Cache behavior for the query "find John's salary history". (c) Cache behavior for the query "find John's salary and department at t_4 ".

vector. Consequently, a PSP sub-page has a layout similar to a typical tuple format in NSM. As a result, a PSP page and an NSM page have similar layouts and behaviors when used for non temporal data.

Consider the query "find John's salary and department at t_4 ". As shown in Fig. 6.c, to reconstruct a tuple value, PSP only needs to perform joins among attribute zones stored contiguously within a single (sub-) page, without searching outside the page (as opposed to DSM and TDSM).

4 Related Work

Several approaches have been recently proposed in order to achieve high performance for read operations. This section focuses on approaches that use page layouts different from NSM. Other approaches that focus on buffer pool management and on cache-conscious algorithms are less closely related.

In [28], [7] and [9] cache-conscious page layouts have been proposed, namely Data Morphing, Partition Attributes Across (PAX) and Clotho. Among these, PAX is the closest to PSP. Given a relation R with arity n , PAX partitions each page into n *minipages*. The i^{th} minipage stores all the values of the i^{th} attribute of R . With this approach, PAX provides a high degree of spatial locality for sequential access to values of one attribute. Nevertheless, PAX stores data in entry sequence to achieve good performance for updates. Thus, using PAX for temporal data does not avoid version redundancy.

A number of academic and commercial read-optimized systems implement DSM: Sybase IQ [11], Fractured Mirrors [19], Monet [14], C-Store [4, 16], etc.. These systems reduce the tuple reconstruction cost of DSM using techniques such as join indexes and chunk-based reconstructions [19, 4]. However, except C-Store and Fractured Mirrors, as they store data in entry sequence order, their performance suffer from the same problems as NSM.

Fractured Mirrors [19] stores two copies of a relation, one using DSM and the other using NSM. The read requests generated during query execution are appropriately scheduled between mirrors. With this approach, Fractured Mirrors provides better query performance than either storage model can provide separately. However, as Fractured Mirrors have not been designed to handle temporal data the problem of version redundancy has not been considered.

C-Store [4] implements a relation as a collection of materialized overlapping projections (*i.e.* each attribute appears in as many projections as needed). The implemented projections are determined according to frequent queries; a training workload is created for this purpose. To achieve high performance for query-intensive workloads, C-Store: (*i*) sorts projections from the same relation on different attributes; (*ii*) allows a projection from a given relation to contain any number of other attributes from other relations (*i.e.* pre-joins); (*iii*) stores projections using DSM; (*iv*) uses join indexes to reconstruct tuples; and (*v*) uses compression to avoid an explosion in storage space. The implementation of C-Store implicitly assumes that projections from the same relation have the same number of tuples. Thus, C-Store is unable to avoid version redundancy.

5 Performance Evaluation

This section compares the performance of the different storage models. For vertical decomposition, as TDSM is expected to outperform the straight-forward DSM and due to the lack of space, only TDSM is compared to NSM and PSP.

NSM and DSM systems often use their own sets of query techniques that can provide additional performance improvements [20]. As the present paper only focuses on the differences between NSM, TDSM and PSP related to the way data are stored in pages, we have implemented a TDSM, an NSM and a PSP storage manager, in C++ from scratch. The performance of these storage managers is measured with identical datasets and query workloads, generated following the specifications of the cost models presented in [29, 30].

In order to make temporal joins easier and not to penalize vertical decomposition, the different storage models are compared in the context of clustering index. The implemented storage managers use the Time Split B-tree (TSB-tree) [31], a bi-dimensional index structure commonly used for temporal data. In the sequel, subsection 5.1 reviews the TSB-tree and presents its implementation with PSP. Subsection 5.2 evaluates the performance of the storage models.

5.1 Time Split B-Tree (TSB-Tree)

Standard TSB-Tree. The TSB-tree is a variant of the B+tree. Leaf nodes contain data and are called *data pages*. Non-leaf nodes, called *index pages*, direct search from the root and contain only search information. At a given level, TSB-tree pages partition the surrogate-time space. An entry of an index page is a

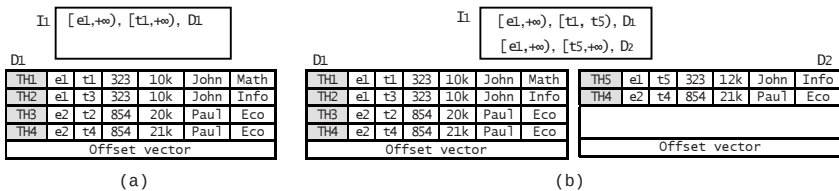


Fig. 7. (a) Initial TSB-tree on top of PSP. (b) Time split of D_1 at t_5 .

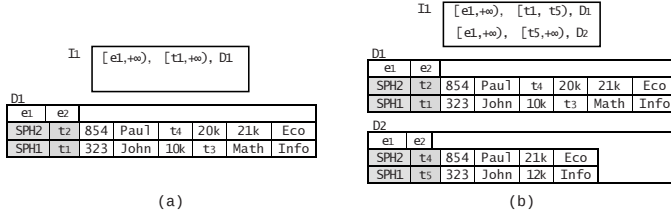


Fig. 8. (a) Initial TSB-tree on top of PSP. (b) Time split of D_1 at t_5 .

triplet $([e_{min}, e_{max}), [t_{start}, t_{end}), I)$, where $[e_{min}, e_{max})$ is a surrogate interval, $[t_{start}, t_{end})$ is a time interval and I the identifier of a child page. Such entry indicates that the data pages of the subtree rooted at I contain tuples (e, t) , such that $e \in [e_{min}, e_{max})$ and $t \in [t_{start}, t_{end})$. I is said to be valid for each timestamp $t \in [t_{start}, t_{end})$.

Tuples within a data page are ordered by entity surrogate and then by timestamp. If the insertion of a tuple causes a data page overflow, the TSB-tree uses either, *time split*, *surrogate split* or a combination of both. A surrogate split occurs when the overflowing data page only contains current tuples. It is similar to a split in a B+tree: tuples with surrogate greater than or equal to the split surrogate are moved to the newly allocated data page. A time split occurs when the overflowing data page, D , contains both current and historical tuples. The time split of D separates its tuples according to the current time t : (1) the t_{end} of D is set to t ; (2) a new data page D' with time interval $[t, +\infty)$ is allocated; (3) tuples of D valid at t are copied in D' (Fig. 7.b). After a time split, if the number of tuples copied in D' exceeds a threshold θ , a surrogate split of D' is performed. An index page split is similar to a data page split.

TSB-Tree with PSP. The split of a PSP page is performed as follows. Let D and D' be, respectively, the overflowing and the newly allocated data pages.

Surrogate Split: the sub-pages of D with surrogate greater than or equal to the split surrogate are moved to D' .

Time Split: (1) for each sub-page s of D , a sub-page s' with the same surrogate is created in D' ; (2) each attribute zone in s' is initialized by the most recent value from the corresponding attribute zone of s ; and (3) the largest timestamp appearing in an attribute zone of s , is copied in the header of s' (Fig. 8.b).

5.2 Assumptions, Settings and Results

Workload and Assumptions. The cost models proposed in [30, 29] model a temporal relation R by a set of E entities and T timestamps. Each entity e is subject to updates; each update occurring at timestamp t , generates a new entity version (e, t) , whose value is recorded in a tuple of R . The proportion δ of entities updated at each timestamp, called *data agility* [29], is assumed to be constant. Thus, the total number of tuples in R is: $E + \delta E(T - 1)$. R is assumed to be indexed by TSB-trees, using NSM, TDSM or PSP. The goal is to evaluate the storage, insertion and query costs. The storage cost is measured

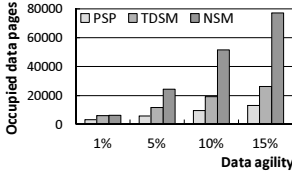


Fig. 9. Storage cost as function of data agility

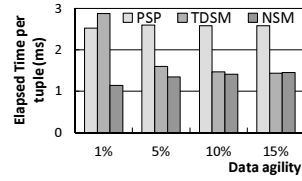


Fig. 10. Average elapsed time per tuple as function of data agility

by the number of occupied data pages. The insertion cost is measured by the average time elapsed during the insertion of a tuple. The cost of a query q is measured by three parameters: (i) the average number of data pages loaded in main memory; (ii) the average number of L2 cache misses when the requested pages are main-memory resident; and (iii) the execution time when the requested pages are main-memory resident. To be as general as possible, we follow [29] and assume that a query q has the following form:

select q_a attributes **from** R **where** $e \in [e_i, e_j)$ **and** $t \in [t_k, t_l)$

where q_a is the number of involved time-varying attributes, $[e_i, e_j)$ an interval of entity surrogates containing q_s consecutive surrogates and $[t_k, t_l)$ a time interval containing q_t consecutive timestamps.

Settings and Measurement Tools. A large number of simulations have been performed to compare NSM, TDSM and PSP. However, due to the lack of space, only few results are presented herein. For the presented simulations, data are generated as follows. A temporal relation R is assumed to have ten 4-byte numeric attributes, in addition to an entity surrogate and a timestamp. Four attributes of R are time-varying. Time-varying attributes are assumed to vary independently over time, with the same agility. E and T are respectively set to 200K entities and 200 timestamps. At the first timestamp, 200K tuples are inserted in R (one tuple per entity). Then, at each of the following 199 timestamps, δE entities, randomly selected, are updated. The data agility δ is varied in order to obtain different temporal relations.

Simulations are performed on a dedicated dual core 2.80 GHz Pentium system, running MS Windows Server 2003. This computer features 1GB main memory (DDR2-667MHz), 800 MHz FSB, and 2×2 MB L2 cache. The cache line size is 64B. The storage managers were configured to use a 8KB page size. The execution time is measured by the function *QueryPerformanceCounter* provided by the API Win32. The L2 cache related events are collected using Intel VTune.

Results

Storage Cost. Figure 9 depicts the storage costs as function of the data agility. PSP requires up to ≈ 7.4 times less storage space than NSM and ≈ 2 times less storage space than TDSM. The superiority of PSP against NSM increases as the agility increases, because, higher is the agility and higher is the number of tuples per entity, hence, higher is the disk space saved by PSP.

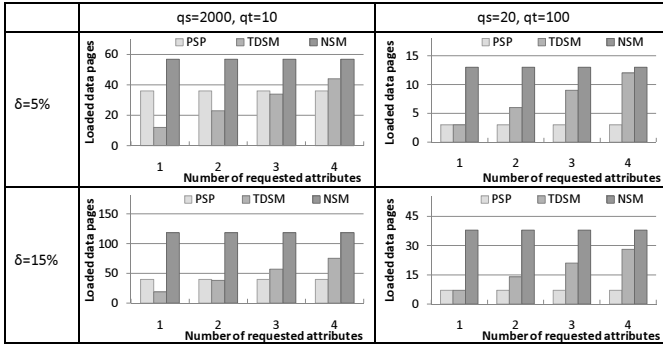


Fig. 11. Average loaded disk pages as function of the number of requested attributes

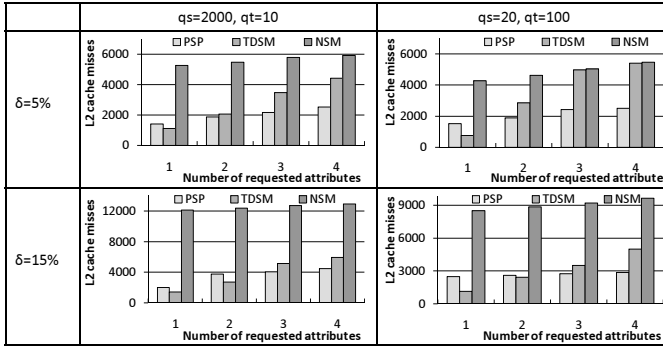


Fig. 12. Average L2 cache misses as function of the number of requested attributes

Insertion Cost. To provide fair comparison of insertion costs, we assume that all data requests are served from disk. Figure 10 depicts the insertion costs as function of data agility. Insertions in NSM are on average ≈ 2 times faster than in PSP. The main reason is that with NSM a single write suffices to push all the attributes of a tuple, while with PSP, an additional effort is needed for sub-page reorganizations. Although TDSM performance are penalized by the fact that the operating system scheduler handles write requests for multiple relations, TDSM exhibits good performance. Note that TDSM performance should be worst if more than one attribute is updated at a time.

Query Cost. To evaluate the query cost, we consider a moderately agile temporal relation, $\delta = 5\%$, and a highly agile temporal relation, $\delta = 15\%$. For each temporal relation, eight query workloads are considered. Each query workload consists of 100 queries. Queries in a workload involve the same of number of consecutive entity surrogates, q_s , the same number of consecutive timestamps, q_t , and the same number of temporal attributes, q_a . The surrogate intervals and the time intervals of queries of a given workload are uniformly distributed in the surrogate-timestamp space. The reported results for a given workload are

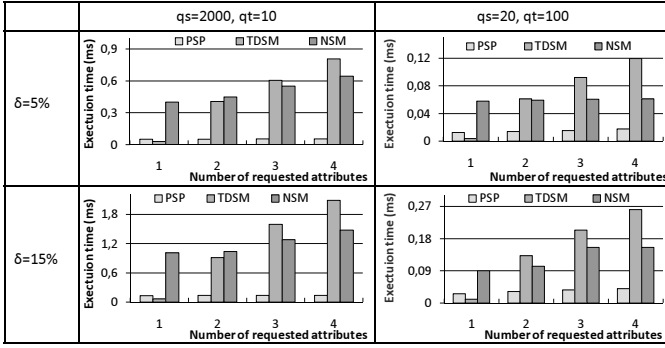


Fig. 13. Average execution time as function of the number of requested attributes

the average of performance achieved by the 100 queries composing it. The first four query workloads involve a relatively large surrogate interval, $q_s = 2000$, a small time interval, $q_t = 10$, and different numbers of temporal attributes: q_a varies from 1 to 4. The latter four query workloads involve a small surrogate interval, $q_s = 20$, a relatively large time interval, $q_t = 100$, and different numbers of temporal attributes: q_a varies from 1 to 4.

Figure 11 depicts the average numbers of data pages loaded in main memory to answer the query workloads described above. As shown in Fig. 11, PSP and TDSM outperform NSM in all cases. In general, TDSM has better behavior than PSP only when few attributes are involved. Figure 12 depicts the average numbers of L2 cache misses, when the requested pages are main-memory resident. As shown in Fig. 12, in all cases, PSP and TDSM generate less L2 cache misses than NSM. In particular, PSP generates up to 9 times less L2 cache misses than NSM. Figure 13 depicts the average execution time, when the requested pages are main-memory resident. TDSM has a better behavior than PSP only when a single attribute is involved. TDSM performance deteriorates very quickly as q_a , the number of involved attributes in a query, increases. This is due to temporal joins. As shown in Fig. 13, PSP is faster than either NSM and TDSM: on average ≈ 6.54 times faster than NSM and ≈ 6.18 faster than TDSM, when data are already in main memory. Note that the performance gap between PSP and NSM should be higher if data were fetched from disk rather than from main memory.

In summary, PSP improves performance by: (i) reducing disk seek and transfer times (less data are loaded in the memory hierarchy); (ii) increasing buffer and cache hit rate (a larger fraction of data fits in buffer pool and in cache lines); and (iii) increasing CPU utilization (less CPU cycles are wasted in waiting for data loading or in tuple reconstructions).

6 Conclusion

The memory hierarchy utilization is highly dependent on the page layout. This paper proposes TDSM and PSP, two new read-optimized, cache-conscious, page

layouts specifically tailored for temporal data. PSP optimizes performance at all levels of the memory hierarchy by combining the advantages of TDSM and NSM while avoiding their respective drawbacks: (i) PSP avoids version redundancy (as opposed to NSM); and (ii) PSP allows easy tuple reconstructions, *i.e.* without performing costly temporal joins (as opposed to TDSM). In addition to the advantages aforementioned, PSP can be used for non temporal data with the same performance as NSM.

Although we implemented a straight-forward TDSM, TDSM exhibits interesting features that need to be further explored. In particular, TDSM performance can be substantially improved if join techniques, more adapted to vertical partitioning than those used for temporal data, are designed: it is our next goal!

References

1. Stonebraker, M., Bear, C., Çetintemel, U., Cherniack, M.: *Al: One Size Fits All? Part 2: Benchmarking Studies*. In: CIDR, pp. 173–184 (2007)
2. Stonebraker, M., Madden, S., Abadi, D., Harizopoulos, S.: *Al: The End of an Architectural Era (it's Time for a Complete Rewrite)*. In: VLDB, pp. 1150–1160 (2007)
3. Stonebraker, M., Çetintemel, U.: “One Size Fits All”: An Idea Whose Time Has Come and Gone. In: ICDE 2005, pp. 2–11. IEEE Computer Society, Los Alamitos (2005)
4. Stonebraker, M., Abadi, D.J., Batkin, A., Chen, X.: *Al: C-store: A Column-Oriented DBMS*. In: VLDB, pp. 553–564 (2005)
5. Baykan, E.: Recent Research on Database System Performance. Technical report, LBD-Ecole Polytechnique Fédérale de Lausanne (2005)
6. Ailamaki, A., DeWitt, D., Hill, M., Wood, D.: *DBMSs on a Modern Processor: Where Does Time Go?* In: VLDB, pp. 266–277. Morgan Kaufmann, San Francisco (1999)
7. Ailamaki, A., DeWitt, D., Hill, M., Skounakis, M.: *Weaving Relations for Cache Performance*. In: VLDB, pp. 169–180 (2001)
8. He, Z., Marquez, A.: *Path and Cache Conscious Prefetching (PCCP)*. The VLDB Journal 16(2), 235–249 (2007)
9. Shao, M., Schindler, J., Schlosser, S.W., Ailamaki, A.: *Al: Clotho: Decoupling Memory Page Layout from Storage Organization*. In: VLDB, pp. 696–707 (2004)
10. Ailamaki, A., Govindaraju, N.K., Manocha, D.: *Query Co-Processing on Commodity Hardware*. In: ICDE, p. 107 (2006)
11. SybaseIq, <http://www.sybase.com/>
12. SenSage, <http://www.sensage.com/>
13. VectorNova, <http://www.vectornova.com/>
14. Boncz, P.A., Zukowski, M., Nes, N.: *MonetDB/X100: Hyper-Pipelining Query Execution*. In: CIDR, pp. 225–237 (2005)
15. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.: *Al: Bigtable: a Distributed Storage System for Structured Data*. In: OSDI 2006, Berkeley, CA, USA, pp. 205–218. USENIX Association (2006)
16. Vertica, http://www.vertica.com/v-zone/product_documentation
17. Copeland, G.P., Khoshafian, S.: *A Decomposition Storage Model*. In: Navathe, S.B. (ed.) ACM SIGMOD 1985, pp. 268–279. ACM Press, New York (1985)

18. Ramakrishnan, R., Gehrke, J.: 3 Data Storage and Indexing. In: Database Management Systems. McGraw-Hill, New York (2000)
19. Ramamurthy, R., DeWitt, D.J., Su, Q.: A Case for Fractured Mirrors. *The VLDB Journal* 12(2), 89–101 (2003)
20. Harizopoulos, S., Liang, V., Abadi, D., Madden, S.: Performance Tradeoffs in Read-Optimized DataBases. In: *VLDB*, pp. 487–498 (2006)
21. Abadi, D.: Column Stores for Wide and Sparse Data. In: *CIDR*, pp. 292–297 (2007)
22. Abadi, D.J., Madden, S., Hachem, N.: Column-Stores vs. Row-Stores: How Different Are They Really? In: *SIGMOD*, Vancouver, Canada (to appear, 2008)
23. Ailamaki, A., DeWitt, D., Hill, M.: Data Page Layouts for Relational Databases on Deep Memory Hierarchies. *The VLDB Journal* 11(3), 198–215 (2002)
24. Abadi, D., Madden, S., Ferreira, M.: Integrating Compression and Execution in Column-Oriented Database Systems. In: *ACM SIGMOD 2006*, pp. 671–682 (2006)
25. Donghui, Z., Tsotras, V., Seeger, B.: Efficient Temporal Join Processing Using Indices. In: *ICDE 2002*, p. 103. IEEE Computer Society, Los Alamitos (2002)
26. Roth, M., Horn, S.V.: Database Compression. *SIGMOD Rec.* 22(3), 31–39 (1993)
27. Raman, V., Swart, G.: How to Wring a Table Dry: Entropy Compression of Relations and Querying of Compressed Relations. In: *VLDB*, pp. 858–869 (2006)
28. Hankins, R.A., Patel, J.M.: Data Morphing: An Adaptive, Cache-Conscious Storage Technique. In: *VLDB*, pp. 417–428 (2003)
29. Tao, Y., Papadias, D., Zhang, J.: Cost Models for Overlapping and Multiversion structures. *ACM Trans. Database Syst.* 27(3), 299–342 (2002)
30. Jouini, K., Jomier, G.: Indexing Multiversion Databases. In: *ACM Sixteenth Conference on Information and Knowledge Management CIKM 2007* (November 2007)
31. Lomet, D.B., Salzberg, B.: The Performance of a Multiversion Access Method. In: *ACM SIGMOD 1990*, May 1990, pp. 353–363. ACM Press, New York (1990)

Exploiting Interactions among Query Rewrite Rules in the Teradata DBMS

Ahmad Ghazal, Dawit Seid, Alain Crolotte, and Bill McKenna*

Teradata Corporation
100 N. Sepulveda Blvd. El Segundo, CA, 90245
{ahmad.ghazal,dawit.seid,alain.crolotte,
bill.mckenna}@teradata.com

Abstract. Query rewrite (QRW) optimizations apply algebraic transformations to a SQL query Q producing a SQL query Q' . Both Q and Q' are semantically equivalent (i.e. they produce the same result) but the execution of Q' is generally faster than that of Q . Folding views/derived tables, applying transitive closure on predicates, and converting outer joins to inner joins are some examples of QRW optimizations. In this paper, we carefully analyze the interactions among a number of rewrite rules and show how this knowledge is used to devise a triggering mechanism in the new Teradata extensible QRW subsystem thereby enabling efficient application of the rewrite rules. We also present results from experimental studies that show that, as compared to a conventional recognize-act cycle strategy, exploiting these interactions yields significant reduction in the time and space cost of query optimization while producing the same re-written queries.

1 Introduction and Problem Definition

The growing sophistication of business intelligence applications developed on top of relational DBMSs is tremendously increasing the complexity of automatically generated queries posed to the DBMS. Nowadays, it is not unusual to see SQL queries that span hundreds of lines. A common feature of these SQL queries is that they are inefficiently composed as compared to, for example, a semantically equivalent query that may be produced by a knowledgeable SQL programmer. This is partly the result of multiple layers of abstraction that are required by the applications. For example, in data warehousing, numerous views and derived tables are used for security, for implementing business rules and semantic layers and for capturing complex intermediate results. Queries over these views tend to be very complex. For example, users tend to define views that project all fields of the underlying tables and then implement applications that generate queries that project only a subset of those fields.

In general, there are two main techniques to optimize queries in a traditional DBMS: logical query rewrite and physical query plan optimization. Query rewrite (QRW) is a process of converting a given SQL query into a more simplified and mostly more efficient SQL query without changing its semantics. The physical optimizer typically uses a cost-based approach to perform join order planning and other types of query optimization.

* This work was completed while the author was employed at Teradata Corp. The author is now employed at ParAccel, Inc.

Query rewrite plays a critical role in optimizing the aforementioned types of complex queries which would normally pose tremendous challenges to a pure cost-based query optimizer that solely relies on enumerating various plans to choose the efficient one. QRW can be rule-based (heuristic), like predicate pushing, or cost-based, like rewriting a query to use a materialized view (join index in Teradata). Also, QRW can be applied *before* or *during* join planning or *after* join planning is complete. For example, group-by push down is done during join planning, while writing to a common buffer space is a rewrite that happens after join planning is done. In this paper we focus on rewriting queries before join planning and discuss the challenges in efficiently implementing these rewrites and our approach that exploits the triggering interactions among these rewrites.

A key challenge in implementing a query rewrite subsystem is determining how the numerous rewrite rules *interact*. As we will later show experimentally, it will not be efficient to reapply all rewrite rules every time we apply a particular rewrite. One of the ways to minimize this search space is to determine cases where a given rewrite triggers another rewrite. To the best of our knowledge, [1] is the only paper that presented a focused discussion of this issue for a limited set of query rewrite rules, namely subquery (view) merging, distinct pushdown/pull-up, INTERSECT to existential subquery conversion and common subexpression replication. In this paper we discuss interactions among a broader set of rewrite rules including projection push-down, view-folding, outer-to-inner join conversion, join elimination, predicate move-around, predicate satisfiability-and-transitive closure (SAT-TC) and set operation simplification.

Exploiting the interactions among an expanded set of rewrites is necessary, for example, to effectively simplify the complex queries involving layered views (derived tables) mentioned above. For instance, queries that project only a subset of fields from views that project all fields of the underlying tables provide opportunities to apply projection pushing into lower blocks which in turn may trigger join elimination as well. Merging derived tables or views is also an important rewrite that allows the optimizer to consider various other rewrites like outer join to inner join conversion and join elimination.

Another important challenge is handling cross-block triggering where rewrite of a particular SQL query block *b* may also enable the rewrite of a parent block of *b* or a sub-block of *b*. Hence, a particular query rewrite rule application to a given block has *three dimensions* of triggers, namely triggering other re-writes on the same block, re-writes on upper level blocks and re-writes on lower level blocks. For example, Predicate Pushing rewrite of a block triggers the same re-write on the block's sub-blocks whereas applying the view folding (view merging) rule on a block may trigger the join elimination re-write on the upper blocks. The upward/downward triggering relationship can also exist between different rewrite rules. For example, predicate push-down may trigger SAT-TC on a child block while outer-to-inner join conversion may trigger SAT-TC on the same block. We give a simple example of how these cross-block interactions are exploited to produce efficient queries using query "Q1" based on the TPC-H data model that we utilize throughout the document. The TPC-H data model is a retail model fully described in [16]. It consists of 8 tables with history over 8 years. We are presenting examples using mainly Ordertbl (primary key (PK) =o_orderkey) containing the orders, Lineitem (PK=l_orderkey, l_linenumber)

containing the line items associated to these orders and Customer (PK=c_custkey) contains customers placing orders. The field names in these tables are self-describing. By convention fields in *Ordertbl* are preceded by “o_” while fields in *Lineitem* are preceded by “l_” and fields in *Customer* by “c_”. There is a FK-PK relationship between *l_orderkey* and *o_orderkey* and a FK-PK relationship between *o_custkey* and *c_custkey* and some customers do not have orders.

```
Q1:SELECT DT.x, DT.y
      FROM ( SELECT l_orderkey, l_shipdate, l_comment
            FROM Lineitem
            WHERE extract(month from l_shipdate) <= 6
            UNION ALL
            SELECT l_orderkey, l_shipdate, l_comment
            FROM Lineitem
            WHERE EXTRACT(MONTH FROM l_shipdate) >= 7
      ) DT(x,y,z), Ordertbl O
      WHERE y = '1995-12-24' AND O.o_orderkey=DT.x;
```

During the first re-write, projection pushing into the derived table *DT* removes the field *z*. Also, predicate *y = '1995-12-24'* can be pushed into each UNION ALL branch. Pushing in *y = '1995-12-24'* then triggers SAT-TC to be applied on each UNION ALL branch. Applying SAT-TC on each UNION ALL branch finds the first branch unsatisfiable since *l_shipdate = '1995-12-24' AND extract(month from l_shipdate) <= 6* is FALSE for all values of *l_shipdate*. This triggers UNION ALL branch elimination, which removes unsatisfiable UNION ALL branches, to remove the first branch. Now, since only one block remains in the sub-query, view (derived table) folding rewrite becomes possible. The application of this rule will merge the inner block to the outer block resulting in the following simplified query:

```
SELECT l_orderkey, l_shipdate FROM Lineitem, Ordertbl O WHERE
EXTRACT(MONTH FROM l_shipdate) >= 7 AND l_shipdate='1995-12-24' AND
l_orderkey = O.o_orderkey;
```

To perform the above kinds of re-writes, the query rewrite system needs to exploit the *order independence property* of the rewrite rules and apply them in multiple iterations. For instance, in the above example, even if we applied SAT-TC first followed by Predicate Pushdown, the rewrite system should still be able to eventually achieve the same effect as above. A rewrite system that applies the rules in some fixed order will miss valid rewrites. In order to retain order independence of rewrite rules, a rewrite system needs a way to prevent those rewrite rules that remove predicates or query blocks from restricting other rewrite rules. In this paper we resolve this problem by adopting an approach that performs redundant predicate removal in a post-processing phase.

Our query rewrite framework follows the typical iterative *recognize-act inference cycle* approach [14] along with an inter-rule triggering scheme. However, the multi-block nature of SQL queries presents the challenge of deciding *how to effectively perform the first iteration* of rule application; from then on, the triggering mechanism takes over. Specifically, given a set of *n* rewrite rules and another set of *m* eligible query blocks, how to apply the *n* rules on the *m* query blocks? One design issue to be addressed here is whether we apply all rules for each block or apply a rule on all

blocks. If we apply a rule on all blocks, how to determine whether a top-down, bottom-up or arbitrary order is appropriate? To address these problems, we propose a two-phased approach. In the first phase, we apply all rewrites on all blocks and then capture (set) the three dimensional triggers for each block. Then, we iterate over the triggered rewrites until we achieve the final rewritten query.

Contributions: In summary, this paper makes the following contributions:

- (1) A detailed study of interactions involving a broad set of query rewrite rules including predicate pushdown, outer-to-inner join conversion, join elimination, predicate satisfiability-and-transitive closure (SAT-TC) and set operation simplification.
- (2) Present a query rewrite framework that is able to handle the multi-dimensional triggering involved in multi-block queries while retaining the order independence of rewrite rules. We also present a two-phased algorithm for efficiently implementing a rewrite driver that is scalable enough for deployment in a commercial database system.

The remainder of this paper is organized as follows: In section 2 we review previous work and highlight how our work relates to existing query rewrite frameworks. In section 3 we give a brief description of the query rewrite rules included in our framework. Section 4 presents the tree-dimensional triggering interactions among query rewrite rules and section 5 describes our query rewrite driver that implements algorithms to exploit the triggering interactions. Section 6 presents experimental results and we conclude in section 7.

2 Previous Work

Query rewrite based optimization (aka semantic optimization) has been studied since the late 80's. The Starburst extensible database system [1] is one of the early efforts to support query rewrite as a distinct phase of query optimization. The starburst paper [1] presented a suite of rewrite rules focused on merging multi-block queries into a single SELECT block, and also described a production rule engine for choosing and executing these rules (i.e. pairs of condition and action). It also discussed the triggering interaction of these rules. A subsequent paper [2] presented the design and implementation of starburst's rule engine for query rewrite optimization. This rule engine grouped rules into rule classes and introduced a variety of execution controls ranging from totally data-driven to totally procedural. It also incorporated a budget control scheme for controlling the resources taken for query optimization as well as guaranteeing the termination of rule execution. In this paper we look at the triggering interaction of a broader set of query rewrite rules than those presented in the starburst papers. While our query rewrite engine is similar to that of starburst's in general, we present a number of novel techniques that address the challenge of efficiently applying rewrite rules on large multi-block queries.

There have also been a number of extensible query optimization frameworks that supported query rewrite rules [3,4,5,6,7,9]. Prominent among them are the Volcano [3] and the Cascades [4] extensible architectures that evolved from Exodus [5]. In these systems, rules are used universally to represent knowledge of the search space.

Two kinds of rules are used, namely transformation rules that map an algebraic expression into another and implementation rules that map an algebraic expression into an operator tree. Unlike the starburst framework, the Volcano/Cascades frameworks (a) do not use two distinct optimization phases because all transformations are algebraic and cost-based, (b) the mapping from algebraic to physical operators occurs in a single step, and (c) instead of applying rules in a forward chaining fashion (Starburst's), they use goal-driven application of rules.

In [15], the query rewrite technique called “predicate move-around” is presented as a generalization of the classical predicate pushdown technique. The predicate move-around technique pushes predicates down, up and sideways in the query graph which makes it particularly helpful for queries which can not be rewritten into single block queries. Predicate move-around is also one of the query rewrites we cover but we focus on how it interacts with other query rewrite techniques. In [8] a method called Chase and Back-chase that exploits the systematic interaction of various query rewrite techniques to achieve better optimization is described. An example of such a useful interaction is when the presence of certain integrity constraints enables the use of indexes and materialized views in rewriting a query. The paper focuses on the use of indexes and materialized views, semantic optimization and join elimination (minimization). The interactions explored in [8] are complementary to those presented here.

In [12], a cost-based query transformation framework is described. It also presents a suite of heuristic and cost-based query optimization techniques. Some of the particular query rewrites rules discussed in [12] overlap with ours. However, [12] does not address the interactions among these rewrites and how these interactions can be exploited to efficiently apply the rewrite rules. Also [13] presented a variety of transformations to un-nest sub-queries in order to get a single SELECT query block. However, this paper does not use a rule based rewrite engine.

In our previous work, we have addressed the problems of optimizing multi-block queries [10] and outer join elimination [11]. These approaches presented in these papers correspond to particular re-write rules discussed in the current paper. Unlike these papers, we present a more generic framework here.

3 Individual Rewrite Rules

This section provides a high level description of each of the rule-based rewrites. We believe that these query rewrites are found in most commercial relational database systems in one form or another. We also do not claim here that the set of rewrite rules below is exhaustive but still it represents the most important rewrite rules. Also, in what follows, we use the term *view* as a generic reference to views, derived tables and WITH definitions.

- *Projection Pushdown (PP)* – This rewrite removes unreferenced columns/expressions from the SELECT lists of views. For example, this rewrite was applied to Q1 in section 1 where the *z* field was removed from the derived table DT.
- *Outer-to-Inner Join Conversion (O2I)* – This rewrite converts left/right outer joins into inner joins when it is guaranteed that all unmatching rows from the

inner table are filtered out by another predicate (i.e. the WHERE clause or another containing ON clause). For example, the left outer join in “SELECT * FROM Lineitem L LEFT OUTER JOIN Ordertbl O ON L.l_orderkey = O.o_orderkey WHERE O.o_orderstatus = ‘F’” can be converted to an inner join since the WHERE clause eliminates all unmatched rows. Similar logic can be used to convert full outer joins to left, right outer joins or inner joins.

- *View Folding (VF)* – This rule transforms a query by folding (merging) views with the main query. The resulting query no longer references the folded view. For example, the derived table DT was folded within Q1 in section 1.
- *Predicate Move-around (PM)* – This transformation moves predicates around views or subqueries. Pushing `DT.y = ‘1995-12-24’` into DT in Q1 is an example of this rewrite.
- *SAT-TC (SATisfiability-Transitive Closure)* – This rewrite rule checks whether a set of predicates are mathematically unsatisfiable (yield empty answer set). In Q1, `l_shipdate = ‘1995-12-24’ AND EXTRACT(MONTH FROM l_shipdate) <= 6` is an example of such a situation. Also, for predicates that cannot be proven to be unsatisfiable, transitive closure is applied to derive new predicates. For example, “`l_orderkey = 1`” is the transitive closure of “`l_orderkey=o_orderkey and o_orderkey = 1`”.
- *Set Operation Branch Elimination (SOBE)* – If a set operation has a branch with an unsatisfiable condition, the query can be rewritten without the branch. Removing the first branch of the UNION ALL in Q1 is an example of that.
- *Join Elimination (JE)* – Redundant joins can be eliminated from queries and this helps the overall performance of those queries. An inner join is redundant if the join is based on equality between a primary key and foreign key. Similarly, outer joins based on equality on unique fields from the inner table are redundant. In both cases the join can be removed along with either the parent table in the inner join case or the inner table in the outer join case. In both cases, the table to be removed should not have any projections that can not be mapped to other tables. An example of inner join elimination is shown below in Q2 and Q3. Assuming that `o_orderkey = l_orderkey` is a PK-FK relationship (based on referential integrity), the join between lineitem and ordertbl in Q2: “SELECT o_orderkey FROM Lineitem, Ordertbl WHERE l_orderkey = o_orderkey” can be removed resulting in query Q3: “SELECT l_orderkey FROM Lineitem WHERE l_orderkey IS NOT NULL”.

4 Interactions among Rewrites

As stated previously, a rewrite can trigger other rewrites as illustrated by the example in section 1. Section 5 will show how the QRW driver takes advantage of the triggering mechanism to apply the rewrites efficiently and completely. In this section we discuss the triggering relationships among rewrites and illustrate them with some simple examples. The trigger information is captured in Table 1 below. In this table, the three dimensions of triggering relationships are represented as (a blank in the table stands for “no trigger”):

- *Self*, i.e. the rewrite in the row can trigger the rewrite in the column on the same block.
- *Parent*, i.e. the rewrite in the row can trigger the rewrite in the column on the parent block.
- *Child*, i.e. the rewrite in the row can trigger the rewrite in the column on all children blocks.

Table 1. Triggers Table

	PP	O2I	VF	PM	SAT-TC	JE	SOBE
PP			SELF			SELF	
O2I			SELF		SELF	SELF	
VF		PARENT			PARENT	PARENT	PARENT
PM		SELF		CHILD/ PARENT	SELF		
SAT-TC				CHILD/ PARENT			PARENT
JE					SELF		
SOBE			SELF				

The following subsections explain, with some examples, the above interactions.

4.1 Projection Pushdown

Removing columns from a view may allow that view to be folded. For example, if a view with some window functions is joined to another table, it cannot be folded. However, if the window functions are not referenced, they can be removed. In Q4 below the derived table DT can not be folded since it has a window function which needs to be applied before the restriction in the main query. Q4: “SELECT Y FROM (SELECT SUM(l_quantity) OVER (ORDER BY l_orderkey), l_orderkey FROM Lineitem) DT(x,y) WHERE y < 1000”. Pushing the main query projection into DT produces Q5: “SELECT y FROM (SELECT l_orderkey FROM Lineitem) DT(y) WHERE y < 1000”. DT can be folded in Q5 producing Q6: “SELECT l_orderkey FROM Lineitem WHERE l_orderkey < 1000”.

To see how Projection Pushdown can affect Join Elimination, consider a view with an FK-PK join that has columns from the dimension table in the SELECT list of the view. If these columns are not referenced and are removed from the view, the join to the dimension table can possibly be eliminated. Q7: “SELECT x FROM (SELECT l_orderkey, o_orderdate FROM Lineitem, Ordertbl WHERE l_orderkey=o_orderkey) DT(x,y)” is an example where DT contains a PK-FK join but has projections on the Ordertbl parent table which prevents eliminating the join. Pushing the projections into DT produce query Q8 : “SELECT x FROM (SELECT l_orderkey FROM Lineitem, Ordertbl WHERE l_orderkey= o_orderkey) DT(x)”. The join in DT can now be removed from Q8 since Ordertbl has no projections other than the PK fields which results in the query Q9: “SELECT x FROM (SELECT l_orderkey FROM Lineitem) DT(x)”.

4.2 Outer to Inner Join Conversion

O2I may allow a view to be folded since there are several semantic restrictions on folding views that either contains outer joins or are involved in outer joins. For example, DT in Q10 cannot be folded since it is involved in a full outer join and its WHERE clause needs to be evaluated before the join. Q10: “SELECT * FROM Customer FULL OUTER JOIN (SELECT o_custkey FROM Ordertbl WHERE o_orderstatus = ‘F’) DT(x) ON c_custkey = DT.x WHERE DT.x BETWEEN 1 AND 100”. The full outer join in Q10 can be converted to a right outer join since the predicate on DT.X filters out all NULL value of DT. This conversion produces Q11: “SELECT * FROM Customer RIGHT OUTER JOIN (SELECT o_orderkey FROM Ordertbl WHERE o_orderstatus = ‘F’) DT(x) ON c_custkey = DT.x WHERE DT.x BETWEEN 1 AND 100”. DT in Q11 can now be folded since its WHERE clause can be combined with the main query WHERE clause. The final query is Q12: “SELECT Customer.*, o_orderkey FROM Customer RIGHT OUTER JOIN Ordertbl ON c_custkey = o_custkey WHERE o_orderstatus = ‘F’ AND o_orderkey BETWEEN 1 AND 100”.

Teradata applies SAT-TC to each ON clause individually and selectively applies SAT-TC across the WHERE clause and ON clauses or between different ON clauses. The details of this logic are beyond the scope of this paper but we can assert that there are more opportunities for SAT-TC with inner joins than outer joins. Therefore, O2I triggers SAT-TC. For example, SAT-TC cannot be applied on the combination of the WHERE and ON clauses in Q13: “SELECT * FROM Lineitem LEFT OUTER JOIN Ordertbl ON l_orderkey = o_orderkey WHERE o_orderkey = 1”. The outer join in Q13 can be converted to inner since the condition on o_orderkey filters out all un-matching rows. Applying O2I on Q13 produces Q14: ” SELECT * FROM Lineitem, Ordertbl WHERE l_orderkey = o_orderkey AND o_orderkey = 1”. Applying SAT-TC on Q14 derives l_orderkey = 1 and the final query Q15: ” SELECT * FROM Lineitem, Ordertbl WHERE l_orderkey = o_orderkey AND o_orderkey = 1 AND l_orderkey = 1”.

Finally, O2I can enable inner join elimination. For example, if o_orderkey = l_orderkey is a PK-FK relationship then the join in Q14 or Q15 can be eliminated

4.3 View Folding

Folding a view containing an outer join into a block that has a join between the view and another table can lead to the view’s outer join being converted to an inner join. For example, consider Q16: “SELECT Lineord.* FROM (SELECT * FROM Lineitem LEFT OUTER JOIN Ordertbl ON L_orderkey = O_orderkey) Lineord INNER JOIN Customer C ON Lineord.o_custkey = C.c_custkey”. Lineord can be folded into the query producing Q17: “SELECT Lineitem.*, Ordertbl.* FROM Lineitem LEFT OUTER JOIN Ordertbl ON L_orderkey = O_orderkey INNER JOIN Customer C ON o_custkey = c_custkey”. The outer join between Lineitem and Ordertbl can be converted to an inner since o_custkey = c_custkey filters out all un-matching rows of Ordertbl.

Folding a view combines the view’s ON/WHERE clause with the containing block’s ON/WHERE clause so additional opportunities to apply SAT-TC may become available. Q18: “SELECT * FROM (SELECT * FROM Lineitem, Ordertbl

WHERE l_orderkey = o_orderkey) Lineord WHERE Lineord.l_orderkey = 1". Lineord can be folded simplifying Q18 to Q19: "SELECT * FROM Lineitem, Ordertbl WHERE l_orderkey = o_orderkey AND l_orderkey = 1". Now, it is obvious that o_orderkey=1 can be derived in Q19.

View Folding can trigger Join Elimination. For example, Ordertbl is a child table of Customer based on c_custkey = o_custkey. If Q16 is modified to have no projections on the Customer table then the join with Customer can be eliminated resulting in Q20: "SELECT * FROM Lineitem, Ordertbl WHERE l_orderkey = o_orderkey and o_custkey IS NOT NULL".

Folding a view with an unsatisfiable condition into a block that is a branch of a set operation allows Set Operation Branch Elimination to remove this branch. To illustrate this concept consider Q21: "SELECT * FROM (SELECT * FROM Lineitem WHERE EXTRACT(MONTH FROM l_shipdate) <= 6) DT1 UNION ALL (SELECT * FROM (SELECT * FROM Lineitem WHERE EXTRACT(MONTH FROM l_shipdate) >= 7 AND l_shipdate = '2008-02-02') DT2)"

The derived table DT2 has a contradiction and when it is folded SOBE can be applied producing Q22: "SELECT * FROM Lineitem WHERE EXTRACT(MONTH FROM l_shipdate) <= 6".

Note that all these triggers are applied on the PARENT block since after view folding the view block is merged to its parent.

4.4 Predicate Move-Around

Moving predicates around, like pushing them into a view may allow outer joins in the view to be converted to inner joins. For example, consider the query Q23: "SELECT * FROM Customer, (SELECT COUNT(*), o_orderkey, o_custkey FROM Lineitem LEFT OUTER JOIN Ordertbl ON l_orderkey = o_orderkey GROUP BY o_orderkey, o_custkey) DT(x,y,z) WHERE x = 1". The condition x = 1 can be pushed into DT which in turn can change the outer join in DT into an inner join. Applying both re-writes produces the query Q24: "SELECT * FROM Customer, (SELECT COUNT(*), o_orderkey, o_custkey FROM Lineitem, Ordertbl WHERE l_orderkey = o_orderkey AND o_orderkey = 1 GROUP BY o_orderkey, o_custkey) DT(x,y,z)"

Introducing new predicates into a view via predicate move-around may also allow SAT-TC to find contradictions or derive new predicates. For example, pushing "o_orderkey=1" in the previous example allows the derivation of "l_orderkey = 1". In addition, these new predicates can be pushed into child blocks or pulled to the parent block.

4.5 SAT-TC

TC-derived predicates may be pushed into child blocks like views or subqueries. They could also be pulled up for the parent block. Unsatisfiable conditions found by SAT in a set operation branch would allow the branch to be removed like query Q1 in section 1.

4.6 Join Elimination

Eliminating an outer join may allow more derivation of predicates and therefore trigger SAT-TC. To see this, consider the join sequence (*Lineitem L1 INNER JOIN OrderTbl ON l_orderkey = o_orderkey*) *LEFT JOIN Customer ON O1.o_custkey = c_custkey INNER JOIN Linetem L2 on L1.l_orderkey = L2.l_orderkey*. If *c_custkey* is a unique column, the left outer join can be eliminated. The remaining two inner joins can then be combined and the condition “*O1.o_orderkey = L2.l_orderkey*” can be derived.

4.7 Set Operations Branch Elimination

If removing branches of a set operation results in a single remaining SELECT or UNION ALL in a view, it may be possible to fold the view. This is illustrated by the simplified query of Q1 in section 1.

5 The Rewrite Driver

The objective of the QRW driver is to maximize the benefit of rewrites without spending too much time doing that. The maximum benefit of the rewrites is accomplished by applying them as much as possible. Without the triggering knowledge, the QRW driver could incur a significant overhead that overshadows the benefits of the rewrites. We use the algorithm *NoTriggersRewrite* to illustrate the potentially large cost of a QRW driver. Before we proceed, it is important to distinguish between fired and applied rewrites. We say a rewrite rule is *fired/attempted* on a query block when the rule's conditions are checked on a given query block. A fired rule may or may not succeed. We say a rewrite is *applied* on a query block when it resulted in a re-written query block. The *NoTriggersRewrite* algorithm, shown below, basically keeps firing all rewrites indiscriminately until no more rewrites are possible.

Procedure NoTriggersRewrite(Q)

- *Q* is the input query. It is also the output after applying rewrites to it.
- Assume that there are *n* rewrites *R1, R2, ..., Rn*. We also assume that *Q* has *m* blocks *B1, B2, ..., Bm*

Begin

While (TRUE) do

Begin

Applied = FALSE

For i=1 to n do

For j=1 to m do

Fire Ri on Bj. If successful then apply Ri on Bj and set Applied = TRUE.

If (NOT Applied) return

End

End

Every time a rewrite is fired at any block, Procedure NoTriggersRewrite tries *all* rewrites on *all* blocks of the query. This is the case since applying a rewrite can make the same rewrite or other rewrites become applicable to the same block or other blocks. For example, deriving a condition at a given block B1 enables applying pushing predicates to all children blocks of B1. The main loop of NoTriggersRewrite could go on for quite some time. This happens, for instance, when a sequence of predicate derivation takes place in a block which enables pushing a condition to all children of that block. Pushing these conditions in turn enables SAT-TC and this sequence could continue for the depth of the nested blocks.

The QRW driver can be implemented more efficiently using the triggers information presented in section 4. The basic idea is that instead of firing all rewrite rules for each applied rule, we only fire those rules that have interaction with the applied rewrite. In order to implement this idea, we introduce a third state for rules, called *enabled*, in addition to fired and applied. We say a rule is *enabled* on a block when a previously applied rewrite makes it eligible to be fired on the block. Below, we give the TriggersRewrite which is based on rule interactions:

Procedure TriggersRewrite(Q)

- Let Q and the rewrite array be the same as NoTriggersRewrite.
- T is an n by n array that represents the triggers table in section 4.
- E is a boolean m by n array used to mark which rewrite is enabled on which block. For example, if E[i,j] is TRUE then Ri is to be fired on Bj. Initially all entries are set to TRUE (i.e. all rewrites to be fired on all blocks).

Begin

While (at least one entry in E is TRUE) do

Begin

For i=1 to n do

For j=1 to m do

If E[i,j]

Begin

Set E[i,j] = FALSE;

Fire Ri on Bj. If successful then

Begin

Apply Ri on Bj

For k=1 to n

if T[i,k] is SELF, set E[k,j] to TRUE

else if T[i,k] is PARENT, set E[k,l] to TRUE where Bl is the parent of Bj

else if T[i,k] is CHILD, set E[k,l1],E[k,l2]...,E[k,lq] to TRUE where Bl1, Bl2, ..., Blq are the child blocks of Bj;

End

End

End

End

To simplify the discussion, we assume that the rewrites are ordered based on Table 1. Algorithm TriggersRewrite terminates when no more rewrites can be applied

just like NoTriggersRewrite. Since all entries in $E[i,j]$ are initialized to TRUE, TriggersRewrite fires all rewrites on all blocks during the first iteration. After that only rewrites enabled on the relevant blocks are fired.

For very complex queries, the cost of the TriggersRewrite algorithm may be too high thereby increasing the overall query optimization time to unacceptable level. To avoid this situation, some control on the resource consumption of the rewrite engine is necessary. Presently, we adopt a simple approach that uses a pre-specified cap on resource consumption including amount of time taken and/or number of rewrite iteration performed.

Finally, we remark that both NoTriggersRewrite and TriggersRewrite are guaranteed to eventually terminate. Although we do not provide a complete proof due to space limitation, termination can be shown by observing that cyclic triggering interaction can occur only in a particularly restricted scenario which eventually lead to a finite rewrite path. As shown in [17], if there are no cycles in the triggering interaction graph of a given set of rules, the rules are guaranteed to terminate. In our algorithms the cycles lead to finite rewrite path in the graph due to two reasons: (1) each rewrite performs a monotonic update and hence at some point it runs out of a rewrite which prevents it from triggering any other rewrite, and (2) no redundant predicates are derived by a rewrite rule and hence one rewrite rule can not undo the result of another.

6 Experiments

The goal of our experimental study was to evaluate the efficiency of the QRW subsystem by comparing the resource consumption of TriggersRewrite with NoTriggersRewrite. We performed the experiments on a 100GB TPC-H database on a 2-node dual-core machine with 36 disks (73GB @15K RPM) attached. We adapted 30 real-life queries to the TPC-H data model representing a mix of queries with aggregates, views and derived tables. We ran the driver in both TriggersRewrite and NoTriggersRewrite versions, as presented in section 5. In all cases, the plans obtained by the two methods were identical. Also in all cases, the amount of system resources utilized by TriggersRewrite was lower than or identical to those utilized by NoTriggersRewrite. The detailed results associated from these experiments are shown in Table 2.

Table 2. Resource Consumption by TriggersRewrite vs. NoTriggersRewrite

Driver Version	QRW parsing time (μ s)	Total optimizer time (μ s)	QRW memory utilized (Kbytes)
TriggersRewrite	5,777	27,803	472
NoTriggersRewrite	18,968	41,529	1,413

As shown in Table 2, we measured the average amount of rewrite parsing time, i.e. the amount of time the parser spent in the rewrite part of the query optimization process, expressed in microseconds, the average total optimizer time (also expressed in microseconds) and the average amount of memory utilized by the rewrite code

(expressed in Kilobytes). Overall, NoTriggersRewrite consumed more than 3 times more resources than TriggersRewrite. In addition, in four of the queries involving a large number of blocks, NoTriggersRewrite required prohibitively large parsing time and amount of memory. In order to avoid bias in the analysis, we excluded these queries from consideration when analyzing the results.

The total optimizer time for the two driver versions has been shown in column two of Table 2. We have shown both the QRW parsing time and the total optimizer time which includes QRW parsing time but for memory we have shown only the total memory used. This is because the optimizer reuses memory previously utilized by the QRW parsing process. For the 26 queries included in the analysis, the TriggersRewrite algorithm is overall more efficient in the sense that it produces the same execution plan at a lower cost than NoTriggersRewrite. Specifically, the total optimizer time is, on the average, 50% lower for TriggersRewrite than NoTriggersRewrite. Such a reduction is particularly important for applications like active data warehousing (ADW) where the DBMS has to simultaneously execute a very large number of short-running queries and for such queries optimization time (including query rewrites) is a significant portion of the total execution time.

7 Conclusions and Future Work

We presented the interaction among a set of important query rewrite rules and showed how this interaction knowledge is used to devise a triggering mechanism in the Tera-data QRW driver in order to efficiently attempt the rewrites on the complex, multi-block queries. We also demonstrated through experiments on a real dataset that the triggering scheme actually reduces the resources consumed by the QRW driver.

There are a number of avenues to pursue in future work. First is to explore adding cost-based rewrites to the suite of the current rule based rewrites. Cost considerations are particularly important when multiple rewrites are possible on a query or to decide whether to further rewrite a query or not. Another issue is developing a more dynamic ordering of rewrite rules since certain orders of rewrites are more efficient than others. Yet another issue coming up with a more intelligent way to dynamically control the resource consumption of the QRW sub-system. A promising approach to this end is using a measure of the overall complexity of the query to estimate the expected resource consumption of the rewrite process in order to dynamically set the resource consumption control. For example, one can use the estimated overall runtime of the query (i.e. computed before query rewrites are applied) and set the control on the query rewrite resource consumption as some fraction of this overall time. Typically, queries that are expected to take a long time to run (sometimes called “strategic queries” in Teradata parlance) may be allowed more resource for query rewrite phase as the potential gains from exhaustive rewrites can be large. In contrast, queries that are expected to run fast can be allocated smaller query rewrite resource since the cost-benefit ratio of query rewrites is potentially small. In addition to setting it dynamically, we also can provide knobs for database administrators to set the optimal ratio appropriate for the particular query workload of their particular databases.

References

- [1] Pirahesh, H., Hellerstein, J.M., Hasan, W.: Extensible/rule Based Query Rewrite Optimization in Starburst. *SIGMOD*, 39–48 (1992)
- [2] Pirahesh, H., Cliff Leung, T.Y., Hasan, W.: A Rule Engine for Query Transformation in Starburst and IBM DB2 C/S DBMS. In: *ICDE*, pp. 391–400 (1997)
- [3] Graefe, G., McKenna, W.J.: The Volcano Optimizer Generator: Extensibility and Efficient Search. In: *ICDE*, pp. 209–218 (1993)
- [4] Graefe, G.: The Cascade Framework for Query Optimization. *IEEE Data Engineering Bulletin* 18(3), 19–29 (1995)
- [5] Graefe, G., Dewitt, D.J.: The Exodus Optimizer Generator. *SIGMOD*, 160–172 (1987)
- [6] Cherniack, M., Zdonik, S.: Changing the Rules: Transformations for Rule-based Optimizers. *SIGMOD*, 61–72 (1998)
- [7] Warshaw, L.B., Miranker, D.P.: Rule-based Query Optimization, Revisited. In: *CIKM*, pp. 267–275 (1999)
- [8] Popa, L., Deutsch, A., Sahuguet, A., Tannen, V.: A Chase Too Far? In: *SIGMOD*, pp. 273–284 (2000)
- [9] Haas, L.M., Kossmann, D., Wimmers, E.L., Yang, J.: Optimizing Queries Across Diverse Data Sources. In: *VLDB*, pp. 276–285 (1997)
- [10] Ghazal, A., Bhashyam, R., Crolotte, A.: Block Optimization in the Teradata RDBMS. In: Mařík, V., Štěpánková, O., Retschitzegger, W. (eds.) *DEXA 2003*. LNCS, vol. 2736, pp. 782–791. Springer, Heidelberg (2003)
- [11] Ghazal, A., Crolotte, A., Bhashyam, R.: Dynamic Constraints Derivation and Maintenance in the Teradata RDBMS. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) *DEXA 2001*. LNCS, vol. 2113, pp. 390–399. Springer, Heidelberg (2001)
- [12] Ahmed, R., Lee, A., Witkowski, A.: Cost-Based Query Transformation in Oracle. In: *VLDB*, pp. 1026–1036 (2007)
- [13] Elhemali, M., Galindo-Legaria, C.A., Grabs, T., Joshi, M.M.: Execution Strategies for SQL Subqueries. In: *SIGMOD*, pp. 993–1003 (2007)
- [14] Brownston, L., Farrell, R., Kant, E., Martin, N.: *Programming Expert Systems in OPS5: An Introduction to Rule-based Programming*. Addison-Wesley, Reading (1985)
- [15] Levy, A.Y., Mumick, I., Sagiv, Y.: Query Optimization by Predicate Move-around. In: *VLDB*, pp. 96–108 (1994)
- [16] TPC-H specification – Transaction Performance Council, <http://www.tpc.org>
- [17] Aiken, A., Widon, J., Hellerstein, J.M.: Behavior of Database Production Rules: Termination, Confluence, and Observable Determinism. In: *SIGMOD*, pp. 59–68 (1992)

Optimal Preference Elicitation for Skyline Queries over Categorical Domains

Jongwuk Lee¹, Gae-won You¹, Seung-won Hwang¹,
Joachim Selke², and Wolf-Tilo Balke²

¹ Department of Computer Science and Engineering, POSTECH, Korea

² L3S Research Center, Leibniz Universität Hannover, Germany

{julee,gwyou,swhwang}@postech.edu

{selke,balke}@L3S.de

Abstract. When issuing user-specific queries, users often have a vaguely defined information need. Skyline queries identify the most “interesting” objects for users’ *incomplete preferences*, which provides users with intuitive query formulation mechanism. However, the applicability of this intuitive query paradigm suffers from a severe drawback. Incomplete preferences on domain values can often lead to impractical skyline result sizes. In particular, this challenge is more critical over categorical domains. This paper addresses this challenge by developing an iterative elicitation framework. While user preferences are collected at each iteration, the framework aims to both minimize user interaction and maximize skyline reduction. The framework allows to identify a reasonably small and focused skyline set, while keeping the query formulation still intuitive for users. All that is needed is answering a few well-chosen questions. We perform extensive experiments to validate the benefits of our strategy and prove that a few questions are enough to acquire a desired manageable skyline set.

1 Introduction

The information need of users in today’s databases and information systems has evolved from SQL-style exact match queries to answering vague queries. To address this need, new query paradigms like top- k retrieval or skyline queries have been recently studied. These paradigms assess the grades of match in all database objects with respect to a given query, and only identify the best matching results.

More specifically, the strengths of two paradigms are complementary. Top- k retrieval returns only the best k objects based on a user-specific *utility function* combining scores with respect to all queried attributes. While top- k queries always provide a focused and manageable set, it is difficult for end-users to define an exact utility function for their individual preferences. In contrast, skyline queries do not require users to define a utility function, and simply identify “interesting” objects that are not “dominated” by any other objects. While this intuitive query formulation has been a key strength of skyline queries, it is impossible for users to control the size of skyline. In particular, when the number of

Table 1. Toy dataset for Example 1

<i>ID</i>	<i>type</i>	<i>color</i>	<i>brand</i>
o_1	convertible	red	Ferrari
o_2	sedan	red	Ferrari
o_3	convertible	blue	Ferrari
o_4	sedan	blue	Toyota
o_5	roadster	blue	Honda

queried attributes increases, the size of skyline also increases exponentially, *i.e.*, *curse of dimensionality*. This challenge is especially more critical over categorical domains.

This paper deals with skyline queries over *categorical domains* in which the challenge of skyline queries is more critical. Although both paradigms have been mostly applied for numerical domains in the previous literatures (*e.g.*, minimizing *price* or *distance*), these can also be used for categorical domains as well (*e.g.*, maximizing a preference on favorite *color* or *brand*). To illustrate, Example 1 describes how skyline queries work in categorical domains.

Example 1. Consider a customer shopping for an ideal car with respect to three attributes *type*, *color* and *brand*. Suppose that a user gives specific preferences that he/she prefers ‘convertible’ to ‘sedan’ for *type*, ‘red’ to ‘blue’ for *color*, and ‘Ferrari’ to ‘Honda’ for *brand*. Based on these preferences, we identify car o_1 as one of the best choices, *i.e.*, a skyline object, from the toy dataset in Table 1. This means o_1 is superior to o_2 and o_3 in all dimensions, *i.e.*, o_1 *dominates* o_2 and o_3 . However, the user preferences are not sufficient to determine a preference between o_1 and o_4 , or o_1 and o_5 , *i.e.*, o_1 is *incomparable* with o_4 or o_5 .

As Example 1 illustrates, in practical scenarios, the amount of preference information available to query processing is usually limited, because specifying all relationships requires considerable effort for the user. Missing relationships are thus interpreted as *indifference*, or equal importance for the user. As a result, skyline query results will typically include all the incomparable objects, due to incomplete user preferences.

This paper studies the problem of eliciting preferences enough to acquire a concise skyline result set. In particular, we use the cardinality of different domain values with respect to the database instance (and *a priori* knowledge on user preferences, if exists). This makes users elicit more useful preferences with minimal user efforts. Ideally, such an elicitation process achieves both minimizing user interaction and maximizing skyline reduction. We thus aim at developing and evaluating an optimal elicitation process. In summary, this paper has the following contributions:

- We study preference elicitation in numerical and categorical domains and design an optimal elicitation strategy (Section 2)

- We develop Framework *MaxPrune* to identify skylines with reasonable size by implementing our optimal elicitation strategy. (Section 3)
- We validate effectiveness and efficiency of Framework *MaxPrune*. (Section 4)

This paper is organized as follows. Section 2 presents preliminaries on qualitative preference and elicitation model over categorical domains. Section 3 proposes a framework adopting optimal elicitation method in the given problem setting, and Section 4 validates Framework *MaxPrune*. Section 5 briefly reviews existing efforts related to our work. Finally, Section 6 discusses our future work.

2 Preliminaries

This section states preliminaries to help understand our framework. Let D be a data space with finite n attributes $\{D_1, D_2, \dots, D_n\}$, where D_i denotes a set of possible *domain values* on i^{th} attribute. Specifically, let D be *possible alternatives*, i.e., $D := D_1 \times D_2 \times \dots \times D_n$, and A be *actual alternatives* as a subset of D , i.e., $A \subseteq D$. An alternative $a = (a_1, \dots, a_n)$ is contained in a product set $A := A_1 \times A_2 \times \dots \times A_n$. A *weak order* is denoted as $\succeq$ on the set of alternatives A , by setting $a \succeq b$ if and only if a is equal to or more preferred than b . The asymmetric part and symmetric part of weak order, denoted as $\succ$ and $\sim$, correspond to *strict order* and *indifference*, respectively.

2.1 Qualitative Preferences

We first discuss strengthes and weaknesses of qualitative preferences. Specifically, given alternatives a and b , it clearly requires much less cognitive effort to tell which one among $a \succ b$, $b \succ a$, and $a \sim b$ holds. This ignores any numerical values and solely considers an induced weak order. However, for large D and A , it seems hopeless to ask the user about his/her preferences in a qualitative way, since there are $\binom{|D|}{2}$ pairs to be compared. An exception is a numerical attribute domains with an inherent order based on which users can express preference straightforwardly, e.g., ascending order of “price”.

For practical aspects, we introduce *ceteris paribus* semantics which provides an intuitive meaning [24,13]. For instance, saying “red $\succ_{\text{color}}$ blue” means “The user prefers red cars to blue ones, if everything else is equal”. Since this is exactly the meaning of just saying “red cars are better than blue ones”, stating preferences in terms of attribute value comparisons is highly intuitive. Based on *ceteris paribus* semantics, preference monotonicity between alternatives can be constructed over multi-attribute domains. This construction rule exploits *Pareto aggregation*, a relational operator that maps a sequence of weak orders into a binary relation on a set. Specifically, let $W_1, \dots, W_k$ be weak orders on set $S := S_1 \times \dots \times S_k$. The operator of Pareto aggregation, denoted as Par , is defined as follows: For $a = (a_1, \dots, a_k), b = (b_1, \dots, b_k) \in S$, it is $(a, b) \in \text{Par}(W_1, \dots, W_k)$ if and only if $a_i W_i b_i$ is true, for any $1 \leq i \leq k$. It is easy to show that $\text{Par}(W_1, \dots, W_k)$ is derived from weak orders on S .

Returning to the problem of reconstructing a total order $\succeq$ on A from attribute orders $\succeq_1, \dots, \succeq_n$, it is known that $\text{Par}(\succeq_1, \dots, \succeq_n)$ is the best reconstruction of $\succeq$. We have the following reasons: First, $\text{Par}(\succeq_1, \dots, \succeq_n)$ is always a subset of $\succeq$. Second, for any superset of $\text{Par}(\succeq_1, \dots, \succeq_n)$, there exists a utility function inducing weak orders $\succeq_1, \dots, \succeq_n$. We thus will base our model on $\succeq_1, \dots, \succeq_n$ and $\text{Par}(\succeq_1, \dots, \succeq_n)$. For the sake of representation, we simplify $\text{Par}(\succeq_1, \dots, \succeq_n)$ into $\succeq_{\text{Par}}$. Also, The symmetric part and asymmetric part of $\succeq_{\text{Par}}$ correspond to $\succ_{\text{Par}}$ and $\sim_{\text{Par}}$, respectively.

The final questions to be answered are then: What are the “best” alternatives? What alternatives should be returned by the database system when the attribute orders $\succeq_i$ are known? To answer these questions, we adopt skyline queries leveraging Pareto aggregation, and define the “best” actual alternatives to be exactly those that are not strictly dominated in A with respect to $\text{Par}(\succeq_1, \dots, \succeq_n)$. More formally, we define dominance and skyline, respectively. (These definitions are consistent with the definition of skyline used in all the existing skyline work.)

Definition 1. An alternative $a \in A$ strictly dominates an alternative $b \in A$ if and only if $a_i \succeq_i b_i$, for any index i , and there is an index j such that $a_j \succ_j b_j$.

Definition 2. An alternative $a \in A$ is a skyline object if and only if there is no alternative $b \in A$ that strictly dominates a .

2.2 Preference Elicitation

The term *preference elicitation* refers to the task of collecting information about the user’s preferences. In the existing skyline work, it is usually assumed that $\succeq_1, \dots, \succeq_n$ are complete total orders for preference elicitation. However, this assumption is unrealistic over categorical domains.

We first discuss how to model preference elicitation for collecting more “informative” user preferences. In particular, we model preference elicitation as an *iterative* process in which the user answers which one among $a \succ_i b$, $b \succ_i a$, and $a \sim_i b$ holds, where $a, b \in D_i$. User preferences are consistently collected for t iterations, which is essentially binary relations $\succeq_1^{(t)}, \dots, \succeq_n^{(t)}$, where index $t \in \mathbb{N}$ refers to the time index of elicitation iteration. As preference elicitation accumulates monotonically, preference knowledge also accumulates in any elicitation step, i.e., for any i and t , the relation $\succeq_i^{(t+1)}$ is a superset of $\succeq_i^{(t)}$. Since we know that the “true” orders $\succeq_i$ are reflexive and transitive, an elicited order $\succeq_i^{(t)}$ also must have these properties. We formally state an elicitation step at time t . (For simplicity, we denote the existing derived notations as follows: $\succ_i^{(t)}$, $\sim_i^{(t)}$, $\succeq_{\text{Par}}^{(t)}$, $\succ_{\text{Par}}^{(t)}$, and $\sim_{\text{Par}}^{(t)}$.)

Definition 3. Given weak orders $\succeq_1^{(t)}, \dots, \succeq_n^{(t)}$, an elicitation step from time t to time $t + 1$ is the following procedure:

- (1) Choose attribute values $a, b \in D_i$ on i^{th} attribute, where neither $a \succeq_i^{(t)} b$ nor $b \succeq_i^{(t)} a$ is true.
- (2) Ask the user which one among $a \succ_i b$, $b \succ_i a$, and $a \sim_i b$ is true.

- (3) If $a \succ_i b$ is true, define $\succeq_i^{(t+1)}$ to be the transitive closure of $\succeq_i^{(t)} \cup \{(a, b)\}$.
 If $b \succ_i a$ is true, define $\succeq_i^{(t+1)}$ to be the transitive closure of $\succeq_i^{(t)} \cup \{(b, a)\}$.
 If $a \sim_i b$ is true, define $\succeq_i^{(t+1)}$ to be the transitive closure of $\succeq_i^{(t)} \cup \{(a, b), (b, a)\}$.

The elicitation process starts at time $t = 0$ with weak orders $\succeq_i^{(0)}$, which can contain initial information on user's preferences. That is, it can contain domain-specific preferences shared by all users, or personalized preference information based on a user profile.

2.3 Optimal Elicitation Method

The hardest part of preference elicitation is asking the user the right questions. Some questions may result in a large decrease of skyline size when stepping from $\text{SKY}(A, \succeq_{\text{Par}}^{(t)})$ to $\text{SKY}(A, \succeq_{\text{Par}}^{(t+1)})$, while other questions might not. For example, we know nothing about the user's preferences, but we know A to contain roughly as many blue cars as red cars. It thus would be a reasonable strategy to ask the first question about the preference relationship between attribute values "red" and "blue". If the user is not indifferent between both, the answer to this question can be expected to result in a large decrease of skyline size (assuming a good-natured data distribution in A). Based on this property, we formally state an elicitation method as follows:

Definition 4. *An elicitation method E is a deterministic algorithm that takes initial attribute weak orders $\succeq_1^{(0)}, \dots, \succeq_n^{(0)}$ as input and performs a sequence of elicitation steps until time t is reached with $\succeq_1^{(t)}, \dots, \succeq_n^{(t)}$ being weak orders.*

Clearly, the optimality of elicitation method depends on the distribution of actual alternatives A and prior knowledge of typical user preferences. To represent these notions, we introduce the following notations: Let $\mathcal{W}_i$ be the set of all possible weak orders on D_i . Also, let $\mathcal{W} := \mathcal{W}_1 \times \dots \times \mathcal{W}_n$, and let $\mathcal{Q}$ be a probability distribution on $\mathcal{W}$, where $Q = (Q_1, \dots, Q_n)$ denotes a random variable having distribution $\mathcal{Q}$. We use distribution $\mathcal{Q}$ to model prior knowledge of user preferences. (We will later discuss different elicitation decisions based on the distribution $\mathcal{Q}$ in details.) Up to now, we assumed both the elicitation method E and the user's preferences $w \in \mathcal{W}$ used for answering the elicitation questions to be fixed. To allow more precision in further definitions, we extend notations by making these assumptions explicit. We thus denote notation $\succeq_{\text{Par}}^{(t)}$ at time t as $\succeq_{E, w, \text{Par}}^{(t)}$.

Our goal finds a minimal sequence of questions with the smallest number t for $|\text{SKY}(A, \succeq_{E, w, \text{Par}})| \leq k$ by steps $_{E, w}(k)$, where k means user-specific retrieval size. This setting is similar to that of top- k retrieval except for two major differences: First, while a result set in top- k retrieval is assumed to contain the "best" k objects, in our setting a set of all the optimal objects (of size at most k) is required to be returned. Second, in top- k retrieval the user has to be proactive and state her/his preferences in advance. In our case, the active part is played by an internal framework. We formally state the optimality of elicitation method.

Definition 5. Denote the class of all elicitation methods that take an additional $\mathbb{N}$ -valued input argument by $\mathcal{C}$. For any $E \in \mathcal{C}$, write $E(k)$ to denote the behavior of E when given input k . An elicitation method $E \in \mathcal{C}$ is optimal (with respect to D , A , and $\mathcal{Q}$) if and only if

$$\mathbb{E}(\text{steps}_{E(k), \mathcal{Q}}(k)) \leq \mathbb{E}(\text{steps}_{F(k), \mathcal{Q}}(k))$$

holds, for $k \in \mathbb{N}$, $\forall F \in \mathcal{C}$, and the number of questions $\mathbb{E}(\cdot)$.

The idea underlying this definition is that an optimal elicitation method should ask questions in a way such that the number of expected questions needed to reach the target skyline size is as small as possible. Note that the definition of optimality is relative to D , A , and $\mathcal{Q}$. In practice, we are looking for a general optimal algorithm that work well regardless of the choice of D , A , and $\mathcal{Q}$. This corresponds to a greedy algorithmic approach, where in any elicitation step, the most desirable question leading to the best possible reduction in skyline size will be chosen. Without loss of generality, the next definition presents a step-optimal elicitation method, which also guarantees global optimality. (We will discuss this property in Section 3.)

Definition 6. A greedy elicitation method E is called step-optimal (with respect to D , A , and $\mathcal{Q}$) if and only if, when given attribute preorders $\succeq_1^{(t)}, \dots, \succeq_n^{(t)}$ for input, the algorithm E maximizes the term

$$\left| \text{SKY}(A, \succeq_{Par}^{(t)}) \right| - \mathbb{E} \left(\left| \text{SKY}(A, \succeq_{F, Q', Par}^{(t+1)}) \right| \right)$$

over all greedy elicitation methods F (given the same input), with $Q' = (Q'_1, \dots, Q'_n)$ as a random variable that is distributed according to $\mathcal{Q}$ conditioned on the fact that Q'_i is a superset of $\succeq_i^{(t)}$, for any i .

3 Optimal Elicitation Framework

In this section, we implement an optimal elicitation framework (discussed in Section 2.3) in a restricted problem setting. In particular, our framework aims at maximizing *expected pruning cardinality* (Definition 5) at each elicitation step, based on which the greedy elicitation strategy has global optimality.

3.1 Problem Setting

At each iteration, our framework shows a *sample pair*¹ (a, b) such that $a, b \in D_i$, and prune out objects that are never qualified as skyline objects. This means, when a user selects a over b , every object o with non-preferred attribute value

¹ Note the sample corresponds to a question on some pair a and b , asking which among $a \succ b$, $b \succ a$, or $a \sim b$ holds.

b can be pruned, if there exists another object o' having the same values as o in all dimensions except for $o'(D_i) = a$. We formally state this pruning process as follows. (Due to the space limitation, we leave all the proofs to our technical report [20].)

Lemma 1 (Pruning Process). *For user preference $a \succeq_i b$ on $a, b \in D_i$, we safely prune out object o such that $o(D_i) = b$, if there exists object o' such that $o'(D_i) = a$ and $\forall j(1 \leq j \leq n, j \neq i) : o(D_j) = o'(D_j)$.*

Note that, we assume that there always exists such dominating object o' , which simplifies the pruning process. We argue that this assumption is often true in real-life data, as highly preferred values often have high frequency as well (as Zipf's law similarly observed, *i.e.*, the frequency of any word is roughly inversely proportional to its rank in the number of frequency). This observation implies that a dominating (or highly preferred/ranking) object o' is highly likely to exist as in our assumption.

3.2 Framework *MaxPrune*

We first derive a sample (a, b) maximizing *pruning cardinality* $PC(\cdot)$ at the t^{th} step. Specifically, pruning cardinality $PC(\cdot)$ means the number of pruned objects by Lemma 1. Let s_i denote a sample at the i^{th} step. Note that $PC(\cdot)$ is conditional for prior elicitation— For a set of skyline objects after an elicitation of the t^{th} step, denoted as $SKY(A, \succeq_{s_t, w, \text{Par}}^{(t)})$, $PC(s_t)$ of sample $s_t = (a, b)$ depends on prior samples, $s_1, \dots, s_{t-1}$. In particular, when a user answers his/her preference on s_t , $w = a \succeq_i^{(t)} b$, we remove objects with value b from a set of current skyline objects $SKY(A, \succeq_{\text{Par}}^{(t-1)})$. $PC(s_t, w | SKY(A, \succeq_{\text{Par}}^{(t-1)}))$ is thus denoted as

$$\left| SKY(A, \succeq_{\text{Par}}^{(t-1)}) \right| - \left| SKY(A, \succeq_{s_t, w, \text{Par}}^{(t)}) \right|.$$

Observe that $PC(s_t, w | SKY(A, \succeq_{\text{Par}}^{(t-1)}))$ is maximized when the number of objects in $SKY(A, \succeq_{\text{Par}}^{(t-1)})$ with less preferred attribute value is maximal. We formally state this property as follows:

Lemma 2 (Maximizing Pruning Cardinality). *For user preference $w = a \succeq_i^{(t)} b$ of $s_t = (a, b)$ on $a, b \in D_i$, $PC(s_t, w | SKY(A, \succeq_{\text{Par}}^{(t-1)}))$ is maximized, when the number of objects with less preferred value b is maximal.*

Based on Lemma 2, we discuss how to decide a sample maximizing pruning cardinality. In fact, since pruning cardinality depends on user preference w , we develop a *probabilistic* framework with the following two scenarios, with and without *a-priori* knowledge based on distribution $\mathcal{Q}$ for user preference.

- **Without a-priori knowledge on $\mathcal{Q}$:** With no a-priori knowledge, we assume that a probability that each value in a sample pair is selected is equal chances, *i.e.*, $\frac{1}{2}$. That is, this probability implies that the user equally prefers

either one among the two values. Distribution $\mathcal{Q}$ for user preferences thus follows uniform distribution on $\mathcal{W}$. For instance, when showing a sample ('Ferrari', 'Honda') for 'brand', we assume that a user prefers each value with $\frac{1}{2}$ probability.

- **With a-priori knowledge on $\mathcal{Q}$:** On the other hand, we may have *a-priori* knowledge on user preferences, *e.g.*, such as *query frequency* from prior query logs. Based on this we can model user preferences more realistically. The distribution $\mathcal{Q}$ for user preferences shows different distributions according to $\mathcal{W}$. For instance, when a relative preference probability between 'Ferrari' and 'Honda' is p_f and p_h respectively, the probability of choosing 'Ferrari' over 'Honda' can be computed as $p_f/(p_f + p_h)$, while that of choosing 'Honda' is $p_h/(p_f + p_h)^2$.

For ease of understanding, we first develop our framework assuming *no a-priori knowledge*, which is later extended to consider also *a-priori knowledge*. In particular, we develop the notion of *expected pruning cardinality* based on the probabilistic assumption that the selected probability of the two values is equivalent.

Theorem 1 (Expected Pruning Cardinality). *Assuming no a priori knowledge, expected pruning cardinality $\text{EPC}(s_t | \text{SKY}(A, \succeq_{\text{Par}}^{(t-1)}))$ is maximized, when presenting sample $s_t = (a, b)$ in which the number of objects with value a and b in $\text{SKY}(A, \succeq_{\text{Par}}^{(t-1)})$ is maximal.*

Theorem 1 can be straightforwardly extended to consider the case of a-priori knowledge. Let the relative preferences of a and b in $s_t = (a, b)$ be p_a and p_b , respectively. In that case, expected pruning cardinality $\text{EPC}(s_t | \text{SKY}(A, \succeq_{\text{Par}}^{(t-1)}))$ is maximized, when choosing sample $s_t = (a, b)$ which maximizes $p_b \times c_a$ and $p_a \times c_b$ in which c_a and c_b is the cardinality of objects with value a and b in $\text{SKY}(A, \succeq_{\text{Par}}^{(t-1)})$.

Based on Theorem 1, we derive the global optimality of greedy elicitation at each step. Specifically, when choosing a sample with the highest expected pruning cardinality at each step, global pruning cardinality also guarantees optimality. To prove this property, we first show properties on the order of samples selected from each step in Lemma 3 and Lemma 4. Based on these properties, we can show that global pruning optimality in Theorem 2 can be derived from selecting maximal expected pruning cardinality at each step.

Lemma 3 (Exchange of adjacent samples). *Given a sequence of samples $S = (s_1, \dots, s_t)$, changing the order of an arbitrary pair of adjacent samples has no effect on the sum of the expected pruning cardinality.*

Lemma 4 (Ordering Independence). *Changing the order of a given sequence of samples $S = (s_1, \dots, s_t)$ has no effect on the sum of the expected pruning cardinality.*

² Assume that this relative preference is independent regardless of other relative preferences.

Table 2. Illustration of *MaxPrune*

<i>type</i>	<i>card.</i>	<i>color</i>	<i>card.</i>	<i>brand</i>	<i>card.</i>
convertible	40	red	60	Ferrari	35
sedan	30	blue	40	Honda	35
roadster	20	-	-	Toyota	30
sports car	10	-	-	-	-

Theorem 2 (Global Pruning Optimality). *Choosing s_i ($1 \leq i \leq t$) maximizing expected pruning cardinality at each iteration leads to optimal sampling $S = \{s_1, \dots, s_t\}$, which maximizes overall expected pruning cardinality $\text{SUM}(S) = \left| \text{SKY}(A, \succeq_{\text{Par}}^{(0)}) \right| - \left| \text{SKY}(A, \succeq_{\text{Par}}^{(t)}) \right|$.*

We now develop our framework based on Theorem 2. We name our framework as *MaxPrune*, where overall expected pruning cardinality is maximized by identifying an optimal sample at each steep. We briefly describe how Framework *MaxPrune* works. As an initial state, all current skyline objects are initialized as the entire set of data instances. Framework *MaxPrune* then follows the following three steps— First, it selects a sample with the highest expected pruning cardinality from all current skyline objects. Second, it collects a user preference with respect to the given sample, based on which it prunes all dominated objects having a non-preferred value from the current skyline. Lastly, it updates the cardinalities in each dimension. The processes are repeated until the number of skyline objects is reduced to at most k .

To illustrate, we describe how Framework *MaxPrune* works over our example dataset in Table 2. First, we consider samples with the highest expected pruning cardinality, *e.g.*, ‘convertible’ and ‘sedan’ for *type*, ‘red’ and ‘blue’ for *color*, and ‘Ferrari’ and ‘Honda’ for *brand*. Among these, we decide to obtain a preference elicitation on ‘red’ and ‘blue’ first, since its expected pruning cardinality (Theorem 1) is the highest, *e.g.*, $\frac{1}{2}(60 + 40)$. Once the elicitation result is obtained, for instance ‘red’ $\succ_{\text{brand}}$ ‘blue’, for each object with ‘red’, we can prune out objects with ‘blue’ sharing the same remaining attribute values. For the remaining objects, we then update the cardinality of attribute values for each attribute. Framework *MaxPrune* continues this iterative process until the size of skylines is reduced to k or less. We formally state Framework *MaxPrune* as follows:

1. Initialize $\text{SKY}(A, \succeq_{\text{Par}}^{(0)})$ as the entire data set.
2. Select the most effective sample as a pair of values with the highest expected pruning cardinality (Theorem 1).
3. Elicit preference information on the sample, and according to the user preference, prune out “dominated” objects from current skylines. (Lemma 1)
4. Update the cardinality of attribute values for each attributes.
5. Repeat step 2, 3, and 4 until $|\text{SKY}(A, \succeq_{\text{Par}}^{(t)})| \leq k$.

Table 3. Parameters for Experimental Setup

Parameter	Value : Default
Database Size N	100K
Dimensionality n	[3,7] : 4
Number of distinct values m	[3,7] : 4
Retrieval Size $k(\%)$	[1,20] : 5
Skewness of Data z	[0,2] : 1
Skewness of Query Frequency z'	[0,2] : 1
Kendall τ distance d	[0,1] : 0.5

4 Experimental Evaluation

This section validates the effectiveness and efficiency of frameworks *MaxPrune* and *MaxPruneQF*³ using various synthetic datasets. Our experiments were carried out on a Intel(R) Xeon(TM) machine with 3.20 GHz dual processors and 1GB RAM running LINUX. Our algorithms were implemented in C++ language.

4.1 Data and Preference Generation

For the purpose of extensive evaluations, we generate synthetic datasets by varying experiment settings, including the data size N , the number of distinct attribute values m , and the user-specified retrieval size $k\%$ (of N), as described in Table 3. Especially, we randomly generate m distinct attribute values and query frequency for each dimension, according to Zipfian and Uniform distributions, varying the skewness from $z = 0$ (uniform) to $z = 2$. Note that we generate datasets to follow the assumption described in Section 3.1, *i.e.*, there exists at least one object for every attribute value combination. Specifically, we first populate one object for every alternative, and then generate N objects according to Zipfian distribution.

We then generate user preferences and interactions to compare our frameworks *MaxPrune* and *MaxPruneQF* with and without a priori knowledge, *e.g.*, query frequency $\mathcal{Q}$. In particular, we randomly generate query frequency for each dimension based on Zipfian distribution with the skewness $z' = [0, 2]$. We then follow user interactions on $\mathcal{Q}$, to prefer values in the descending order of query frequency for each dimension. Note, if this descending order of query frequency coincides with the descending order of cardinality, *i.e.*, when two orders are perfectly correlated, the behavior of *MaxPrune* and *MaxPruneQF* will be identical. We thus observe their behavior over the varying correlations. In particular, we adopt the Kendall τ distance $d = \frac{1}{n} \sum_1^n \mathcal{K}_i(\mathcal{W}_i, \mathcal{O}_i)$ [15], a widely-adopted metrics to quantify the correlation between two orderings $\mathcal{W}_i$ and $\mathcal{O}_i$. $\mathcal{K}_i(\mathcal{W}_i, \mathcal{O}_i)$ is defined as follows:

$$\mathcal{K}_i(\mathcal{W}_i, \mathcal{O}_i) = \frac{\sum_{(p,q)} |\mathcal{W}_i(p) > \mathcal{W}_i(q) \wedge \mathcal{O}_i(p) < \mathcal{O}_i(q)|}{m(m-1)/2}$$

³ Framework *MaxPruneQF* extends *MaxPrune* to use prior knowledge, *i.e.*, realistic distribution of user preferences, as discussed in Section 3.

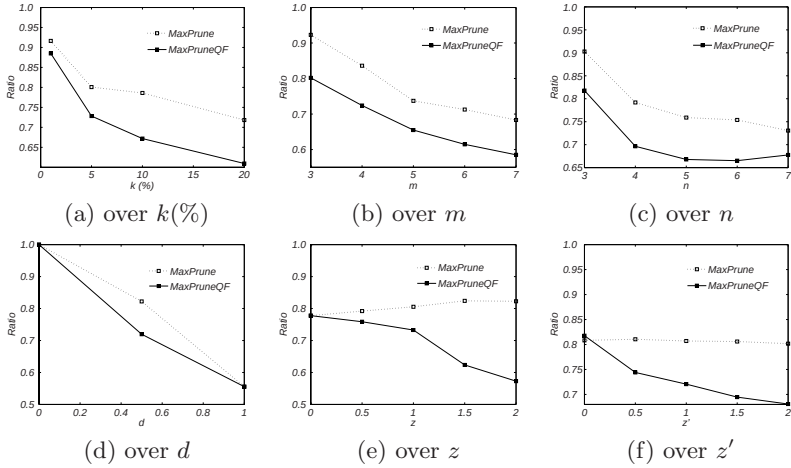


Fig. 1. Number of iterations over varying parameters

where (p, q) denotes a possible pair of values and $W_i(p)$ and $O_i(p)$ represent each position in the respective ordering.

4.2 Experimental Results

In this section, we report our experiment results validating the pruning effectiveness and efficiency of our proposed frameworks. Table 3 describes experiment settings used. In particular, we adopt the following performance metrics:

- **Effectiveness:** We use the number of iterations until we identify the k best results, averaged over 100 runs. We report relative performance against that of Framework *Random* which randomly selects a sample to present.
- **Efficiency:** We measure the runtime performance of our framework at each iteration, compared with that of Framework *Random*.

Pruning Effectiveness: Figure 1 reports the effectiveness of our frameworks over varying parameters. Note, the y -axis represents the relative number of iterations of our proposed frameworks, compared to that of *Random*.

$$Ratio = \frac{\# \text{ iterations with our framework}}{\# \text{ iterations with } Random}.$$

First, Figure 1(a) reports our results over varying retrieval size $k\%$. Observe that, Framework *MaxPrune* and *MaxPruneQF*, by minimizing user interactions, significantly outperforms Framework *Random*– For instance, when $k = 20\%$, Framework *MaxPrune* saves 28% and 38% from Framework *Random* and Framework *MaxPruneQF* respectively. Our frameworks similarly dominate the baseline

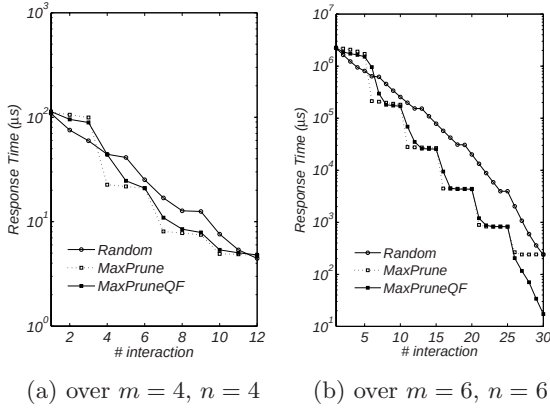


Fig. 2. Number of remaining skyline tuples

approach over m and n , which can be observed in Figure 1(b) and (c) respectively. Also, observe that our frameworks scale more gracefully compared to *Random*. That is, the performance gaps increase as m and n increase—For instance, when $m = 7$ in Figure 1(b), *MaxPrune* and *MaxPruneQF* save about 27% and 32% from *Random* respectively, while they save 10% and 18% when $m = 3$. Similarly, when $n = 7$ in Figure 1(c), this saving reaches up to 32% and 41% respectively.

Second, Figure 1(d) validates the effectiveness over varying correlation distance (quantified as a Kendall τ distance d discussed above). As already analyzed above, *MaxPrune* and *MaxPruneQF* behave identically when $d = 1$, which can be observed from Figure 1(d). As d increases, the performance gap increases, which reaches up to 55% when $d = 0.5$. Third, Figure 1(e) and (f) validate the effectiveness with respect to the skewness of datasets and query frequency, respectively. Observe from the figures that, the relative effectiveness of our proposed frameworks increase as the skewness increases, especially for *MaxPruneQF*. For instance, when $z = 2$ and $z' = 2$ Framework *MaxPruneQF* saves 43% and 32% from that of *Random*, respectively. In summary, we validate that our framework significantly outperforms *Random* in datasets with high cardinality and dimensionality, especially in the presence of high skewness in both data and query frequency.

Runtime Performance: Figure 2(a) validates the efficiency of our frameworks by reporting an average response time for each iteration over the default setting (Table 3). Note that y-axis is log-scaled. As the figure reports, the response time of our frameworks is comparable to *Random* at all iterations, which is impressive considering *Random* blindly picks a random sample. Further, our framework starts to outperform *Random*, as the number of remaining skyline objects rapidly decreases over the iterations. For instance, after the 4th iteration, our framework begins to deal with a much smaller sample pool and outperforms *Random* from this point on. Similarly, Figure 2 reports results for extended parameters with $m = 6$ and $n = 6$.

5 Related Work

We summarize related work on skyline computation and the representation of user preferences.

Skyline computation: Skyline queries have been first studied as maximal vectors in [19]. Later, Börzsönyi et al. [5] introduced skyline queries in database applications. Next, Tan et al. [23] proposed progressive skyline computation using auxiliary structures. Kossmann et al. [18] improves (D&C) algorithm, and proposed nearest neighbor (NN) algorithm. Similarly, Papadias et al. [22] developed branch and bound skyline (BBS) algorithm which achieves I/O optimal property. Meanwhile Chomicki et al. [10] developed sort-filter-skyline (SFS) algorithm leveraging pre-sorting lists, and Godfrey et al. [12] proposed linear elimination-sort for skyline (LESS) algorithm with attractive average-case asymptotic complexity. Recently, there have been active research efforts to address “curse of dimensionality” problem of skyline queries [6,7,21] using inherent properties of skylines such as *skyline frequency*, *k-dominant skylines*, and *k-representative skylines*. All these efforts, however, focused only on numerical domains with inherent orders, and did not consider skyline queries over categorical domains.

Preference foundation: For representing a variety of user preferences, Kießling [16,17] proposed a framework using binary preference relations. Similarly, Chomicki [9,10] developed a preference model, which consists of a basic preference *winnow* operator and its combinators. These preference models refer that *qualitative* models are more “intuitive” than *quantitative* models [11,14], which is consistent with our view. Meanwhile, Balke et al. [3,4,1,2] studied how to use *incomplete* preference information for skyline queries: In particular, [3,4] studied how to identify skylines over user-specified partial orders. More recently, [1] extended the notion of equivalence to include the inter-attribute equivalence, and [2] discussed a sophisticated user interface in the cooperative process of identifying partial orders. Meanwhile, Chen and Pu [8] summarizes methods eliciting user preferences. However, this framework does not address how to collect and leverage user-specific preferences. Our work helps users to elicit the most informative partial information on their preferences.

6 Future Work

We plan to extend our work in several ways. First, we can extend our framework into general environment combining numerical and categorical domains. Our technical report [20] discusses this extension in more details. Second, we want to develop new pruning heuristics that are less restricted than the one used in *MaxPrune* yet computationally feasible, to support sparse data sets. Lastly, we plan to explore how elicitation methods can make use of more complicated a-priori knowledge on the preference distribution (*e.g.*, dependencies in probability between attributes or attribute values).

Acknowledgments

This research has been partially supported by the Korean Research Foundation Granted funded by the Korean Government (MOEHRD, Basic Promotion Fund; KRF-2007-331-D00377) and the German Research Foundation (DFG) within the Emmy Noether Programme.

References

1. Balke, W.-T., Güntzer, U., Lofi, C.: Eliciting matters—controlling skyline sizes by incremental integration of user preferences. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 551–562. Springer, Heidelberg (2007)
2. Balke, W.-T., Güntzer, U., Lofi, C.: User interaction support for incremental refinement of preference-based queries. In: RCIS (2007)
3. Balke, W.-T., Güntzer, U., Siberski, W.: Exploiting indifference for customization of partial order skylines. In: IDEAS (2006)
4. Balke, W.-T., Güntzer, U., Siberski, W.: Getting prime cuts from skylines over partially ordered domains. In: BTW (2007)
5. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2001)
6. Chan, C.-Y., Jagadish, H., Tan, K., Tung, A.K., Zhang, Z.: On high dimensional skylines. 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. 478–495. Springer, Heidelberg (2006)
7. Chan, C.-Y., Jagadish, H., Tan, K.-L., Tung, A.K., Zhang, Z.: Finding k-dominant skyline in high dimensional space. In: SIGMOD (2006)
8. Chen, L., Pu, P.: Survey of preference elicitation methods. In: EPFL Technical Report (2004)
9. Chomicki, J.: Querying with intrinsic preferences. In: Jensen, C.S., Jeffery, K.G., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, Springer, Heidelberg (2002)
10. Chomicki, J., Godfery, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE (2003)
11. Fishburn, P.C.: Utility Theory for Decision Making. Wiley, Chichester (1976)
12. Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: VLDB (2005)
13. Hansson, S.O.: What is ceteris paribus preference? *Journal of Philosophical Logic* 25(3), 307–332 (1996)
14. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives. Preferences and Value Tradeoffs. Wiley, Chichester (1976)
15. Kendall, M.: A new measure of rank correlation. In: *Biometrika* (1938)
16. Kießling, W.: Foundations of preferences in database systems. In: VLDB, pp. 311–322 (2002)
17. Kießling, W., Köstler, G.: Preference SQL—design, implementation, experience. In: VLDB, pp. 311–322 (2002)
18. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB (2002)
19. Kung, H.T., Luccio, F., Preparata, F.: On finding the maxima of a set of vectors. *Journal of the Association for Computing Machinery* (1975)

20. Lee, J., You, G., Hwang, S., Selke, J., Balke, W.-T.: Optimal preference elicitation for skyline queries over categorical domains. POSTECH Technoical Report (2008), <http://ids.postech.ac.kr/~parfum/skyline.pdf>
21. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: The k most representative skyline operator. In: ICDE (2007)
22. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progessive algorithm for skyline queries. In: SIGMOD (2003)
23. Tan, K., Eng, P., Ooi, B.C.: Efficient progressive skyline computation. In: VLDB (2001)
24. von Wright, G.H.: The Logic of Preference. An Essay. Edinburgh University Press (1963)

Categorized Sliding Window in Streaming Data Management Systems

Marios Papas¹, Josep-L. Larriba-Pey², and Pedro Trancoso^{1,*}

¹ Department of Computer Science
University of Cyprus, Nicosia, Cyprus
{cs03pm3,pedro}@cs.ucy.ac.cy

² DAMA-UPC and Department of Computer Architecture
Universitat Politècnica de Catalunya, Barcelona, Spain
larri@ac.upc.edu

Abstract. For many applications, data is collected at very large rates from various sources. Applications that produce results from this data have a requirement for very efficient processing in order to achieve timely decisions. An example of such a demanding applications is one that takes decisions on stock acquisition based on the price updates that happen constantly while the market is open for transactions. Our proposed technique is a simple yet effective way to reduce the access time to the streaming data.

In this paper we propose an efficient indexing technique that improves the access time to data elements in sliding windows of streamed database systems. This technique, called *Categorized Sliding Window*, is based on splitting the data into categories and using bit vectors to avoid accesses to non-relevant data.

Our experimental results show large improvements compared with simpler techniques. For the standard Linear Road benchmark we observe a performance improvement of 3.3x for a complex continuous query. Also relevant is the fact that 90% of the performance improvement is achieved with only 65% of the maximum number of categories, which represents a memory overhead of only 13.5%.

1 Introduction

The increasing capability of efficiently collecting and transferring large amounts of data has triggered a new set of applications that are able to process information in real-time and produce results to support decision making. The amount of data that usually flows per unit of time is so large that storing it for future analysis is in many cases out of the question. As such, the processing needs to be performed as the elements of the *data stream* arrive. As the processing can not be done over all incoming data, it is instead performed over a window of data that changes over time, the *sliding window*. Due to the time constraints imposed

* Pedro Trancoso is member of the HiPEAC Network of Excellence (EU FP7 program).

by this online real-time operation, traditional database management systems are not able to handle the queries on the streamed data. As such, new Streaming Database Management Systems have been proposed [2,7]. Nevertheless, the access to the data elements of the stream in such systems is still a challenge [9,13]. In some cases, the systems analyze only part of the data [8]. In other cases, potential query results (e.g. statistical metrics such as max, min, avg, etc.) are updated as data is added to the window statistics [11]. Finally, there is some work on implementing indexing techniques for sliding windows [12].

In this paper we propose a new indexing technique for the incoming streamed data, the *Categorized Sliding Window* (CSW). This technique uses multiple bit vectors as an efficient way to index the streaming data, which is stored in a sliding window. Each bit vector represents a *category* of data, hence the name of the technique. A *category* could be a range of values for a certain attribute field of the streaming data. The bit vectors have as many bits as there are elements in the sliding window. If the N th bit has value “1” in the bit vector of category “A”, then the corresponding N th element in the sliding window belongs to that same category “A”. Figure 1 depicts an example of the CSW structures.

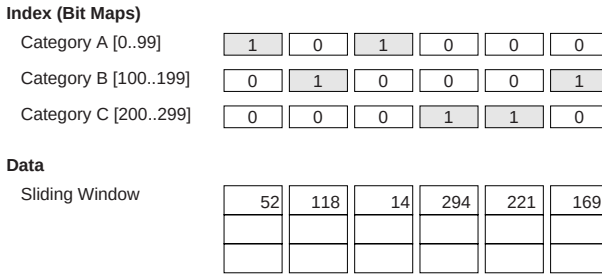


Fig. 1. Categorized Sliding Window Structures

To validate the proposed structure, an implementation of the *Categorized Sliding Window* is presented. This implementation is evaluated on a real system using both synthetic data and data from the standard Linear Road benchmark [6]. The queries tested are also part of the same benchmark. The results show that while the memory overhead of the proposed structure is only around 20% for 100 categories, the performance improvement, compared to the sequential scan baseline case, can be up to 3.3x for the Linear Road benchmark queries. Also relevant is the fact that 90% of the performance improvement is achieved with only 65% of the maximum number of categories. This performance benefit is achieved with a memory overhead of only 13.5%.

This paper is organized as follows. The *Categorized Sliding Window* technique is presented in Section 2, the experimental setup and results are shown in Sections 3 and 4, respectively. The relevant related work is presented in Section 5 while the conclusions for this work are discussed in Section 6.

2 Categorized Sliding Window

Given that the data in the sliding window changes frequently, traditional indexing techniques used in DBMS are not applicable. Therefore, searching data is usually done by performing a simple sequential scan over all the elements in the sliding window. We call this technique the *Baseline Sliding Window* (BSW). Each element of the window is a pointer to the element of the data structure holding the complete information, *i.e.* all the data fields of the incoming data. Elements are added to the window as they arrive from the input stream and are removed from the window as their “lifetime” expires. This happens when either the list has reached its maximum number of pre-determined elements or the element has stayed in the list for more time than what had been pre-defined.

The contribution of this work is to provide a mechanism that reduces the processing time by accessing the data in a faster way. To achieve this goal we propose an indexing capability that is efficiently updated even with streaming data. Newly incoming data do not trigger time consuming updates to the index structure as opposed to traditional indexing techniques, which need to be rebuilt whenever the data changes.

With the proposed technique, the incoming data is *logically divided* into different categories. Therefore, we call the proposed approach *Categorized Sliding Window* (CSW). Categories may be defined, for example, as ranges of values of a certain attribute. For each of these categories, there is a *bit vector* with as many bits as there are elements in the sliding window. Therefore, there is a one-to-one relation between the bits in the bit vector of each category and the elements in the sliding window. This bit vector is used to index the data. The result is that whenever a query is performed, if the query is looking for a particular data that is found on a certain category, the search is limited to the elements belonging to that category. This technique helps in accessing the relevant data to the query in a faster way. It is important to note that a “1” in the bit vector does not necessarily represent an entry for the data being looked for. This is due to the fact that the bit vector represents all the values in the corresponding category. Therefore, when looking for a certain value, the bit vector may have “1” values that are actually representing other values of that same category. Such values are called *False Positives*.

The maintenance of the bit vector results in a memory overhead. On the one hand, a CSW with a single category is the structure with the smallest memory overhead but conceptually performs the same as a BSW, with the difference that the access through the bit vector results in a penalty compared to the simpler access to the BSW structure. On the other hand, a CSW with as many categories as distinct values in the categorized field, is the structure with the most efficient accesses, *i.e.* with no false positive accesses, but is also the one with the largest memory overhead due to all the bit vectors. An efficient CSW structure is one with a configuration that achieves a good balance between the number of false positives and the memory overhead.

The next paragraphs describe the implementation details for the main operations of CSW.

Allocation. There are two basic structures required for the implementation of CSW. First is a buffer that is allocated in order to hold all data elements of the sliding window. This buffer is to be used as a *circular buffer*. In order to support that feature, two counters are needed: one indicating the head of the buffer (*Start*) and another one indicating the tail of the buffer (*Finish*). Second is a table with N bit vectors, where N is the number of supported categories. Each supported category has a corresponding logical *Category Bucket*, *i.e.* a bit vector. Each *Category Bucket* can be mapped into chunks that we call blocks. Each block has the size of the processor register, allowing for optimized bit-wise operations.

Insertion. First we need to map the category identifier of the tuple that we are inserting to a Category Bucket. We achieve this by applying a *mapping function* to the category attribute of the incoming tuple. Then we need to find within the bit vector of that Category Bucket, the block that contains the bit in position *Finish*, that corresponds to entry in the buffer used to store the tuple. Notice that the block, *i.e.* set of contiguous bits, is the granularity of the operations performed on the bit vector. If the buffer is not completely full we insert the new element in the buffer position pointer by the counter *Finish* and we advance that counter to the next position. Finally, we update the bits of the block we retrieved previously in order to have a “1” in the position corresponding to the entry where the tuple was inserted in the buffer. If the buffer is full, then we insert the tuple in the buffer position pointed by the *Finish* counter, we update the bits of the block we retrieved, and we advance both the *Start* and *Finish* counters by one position. The tuple that is being replaced can either be deleted or stored somewhere else but the bit in the bit vector of the corresponding category of the replaced tuple must be reset to “0”.

Retrieval. In order to retrieve a specific data element we first need to determine the category which the requested data belongs to. This is done using a *mapping function*, which may be implemented as a *hash function*, that takes as input the requested field value and returns the corresponding category. Then, we search over the bit vector of the corresponding category and for every bit set to “1”, we access the corresponding data element in the window. The search on the bit vector is done in a blocked manner, *i.e.* bits are analyzed in blocks. If a block of bits contains only “0” bits (simple check operation), then we will immediately continue the search on to the next block. Otherwise, we will analyze the different bits within the block in order to determine which bits are set to “1”. The advantage of first applying the simple check operation is that there will be enough categories so that data is spread across the different categories and consequently the bit vector will be sparse, *i.e.* many blocks contain only “0”s. Every time a “1” bit is found, the corresponding data element in the window is checked to verify that it matches the requested field value and is not a false positive.

3 Experimental Setup

In order to test the proposed technique, a system containing all the structures as discussed in the previous section was implemented. The system implements both BSW and CSW techniques to store and query streaming data. The implementation was done in *C++* and compiled using *g++* with the *-O3* optimization flag option. For this work we used gcc version 4.1.1. The machine on which we compiled and executed our implementation was a typical home desktop system using an Intel Pentium 4 Processor 640 at 3.2GHz with 2MB L2 Cache and 800MHz Front Side Bus with Hyper Threading Technology. The main memory size is 1GB and the hard drive is an 80GB SATA at 7200RPM. The operating system used was Fedora Core 5.

The benchmark used to evaluate the proposed technique was the Linear Road Benchmark [6]. The Linear Road benchmark was designed to stress Stream Data Management Systems (SDMS). The benchmark models the data that can be gathered from the traffic in expressways and the possible queries that may be posed to this data. The benchmark defines four queries: *Toll Notification*, which is a *Continuous Query*, and *Account Balance*, *Daily Expenditures*, and *Travel Time Estimation*, which are *Historical Queries*.

For our experiments, we used as input data set 1 million *Position Reports*, *i.e.* the basic data structure for the *Toll Notification* query. Each such structure is composed of 15 fields, where each field was 4-Byte. The data populated in these structures is of two types: *Synthetic Data (SynD)* and *Simulated Data (SimD)*. For the *Synthetic Data*, the data values were created in order to satisfy certain specific criteria, such as the degree of selectivity (ratio between the number of data elements that satisfy the query criteria and the total number of data elements). For the *Simulated Data*, the data values were created from Linear Road Benchmark [6] specifications. In order to test our proposed technique, we categorized the data based on the position reports in the segment field (*miSeg*), which has 100 discrete values for each expressway. This field represents the segment of the expressway on which the vehicle emitted its position report. In our implementation we only considered one simulated expressway.

For our experiments we used two types of queries. The *Simple Query (SQ)* is defined as how many position reports were received from segment X of the expressway 0 that are stored inside the sliding window scope. This represents a subquery from a more complex Linear Road Benchmark query. The *Complex Query (CQ)* represents the *Toll Notification* query of the Linear Road Benchmark. This query is triggered whenever a position report is received for which the previous position report of the same car was located in a different segment.

4 Experimental Results

4.1 Memory Space

One important factor in evaluating the proposed technique is to determine its memory overhead. In the case of BSW, the allocated memory depends on only

two factors: the size of the data structure used to store the Tuple (*Timestamp* and *Category ID*) and the Window Size, *i.e.* how many tuples we want to store inside the window. In the case of CSW, the memory allocated depends on one additional factor, the number of Category Buckets of the CSW.

Considering the size of a CSW entry to be 60-Byte, the size of a *Position Report*, the allocated memory needed for CSW in comparison with BSW is 20.8%, 41.6%, and 62.5% more for 100, 200, and 300 Category Buckets, respectively.

4.2 Query Selectivity

From the characteristics of the CSW technique, we expect that as the selectivity is reduced, the performance benefits of CSW against the existing BSW will increase. To prove this point we have tested the simple query on synthetic data, therefore controlling the selectivity values. The results of the speedup achieved by CSW compared to BSW is shown in Figure 2.

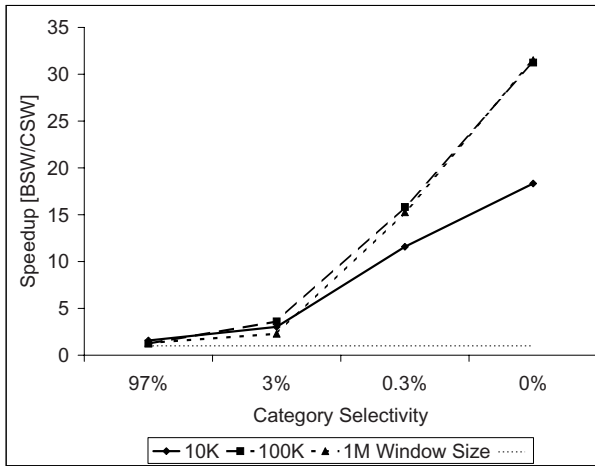


Fig. 2. Speedup for different selectivities and window sizes

From Figure 2 we can observe that, as expected, as the selectivity decreases, the speedup increases. This effect is more relevant as the data window size increases, because the speedup observed for 100K and 1M data window sizes is much larger than for the 10K data window size. While it is important to notice that the speedup achieved is larger than 30% for the smallest selectivity, it is also important to note that the speedup is always larger than one (the dotted horizontal line on the chart) even for the larger selectivity values.

4.3 Window Size

Another interesting factor to consider in the analysis is the data window size that is handled by the techniques. To achieve this goal we analyzed the performance

of CSW and BSW for window sizes ranging from 10K to 100K elements. Also, we considered different setups, the synthetic data and simple query (*SynD-SQ*) and also the simulated data with both the simple (*SimD-SQ*) and complex (*SimD-CQ*) queries. Although not depicted due to space limitations, the results show that for every configuration and window size, the speedup is always larger than one. It is also relevant to notice that although stable results can only be produced with the synthetic data as the selectivity of the simulated data changes for different window sizes and runs, the speedup achieved by the simple query is much larger than the one achieved by the complex query. Also, the trend seems to indicate that the speedup does not change much with the increase of the window size although for the simulated data and simple query the speedup seems to increase as the window size increases.

4.4 Number of False Positives

Another relevant factor is the number of false positives, *i.e.* “1” values in the bit vectors that actually represent real values that do not belong to the value being searched. To study the effect of false positives on the CSW we have tested this technique with synthetic data that represents a range of selectivity values and false positive rates. The results are depicted in Figure 3.

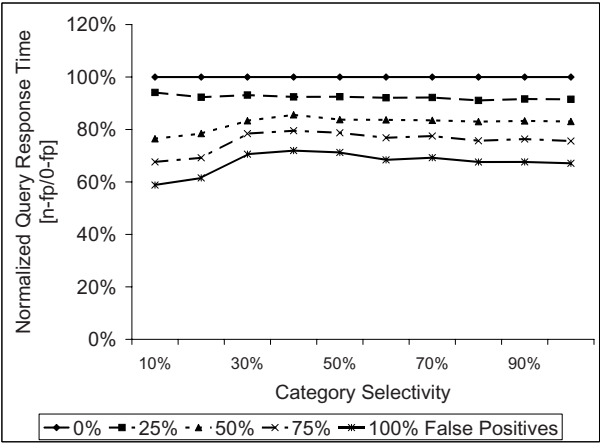


Fig. 3. Normalized query response time for different for selectivity and false positive rates

In Figure 3 we can observe that for selectivity values larger than 30%, the normalized response time is independent of the selectivity value. Also relevant is the fact that the difference in the response time is at most approximately 30% between a false positive rate of 100% (all entries are false positives) and one of 0% (no entries are false positives).

4.5 Number of Categories

One of the most important questions that CSW is faced is how many category buckets, or categories, are necessary in order to obtain good performance. The tradeoff in this case is that the larger the number of buckets, the better the performance should be. At the same time, the larger the number of buckets, the larger the amount of memory consumed with the extra data structures. The speedup achieved by CSW against BSW for different number of category buckets is show in Figure 4. The experiments are performed for a data window size of 100K elements and we tested the complex query as described in Section 3.

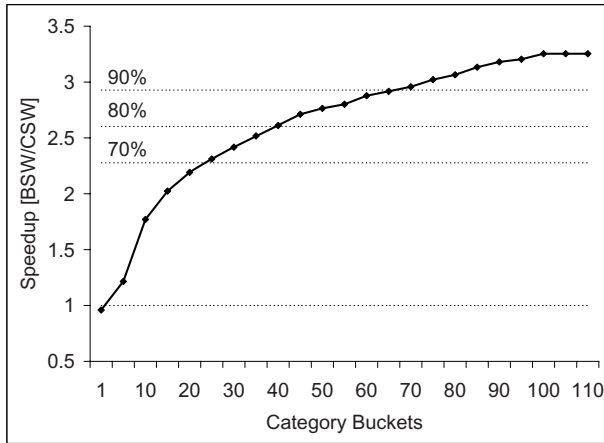


Fig. 4. Speedup for different for categories buckets

In Figure 4 we can first observe that other than the case for a single category bucket, the speedup is always larger than one (lower horizontal dotted line). Also important is the fact that, as expected, the larger the number of buckets, the larger the speedup achieved. In this case, the speedup reaches 3.3x for more than 100 buckets, in a maximum of 100 distinct values for that specific category. In addition to the line of speedup equal to one, we have drawn three more horizontal dotted lines: 90%, 80%, and 70% of the maximum speedup. It is interesting to notice that 90% of the maximum speedup is achieved for 65 category buckets, which is only 65% of the maximum buckets. If we settle for 80% of the maximum speedup, this is achieved for 40 category buckets or only 40% of the maximum buckets. As such, the results show that it is possible to achieve a very high speedup with a small memory overhead.

5 Related Work

The need to process large amounts of dynamically evolving data demands for new systems and query languages. AQuery [15] is a query language based on

the relational data model that can support order-oriented queries. Aurora [1] is a database system that is able to process continuous streams of data as well as combine historical with real-time data. Aurora provides the basic data-stream processing functionality. Aurora has lead to Borealis [2,3], a distributed stream processing engine. STREAM [5] is a data stream management system which addresses the issue of data management and query processing in the presence of multiple continuous and time-varying data streams. The query language supported by STREAM is known as CQL [4]. TelegraphCQ [10] is a relational-based system that is able to process continuous streams of data and is the implementation of the Telegraph dataflow engine. Tribeca [16] is a system that applies a compiled query to arbitrarily long streams of data. PIPES [14] is an infrastructure that supports the implementation of data stream management systems for continuous data-driven query processing.

An important factor in the performance of stream query processing is the ability to efficiently access the relevant data from the sliding window. It is possible to address this issue in various ways. One way is to keep updating certain statistical values for each data element that is inserted into and deleted from the window [11]. This way, a query on such a value may be answered without accessing the data at all. This is very efficient but not general enough. Another approach is to use only a sample of the data in order to determine the outcome of the query [8]. While this approach is also efficient, the accuracy of the results is not guaranteed. Some initial projects used traditional indexing structures on the sliding window data elements. The problem with this approach is that updating the index structures is a costly operation that, in the case of streaming data, needs to be performed very often as the data elements in the window are replaced constantly. To avoid this problem you may decide to update the indexes only after n elements have been changed. Alternatively, Golab *et al.* [12] have suggested to partition the sliding window and create an index per partition, as opposed to a global index for all data. This way, the index that needs to be built is only the one corresponding to the new partition, and only at the point that there have arrived enough data to form a new partition. This method is more efficient than the previous ones but still not enough for high rates of data. The simplicity of the bit vector-based category approach allows it to perform efficiently and scale well to high rates of streams of data.

6 Conclusions

In this paper we propose an efficient indexing technique that improves the access time to data elements in sliding windows of streamed data, which we call *Categorized Sliding Window*. This technique is based on splitting the data into categories and using bit vectors to avoid accesses to non-relevant data. Experimental results show large improvements compared with traditional techniques in stream data management systems. For the standard Linear Road benchmark we observe a performance improvement of 3.3x for a complex continuous query.

Also relevant is the fact that 90% of the performance improvement is achieved with only 65% of the maximum number of categories, *i.e.* a memory overhead of 13.5%.

Acknowledgments

Josep-L. Larriba-Pey would like to thank Generalitat de Catalunya for its support through grant number GRE-00352 and Ministerio de Educación y Ciencia of Spain for its support through grant TIN2006-15536-C02-02.

References

1. Abadi, D.J., et al.: Aurora: a New Model and Architecture for Data Stream Management. The VLDB Journal 12(2), 120–139 (2003)
2. Abadi, D.J., et al.: The Design of the Borealis Stream Processing Engine. In: Proc. of CIDR 2005 (2005)
3. Ahmad, Y., et al.: Distributed Operation in the Borealis Stream Processing Engine. In: Proc. of SIGMOD 2005, pp. 882–884 (2005)
4. Arasu, A., Babu, S., Widom, J.: The CQL Continuous Query Language: Semantic Foundations and Query Execution. The VLDB Journal 15(2), 121–142 (2006)
5. Arasu, A., et al.: Stream: the stanford stream data manager (demonstration description). In: Proc. of SIGMOD 2003, pp. 665–665 (2003)
6. Arasu, A., et al.: Linear Road: A Stream Data Management Benchmark. In: Proc. of VLDB 2004, pp. 480–491 (2004)
7. Avnur, R., Hellerstein, J.M.: Eddies: Continuously Adaptive Query Processing. SIGMOD Record 29(2), 261–272 (2000)
8. Babcock, B., Datar, M., Motwani, R.: Sampling from a Moving Window Over Streaming Data. In: Proc. of SODA 2002, pp. 633–634 (2002)
9. Babcock, B., et al.: Models and Issues in Data Stream Systems. In: Proc. of PODS 2002, pp. 1–16 (2002)
10. Chandrasekaran, S., et al.: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In: Proc. of CIDR 2003 (2003)
11. Datar, M., et al.: Maintaining Stream Statistics over Sliding Windows. SIAM Journal of Computing 31(6), 1794–1813 (2002)
12. Golab, L., Garg, S., Ozsu, M.T.: On Indexing Sliding Windows over On-Line Data Streams. In: Proc. of EDBT, pp. 712–729 (2004)
13. Golab, L., Ozsu, M.T.: Issues in Data Stream Management. SIGMOD Rec. 32(2), 5–14 (2003)
14. Krämer, J., Seeger, B.: PIPES: A Public Infrastructure for Processing and Exploring Streams. In: Proc. of SIGMOD 2004, pp. 925–926 (2004)
15. Lerner, A., Shasha, D.: AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments. In: Proc. of VLDB 2003, pp. 345–356 (2003)
16. Sullivan, M.: Tribeca: A Stream Database Manager for Network Traffic Analysis. In: Proc. of VLDB 1996, p. 594 (1996)

Time to the Rescue - Supporting Temporal Reasoning in the Rete Algorithm for Complex Event Processing

Karen Walzer, Matthias Groch, and Tino Breddin

SAP Research CEC Dresden, Germany
`karen.walzer@sap.com`

Abstract. Complex event processing is an important technology with possible application in supply chain management and business activity monitoring. Its basis is the identification of event patterns within multiple occurring events having logical, causal or temporal relationships.

The Rete algorithm is commonly used in rule-based systems to trigger certain actions if a corresponding rule holds. The algorithm's good performance for a high number of rules makes it ideally suited for complex event detection. However, the traditional Rete algorithm does not support aggregation of values in time-based windows although this is a common requirement in complex event processing for business applications.

We propose an extension of the Rete algorithm to support temporal reasoning, namely the detection of time-based windows by adding a time-enabled beta-node to restrict event detection to a certain time-frame.

1 Introduction

Complex event processing (CEP) is a technology to monitor and control information systems driven by events. It is applied, for instance, in business process or supply chain monitoring to detect complex events consisting of single events with logical, temporal or causal relationships. Many designs have been proposed for this purpose, for instance [1] uses a finite state automaton for event detection and Li and Jacobsen [2] showed the efficiency of the Rete algorithm to match a high number of rules and thus complex events.

Traditionally, the Rete algorithm [3] is used for production-based logical reasoning, matching a set of facts against a set of inference rules. A rule defines a predicate and the action that should take place, if the predicate holds. This corresponds to defining a complex event combining single events and executing an action such as the creation of a new event when the complex event is detected. The algorithm's linear complexity makes it capable of dealing with a high number of complex event patterns.

Business applications often require the definition of complex events which consider temporal relationships among single events. Then, time-based windows are utilized to ensure that event detection takes place only for the time of an events' relevance. This is achieved by restricting processing to events matching

a pre-defined time-frame, e.g. occurring within 1 hour or between 1 and 2 pm. It is often desirable to aggregate events, for instance to calculate the average of a certain event data item over time, such as a temperature sensor value. Time-based windows allow for this event aggregation over a pre-defined time. However, the traditional Rete algorithm does not support such temporal operators, but is limited to operations such as unification and predicate extraction. A simple matching using comparisons with timestamp attribute values is possible and extensions to allow for detection of relative relationships have been suggested, e.g. in [4, 5]. However, sophisticated temporal operators such as sliding windows were not regarded so far.

In this paper, we present an extension of the Rete algorithm to support temporal reasoning using time-based windows and thus to enable complex event processing. Our method to support temporal constraints in the Rete algorithm significantly extends existing work [2, 6] for the detection of event patterns using time-based sliding windows. It uses an incremental window approach to continuously update the current window and its aggregation values.

The remainder of this paper is organized as follows. We present an overview of related work in Section 2. Then, we continue with a definition of terms and a brief description the Rete algorithm in Section 3. Section 4 presents an overview of our approach for time-based sliding windows in Rete and finally Section 5 concludes the paper.

2 Related Work

This section introduces related work in the area of temporal support for the Rete algorithm.

The Padres [2] and the ILOG JRules [4] event processing systems are based on the Rete algorithm. In both cases, the Rete algorithm is extended to incorporate clocks. Timestamps are used and *before* and *after* predicates are introduced. However, the realization of time-based windows is not addressed.

Gordin and Pasik [6] describe a method to support set-oriented methods for forward chaining rule-based systems. The presented ideas are extended in our work to support event aggregation in Rete.

In [7], a traditional production system based on the Rete algorithm is extended with temporal reasoning by storing past and developing events in a temporal database, a so-called time map. An interval time representation is used. The system supports detection of events occurring *during*, *before* or *after* other events. It is further possible to model uncertain relationships. However, the semantics of the operators as well as the conceptual details remain unclear. It is not stated whether the start or the end time-point of the interval are used for the *before* and *after* operators. Time-based windows are also not considered.

Teodosiu and Pollak [8] present a method to remove obsolete temporal facts where the used Rete network is extended with timers in order to discard events

after a specified time interval elapsed. Their concept can be used to discard events after they are no longer part of a sliding time-based window.

3 Temporal Support in Rete

In the following, we present our notion of instantaneous and complex events and introduce the Rete algorithm.

3.1 Definitions

Events indicate a state change of the world. They are n-tuples containing an arbitrary number of data items. For instance, an *OrderArrival* event contains data related to an arriving order, e.g. the customer name and address, the ordered items and their quantity as well as a timestamp denoting the occurrence time of the order.

Let $\mathbb{T} = (T; \leq)$ be an ordered time domain. Then, let $\mathbb{I} := \{[t_s, t_e] \in T \times T \mid t_s \leq t_e\}$ be the set of time intervals with t_s as start and t_e as the end time-point of the interval. Let $\mathbb{D}$ be the set of atomic values where atomic values are elementary data types, such as strings. Then, let $\mathbb{E} := \{(k_1, \dots, k_n, k_{n+1}, ts, te) \mid k_i \in \mathbb{D}, [ts, te] \in \mathbb{I}\}$ be the *set of events*.

An event is characterized by the time of its occurrence, which is stored as the event's timestamp. *Instantaneous events* are single events which occur at a certain point in time. They have a duration of zero, $t_s = t_e$. The aforementioned *OrderArrival* event is an example of an instantaneous event.

Complex events describe the occurrence of a certain set of events (instantaneous or complex) having relationships defined using logical and/or temporal operators. These operators commonly include conjunction, disjunction and negation as logical operators and can include temporal operators such as *before* and *after* to define the order of events. The supported operators vary for the different CEP systems. We consider the timestamp of a complex event as an interval consisting of a start and end point for each event to avoid the unintended interpretation occurring for time-point semantics [9, 10]. This notion follows [11] and [12]. It allows the usage of Allen's [13] thirteen temporal operators to determine the relationship between two events having interval timestamps. Consequently, complex events always have a duration, i.e. $t_s < t_e$.

3.2 The Rete Algorithm

The Rete algorithm [3] is a pattern matching algorithm traditionally used for production-based logical reasoning systems. Its aim is to match a set of facts against a set of inference rules (productions). *Facts* reside in the *working memory* and are n-tuples containing any number of data items. They represent information on something that is the case in the world. Facts are valid until they turn out to be false and are changed or retracted from the working memory. A

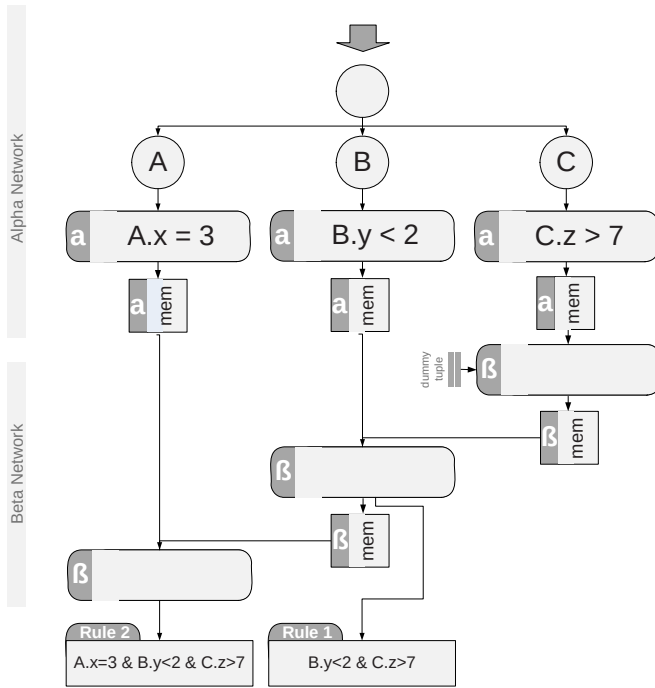


Fig. 1. Example Rete network for two rules

rule contains a premise stating conditions to be met by fact data items and a set of actions to be triggered if the premise holds.

The algorithm creates an acyclic network of the rule premises, the so-called Rete network. Figure 1 shows an example for a network for two rules with a root node (a Rete tree). The *Rete network*, starts with a root node which is split into the *type nodes* which distinguish between different facts. Then, an *alpha node network* is typically followed by a *beta-node network*. Whenever the working memory is changed, i.e. facts are "asserted", "retracted" or "updated", a *working memory element* (WME) is created for the changed fact and then propagated in a forward-chaining fashion through the network nodes from the root to the leaf nodes. Thereby, alpha-nodes perform simple conditional tests, i.e. they act as a filter by passing only the matching WMEs to the next node. At the end of the alpha node network, the resulting WMEs matching all previous nodes are stored in the alpha memory.

Beta-nodes perform joins by combining different WMEs, typically WME lists (from now on called *tuples*) coming from a beta memory with individual WMEs from an alpha memory. A new WME in the input alpha memory leads to a right activation on the beta node. Then, the new WME is compared to specific WMEs of each tuple of the beta input memory. The specific WMEs to be used are specified in the join criteria. When a new tuple is added to the input beta memory, a left activation of the beta-node takes place, specific values of a predefined set

of the new tuples are compared to particular values of each WME in the alpha memory. Upon an occurring match, a new tuple representing the match is added to the beta memory of the beta-node to be passed to subsequent beta nodes or to a terminal node (action trigger).

In other words, beta nodes store partial matches of rules to avoid re-evaluation. When a tuple has reached the end of the beta node branch, it is passed to the terminal node. It represents a complete match of the facts contained in the tuple and results in the execution of the corresponding actions.

4 Proposed Solution

In the following, we will describe the assumptions underlying our system and its semantics. Then, we will introduce a possible approach to sliding window definition in Rete.

4.1 Preliminaries

We assume a loosely-coupled system with a global clock. The occurrence of a complex event can be viewed as the occurrence of a combination of instantaneous events across distributed systems. In our system, the instantaneous events are transmitted asynchronously to the central CEP engine which is based on the Rete algorithm. Lamport's happened-before relation [14] holds. As stated in [15], Lamport's happened-before relation does not always hold. However, techniques to deal with the related issues are known, for instance [16] is using heartbeats to overcome them.

4.2 Event Aggregation Using Time-Based Windows

A window is a limitation of the view on data, i.e. instead of considering all input elements, only an extract of them is considered. Two common types of windows are distinguished: *tuple-based* and *time-based windows*, cf. to [17, 18]. Tuple-based windows limit the view to the last n input items whereas time-based windows provide only elements which arrived within a fixed time span. We focus on time-based windows with relative time-constraints, e.g. 5 minutes. We do not support windows with given absolute time constraints, e.g. a window to occur between 1 and 2 pm.

We define a time-based window as the set of events whose end time-point is in a given window interval, i.e. $\omega(t_{s_w}, t_{e_w}) := \{x \in \mathbb{E} | t_{e_x} \geq t_{s_w} \wedge t_{e_x} \leq t_{e_w}\}$ where t_{s_w} denotes the start and t_{e_w} the end time of the window. The *size of a window* $d_w = d(t_{e_w}, t_{s_w})$ states its duration in time units, e.g. 5 minutes. It is defined by the user for each window. Using this definition means that an event can start before the window in which it is considered.

Assuming, the current time is t_0 , then the boundaries of the *last finished window* can be calculated as follows. The *end time* of it is the current time, i.e. $t_{e_w} = t_0$. The *start time* is the difference between its end time and the window size, i.e. $t_{s_w} = d(t_{s_w}, t_{e_w})$. The data items with end timestamps between the start

and the end time are part of the window. Using this calculation, the window is sliding, i.e. its boundaries are constantly shifted with ongoing time. It is referred to as a *sliding time-based window*.

The Rete algorithm performs eager evaluation, i.e. the node network's inner state is updated with every inserted, modified or retracted tuple or fact, respectively. Hence, we currently consider only sliding time-based windows without a slide parameter. For window evaluation, we pursue an incremental approach, where the current window is continuously updated depending on incoming events and passing time. The resulting changes, i.e. the addition or deletion of window elements, are propagated to successive nodes in the Rete tree. This approach is suited for eager evaluation.

We extend the existing beta nodes with features for window evaluation and event aggregation. Using a beta-node for this purpose allows for the evaluation of windows for instantaneous events as well as for tuples. From now on, we will refer to these extended nodes as *time-driven aggregation nodes* (TDA). Besides the join-function of beta nodes, TDAs are capable of keeping track of the current state of a time window as well as performing on-the-fly aggregation functions to the elements of the window. A TDA node is used just as a normal beta-node in the Rete network. Depending on the rule definition, either its aggregation function, the window function or both are used. For the last case, a TDA node first performs the join of the incoming tuples and WMEs, then it checks for the window constraints and finally performs the aggregation function.

The events that should be considered for the window and/or for the aggregation need to be specified explicitly, e.g. consider the rule $IF(A.x = 3 \wedge B.y < 2 \wedge window[5min, AVG(C.z) > 40](C.z > 7)) THEN X$. It describes the conjunction of particular events A and B with an average value of an attribute of event C . The average is calculated time-frames of for 5 minutes using all attributes with $C.z > 7$. The resulting Rete tree would look like in Figure 1, except that the first beta-node (rightmost) would be a TDA node. This means, the TDA node is inserted right below the alpha node of event C and obtains the WMEs which have matched the filter criteria $C.z > 7$.

The process of the window evaluation will be described in the following. Our approach is inspired by the work of Gordin and Pasik [6] as well as Ghanem *et al.* [19].

Tuples (from the left) and WMEs (from the right) arriving at the TDA are filtered depending on whether their end timestamps are outside the window bounds or not. The stored window boundaries are adjusted continuously in the TDA node. Every time an event $x \in \mathbb{E}$ arrives at the node, the current window end time t_{e_w} is set to the arrival time of that event. The corresponding window start time can easily be determined by simply subtracting the window size from the window end time $t_{s_w} = d(d_w, t_{e_w})$. The arrived event is propagated to be considered in the aggregation function or in the child nodes of the TDA node. Furthermore, each new event is sent to a garbage collection (GC) thread which creates a callback entry in a queue for it in order to discard the event once it is out of the window bounds. The time after which the event can be discarded

is the window size. After this time passed, the positive event is out of window-bounds and a negative event is send to the TDA node by the GC thread. Then, the TDA node sends a message that the corresponding event can be discarded to its child nodes. If the child nodes, e.g. other windows, do not need the event anymore, the WME reference counter for the event can be decreased by one. If no references exist any longer, i.e. the reference counter is zero, the WME can be deleted. In any case, the aggregation function is updated, when an event is outside a window.

The TDA node keeps track of the current state of the aggregation function. For instance, in case of the *average* function, it stores a double value representing the sum of the event attribute of interest and an integer denoting the number of events contributing to the aggregated value. With every arriving or expired event, the beta memory and the aggregation result are updated.

Whenever, an event is no longer part of any sliding windows, it can be discarded from the alpha/beta-memory of the TDA node and possibly also from the working memory. Consequently, the event can no longer fulfill its temporal constraints. The other nodes in the Rete network can be informed to determine if they can also discard the event. Garbage collection mechanisms based on this sliding window timeout, a default event lifetime or a calculated lifetime based on an event's relative relationships are outlined in [8,5].

5 Conclusion

We have presented concepts of how the Rete algorithm can be extended with the detection of event patterns containing time-based sliding windows by the introduction of time-enabled beta-nodes.

We are currently evaluating the proposed concepts by extending the business rule management system JBoss Drools [20] with support for sliding windows. The current version of the Drools extension is online available at [21].

In conjunction with the detection of relative temporal constraints, event garbage collection [5] and the original features of the Rete algorithm, the proposed concepts form a good basis for temporal reasoning in Rete.

Acknowledgements. We wish to acknowledge Maik Thiele, Michael Ameling and Thomas Heinze for helpful comments on earlier drafts of this paper. Furthermore, we thank the Drools developer team, especially Edson Tirelli, for the interesting discussions and their contribution to the presented concepts and the ongoing implementation.

References

1. Demers, A.J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W.M.: Cayuga: A general purpose event monitoring system. In: CIDR, pp. 412–422 (2007)
2. Li, G., Jacobsen, H.A.: Composite subscriptions in content-based publish/subscribe systems. In: Middleware, pp. 249–269 (2005)

3. Forgy, C.: Rete: A fast algorithm for the many patterns/many objects match problem. *Artif. Intell.* 19(1), 17–37 (1982)
4. Berstel, B.: Extending the RETE Algorithm for Event Management. In: *TIME 2002: Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME 2002)*, Washington, DC, USA, vol. 49. IEEE Computer Society, Los Alamitos (2002)
5. Walzer, K., Breddin, T., Groch, M.: Relative Temporal Constraints in the Rete Algorithm for Complex Event Detection. In: *DEBS 2008: 2nd International Conference on Distributed Event-Based Systems* (to appear, 2008)
6. Gordin, D.N., Pasik, A.J.: Set-oriented constructs: from Rete rule bases to database systems. In: *SIGMOD 1991: Proceedings of the 1991 ACM SIGMOD international conference on Management of data*, pp. 60–67. ACM Press, New York (1991)
7. Maloof, M.A., Kochut, K.: Modifying Rete to Reason Temporally. In: *ICTAI*, pp. 472–473 (1993)
8. Teodosiu, D., Pollak, G.: Discarding unused temporal information in a production system. In: *Proc. of the ISMM International Conference on Information and Knowledge Management CIKM 1992*, Baltimore, MD, pp. 177–184 (1992)
9. Bohlen, M.H., Busatto, R., Jensen, C.S.: Point-versus interval-based temporal data models. In: *ICDE*, pp. 192–200 (1998)
10. Yoneki, E., Bacon, J.: Unified semantics for event correlation over time and space in hybrid network environments. In: Meersman, R., Tari, Z. (eds.) *OTM 2005*. LNCS, vol. 3760, pp. 366–384. Springer, Heidelberg (2005)
11. Bry, F., Eckert, M.: Temporal order optimizations of incremental joins for composite event detection. In: *Proceedings of Inaugural Int. Conference on Distributed Event-Based Systems*, Toronto, Canada, June 20–22, 2007. ACM, New York (2007)
12. Galton, A., Augusto, J.: Two approaches to event definition. In: Hameurlain, A., Cicchetti, R., Traummüller, R. (eds.) *DEXA 2002*. LNCS, vol. 2453, pp. 547–556. Springer, Heidelberg (2002)
13. Allen, J.F.: Maintaining knowledge about temporal intervals. *Communications of the ACM* 26, 832–843 (1983)
14. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21(7), 558–565 (1978)
15. White, W.M., Riedewald, M., Gehrke, J., Demers, A.J.: What is "next" in event processing? In: *PODS*, pp. 263–272. ACM, New York (2007)
16. Srivastava, U., Widom, J.: Flexible time management in data stream systems. In: *PODS 2004: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 263–274. ACM Press, New York (2004)
17. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal* 15(2), 121–142 (2006)
18. Ghanem, T.M., Aref, W.G., Elmagarmid, A.K.: Exploiting predicate-window semantics over data streams. *SIGMOD Rec.* 35(1), 3–8 (2006)
19. Ghanem, T.M., Hammad, M.A., Mokbel, M.F., Aref, W.G., Elmagarmid, A.K.: Incremental evaluation of sliding-window queries over data streams. *IEEE Trans. Knowl. Data Eng.* 19(1), 57–72 (2007)
20. JBoss: JBoss Rules (2007), <http://labs.jboss.com/drools/>
21. Red Hat, Inc.: Drools development branch for temporal reasoning, <http://anonsvn.labs.jboss.com/labs/jbossrules/branches/temporal-rete/>

Classifying Evolving Data Streams Using Dynamic Streaming Random Forests^{*}

H. Abdulsalam, D.B. Skillicorn, and P. Martin

School of Computing, Queen's University
Kingston, ON Canada, K7L 3N6
{hanady, skill, martin}@cs.queensu.ca

Abstract. We consider the problem of data-stream classification, introducing a stream-classification algorithm, *Dynamic Streaming Random Forests*, that is able to handle evolving data streams using an entropy-based drift-detection technique. The algorithm automatically adjusts its parameters based on the data seen so far. Experimental results show that the algorithm handles multi-class problems for which the underlying class boundaries drift, without losing accuracy.

1 Introduction

Data-stream based applications are widely exploited by modern organizations. Such applications are required to analyze (or mine) streams of unlimited data records, by observing each data record only once, or possibly few times. Mining data streams require incremental, online, and fast algorithms which extract important information from data records on the fly, and can produce online results. Stream-mining algorithms should also adapt to changes in the distribution of the data, and be able to approximate results when necessary.

This paper addresses the data-stream classification problem. The field of stream classification has received much attention [1–5], but some problems exist:

- Algorithms typically require a large training time as a result of using huge amounts of data to build the classification model.
- Algorithms lack the ability to handle multi-class classification problems.
- Many algorithms have low classification accuracy and cannot handle *concept drift*.

Unlike standard classification algorithms that have three different sequential phases, (training, testing, and deployment), each associated with its own dataset, stream classification algorithms have only one stream of data containing both labelled and unlabelled data records. Based on the distribution of the labelled records through the input stream, possible scenarios are proposed by Abdulsalam *et al.*[6], and shown in Figure 1.

The Streaming Random Forests algorithm [6] is a stream classification algorithm combining the ideas of the standard Random Forests algorithm by Breiman [7] and streaming decision trees [1, 2]. It demonstrates good classification accuracy when tested. It cannot, however, deal with concept drift.

^{*} The work reported in this paper has been supported by Kuwait University.

The main contribution of this paper is to define the *Dynamic Streaming Random Forests* algorithm, a stream-classification algorithm that extends the Streaming Random Forests algorithm. It is able to handle Scenario 1 and so successfully deals with concept changes. We incorporate ideas proposed by Vorburger and Bernstein [8] to define an entropy-based concept-change detection technique. In addition, the algorithm automatically adjusts its parameters based on the data seen so far. It handles only ordinal and numerical attributes, and it is able to handle multi-class problems. Experimental results show that our algorithm successfully handles concept drift.

The remainder of the paper is organized as follows: Section 2 states some background. Section 3 explains the Streaming Random Forests algorithm. Section 4 introduces the Dynamic Streaming Random Forests algorithm. Section 5 shows experimental results. Section 6 presents related work. Finally, Section 7 draws our conclusions.

2 Background

Streaming decision trees. A streaming decision tree under construction consists of internal nodes, containing an inequality on one of the attributes, *frontier nodes* that have not yet been either split or turned into leaves, and leaf nodes. Upon each record arrival, the record is routed down the tree, based on the attributes values and the inequalities of the internal nodes, until it reaches a frontier node. As each frontier node is considered, the Hoeffding bound [9] is used to decide if this node has accumulated enough records for a robust decision [1].

Using an entropy measure to detect concept drift. Shannon's entropy [10] is a measure of the disorder or randomness associated with a random variable. Finding the entropy for a dataset with known distribution requires using the following equation:

$$H(x) = - \sum_x P(x) \log_2(P(x)),$$

where x is a discrete random variable, and $P(x)$ is the probability mass function of x . For detecting concept changes using entropy, the two-windows paradigm [8, 11] is typically used, where two sliding windows are tracked; one window is the current window and the other is a reference window that represents the latest distribution. If the entropies of the two windows are different then a change has occurred.

The standard Random Forests algorithm. The Random Forests algorithm [7] is a classification ensemble technique developed by Breiman. It grows a number of binary decision trees and the classification for each new record is the plurality of the votes from the trees. It uses the Gini index for selecting split attributes. For a dataset of n records and m attributes, three conditions must be satisfied when growing each tree:

- A subset of n records, chosen from the original dataset at random with replacement must be used as the training set.
- A randomly-chosen $M \ll m$ attributes are evaluated in building each frontier node.
- No pruning is needed. Each tree is grown to the maximum size possible.

3 The Streaming Random Forests Algorithm

Streaming Random Forests [6] is a stream-classification algorithm that builds streaming decision trees with the techniques from Breiman's Random Forests. Its classification accuracies are approximately equal to those of Random Forests.

The Streaming Random Forests algorithm grows binary decision trees each from a different block of data. The trees are grown using the Hoeffding bounds to decide when to stop building on each node [1]. The algorithm selects the attribute to split using the Gini index. It takes two parameters, namely the number of trees to be built and the number of records used to grow each tree (*tree window*).

Each newly arrived record is routed down the tree under construction, until it reaches a frontier node, where the attribute values of the record contribute to compute the Gini indexes. The values of each attribute (numerical or ordinal) are discretized into fixed-length intervals. The boundaries between the intervals are the possible split points.

As a frontier node of the current tree accumulates n_{min} records, where n_{min} is a defined parameter, both the Hoeffding and Gini tests are applied. If the Hoeffding test is satisfied, then the frontier node is transformed into an internal node with an inequality based on the best attribute and split point reported by the Gini index test.

If the number of records that have reached the frontier node exceeds a threshold (*node window*), and the node has not yet been split, the node is transformed into a leaf if the accumulated records are almost all from one class. Otherwise, the node is transformed into an internal node based on the best attribute and split point so far.

By the end of the *tree window*, a node might have been split and generated two frontier nodes that might have not seen enough data to get transformed into an internal or a leaf node. A limited form of pruning is therefore required to resolve this situation. A detailed explanation of the tree-building procedure and the tree-pruning method can be found in the Streaming Random Forests paper[6].

4 The Dynamic Streaming Random Forests Algorithm

The Dynamic Streaming Random Forests algorithm is a self-adjusting stream classification algorithm that is able to reflect concept changes. The algorithm, like the basic Streaming Random Forests algorithm, initially grows a defined number of trees. The difference is that the *tree window* is not constant for all trees. Instead, whenever a number of $tree_{min}$ records have contributed to building a tree, the current classification error of the tree is calculated. If the error is greater than a threshold (*tree threshold*), then the algorithm stops building this tree and switches to building the next tree. Otherwise, the algorithm continues building the current tree using half the previous number of records, before it enters another test phase. Each tree has, therefore, a different *tree window* that is never greater than $2 * tree_{min}$. The parameters $tree_{min}$ and $tree_{threshold}$, are initially assigned defined values. A typical value for $tree_{threshold}$ is $1/C$, where C is the number of classes. This threshold ensures that none of the trees performs worse than a random tree.

Once the total number of trees have been grown, the algorithm enters a test phase where the classification accuracy for the forest is calculated using previously unseen

labelled data records. The classification error ($tree_{error_i}$), for each individual tree i , is also calculated and used to assign a weight to the tree.

In addition, the algorithm derives new values of the parameters to use in the subsequent building phase at time $t + 1$ (i.e. when a new block of labelled records arrives):

- A new *tree threshold* is calculated as the average error of trees at time t :

$$tree_{threshold}(t + 1) = \frac{1}{U} \sum_{i=1}^U tree_{error_i}(t)$$

where U is the number of trees in the forest.

- A new n_{min} is calculated using the equation:

$$n_{min}(t + 1) = \frac{\ln(1/\delta)}{2(\overline{\Delta Gini}(t))^2}$$

where $\Delta Gini(t) = Gini_{highest} - Gini_{second\_highest}$ is computed during the building phase at time t , and $\overline{\Delta Gini}(t)$ is the average value of $\Delta Gini(t)$.

- A new $tree_{min}$ is calculated using the equation:

$$tree_{min}(t + 1) = n_{min}(t + 1) * \overline{tree_{size}(t)}$$

where $\overline{tree_{size}(t)}$ is the average of all tree sizes from the building phase at time t .

Whenever a new block of labelled data records arrives, the algorithm replaces $w\%$ of the total number of trees with new trees, grown from newly arrived data records. The trees with the largest $tree_{error}$ are replaced. The value of w is obtained empirically. We choose it to be 25%. We believe that this percentage is enough to keep the learned concept up-to-date while not forcing too much time for each building phase. New trees are grown using the derived parameters. If there is a concept change in the data, then the values of these parameters are reset to their initial values since their most recent values were derived based on data records for the old concept. More trees are replaced in the case of concept changes to reflect the new distribution.

Concept drift can be thought of in different ways. There are three categories:

- *Category 1*: changes that are generated from variation in noise.
- *Category 2*: changes that are generated from a gradual change in the underlying rules defining the classes.
- *Category 3*: changes that are generated by a sudden change in the underlying rules.

We use an entropy measure to detect concept changes in streams based on the two-windows paradigm [8, 11]. We base our idea on the work done by Vorburger and Bernstein [8]. Since the distribution of the data is not known, the probability mass function cannot be used directly. Instead, we use counters to find the probabilities of occurrences of values associated with a specific interval and specific class within each attribute for the current and reference windows. The difference in entropies of the two windows for each attribute is calculated as:

$$H_i = \left| \sum_{c=1}^C \sum_{k=1}^K [u_{kc_{cur}} \log_2 u_{kc_{cur}} - u_{kc_{ref}} \log_2 u_{kc_{ref}}] \right|$$

where C is the number of classes, K is the number of intervals for attribute i , and $u_{kc_{cur}}$ and $u_{kc_{ref}}$ are the ratios of the number of records within an interval k and assigned to

class c , to the total number of records of the current and reference windows, respectively. The absolute value is taken for H_i because the magnitude of the change is what really matters.

For a data stream with m attributes, the final entropy difference between the current and reference windows is calculated as the average of H_i , $0 \leq i \leq m$:

$$H = \frac{1}{m} \sum_{i=1}^m H_i.$$

The possible values of H are in the range $[0, |\log_2 1/C|]$. H is zero when entropies of both windows are equal, and hence, there is no drift, while H is maximized and equal to $|\log_2 1/C|$ when a change appears.

The entropy H is normalized by dividing it by its maximum possible value: $H' = H/|\log_2 1/C|$. The value of H' is hence in the range $[0, 1]$ and represents the percentage of the concept change. The algorithm records a change in distribution if $H' > \gamma + H_{AVG}$, where H_{AVG} is the accumulated average entropy of the entropies computed since the last recorded change, and γ is a constant threshold defined empirically. If a change is detected, more trees must be replaced.

We use H' to find the percentage of the number of trees to replace (R):

$$R = \begin{cases} H' * U & \text{if } (H' * U + \frac{1}{C}((1 - H')U) > U/2), \\ H' * U + U/2 & \text{otherwise} \end{cases}$$

This equation considers H' as the percentage of the trees to replace only if the remaining set of unchanged trees, which is calculated as $(1 - H')U$, has a higher probability of contributing to making the correct classification decision when combined with the replaced set of trees. We approximate the fraction of the number of unchanged trees that might positively share in the classification decision as $1/C$, since a tree with uniformly random guessing still has a probability of $1/C$ of getting the correct classification. If the number of trees to replace plus the number of remaining trees that are expected to give correct classification is less than half of the total number of trees, then the number of trees to replace is calculated as $H' * U + U/2$. The equation, therefore, forces the algorithm to replace at least half of the trees when a change occurs.

5 Experimental Settings and Results

We base the implementation of our algorithm on the Streaming Random Forests [6], implemented using Fortran 77. The machine on which we conduct our experiments uses a Pentium 4 3.2 GHz processor and 512MB RAM. We test our algorithm on synthetic datasets generated using the DataGen tool by Melli [12].

We generate a number of datasets and combine them into one dataset having concept changes of the three categories. The combined dataset has 5 attributes and 5 classes. The size of the dataset is 7 million records, with concepts changes described in Figure 2.

For all our experiments, the algorithm grows 50 trees using $M = 3$ attributes for each node. The initial values of *tree threshold*, n_{min} , and *tree_{min}* are $1/C = 0.2$, 200, and 3000 respectively. The sizes of the current and reference windows are 1000 records. The value of γ we use is 0.05, and w is set to 25%.

We evaluate our algorithm based on the following criteria:

1. Entropy-based concept drift detection technique. Figure 3 shows the values of H' , H_{AVG} , and $\gamma + H_{AVG}$ with respect to the number of records of the stream. The algorithm records seven concept changes at points 501000, 1506000, 2049000, 5011000, 5533000, 6032000, and 6501000. Note that a window of 1,000 records reflecting the new concept needs to be seen before detecting a change. Actual change points therefore, appear earlier than the detected changes. The first recorded change represents a drift of category 1. The entropy value of the second simulated drift, where the noise drops to 1% at point 1,000,000 does not show any significant change. This drift is therefore not recorded. The subsequent five recorded changes represent the points during the gradually drifting concept (category 2 drift), where the noise is unchanged and equal to 1%. Some of the boundaries are, however, not recorded.

The last recorded change is the category 3 drift. All the recorded changes have similar entropy values in the range of $[0.07, 0.1]$ except for the last change, which has a value of 0.27. This is because both the noise and learning rules change.

2. Classification accuracy. Figure 4 shows the classification error, versus the number of processed records. The recorded points of the graph corresponds to each time the algorithm has finished building/modifying the forest.

The results show that the algorithm successfully reacts to concept changes of categories 1 and 3. This can be seen at the first two drift points of the data (category 1), and the last point (category 3). Although the entropy change-detection technique did not detect a change for the second change point (at record 1,000,000), the algorithm still performs well, giving an error that is approximately equal to the data noise.

The experiments record poor classification during the period of category 2 drift, shown at the point where the classification error is about 7.5% and the subsequent few points. The justification for this is that, since the data from new learning rules are added into the old dataset gradually, with the first block of mixed data having only 10% of the new records, the model does not see enough records to learn the new concept. Instead, it considers the data generated from new learning rules as an increase in the noise presented in the data. The classification error for this block of data should therefore, be around 10%. Our algorithm performs better, and records a classification error of 7.5%, which drops as the number of records from the new concept increases until it reaches classification errors in the range of $[1.65\%, 2.55\%]$.

3. Dynamic adjustment of parameters. We illustrate the dynamic nature of the algorithm by testing the variation of the parameter values during the execution of the algorithm. We consider *tree threshold*, n_{min} , *tree_{min}*, and the number of trees to replace. Figures 5, 6, 7, and 8 represent the values of the four parameters respectively.

The figures show that, when a concept change is detected, the values of the parameters are reset to their initial values. As shown in Figures 5, 6, and 7, the values of *tree threshold*, n_{min} , and *tree_{min}* decrease when the model is better representing the data records for the following reasons: 1) The stronger the trees, the higher their classification accuracy, and therefore, the smaller the value of *tree threshold*. 2) Small values of n_{min} and *tree_{min}* mean that the algorithm is performing well and does not require a large number of data records in order to update its model.

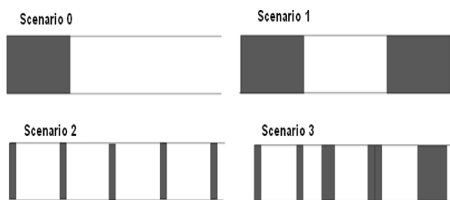


Fig. 1. Possible scenarios for data streams

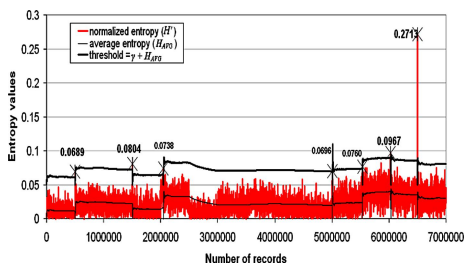


Fig. 3. Entropy versus number of records

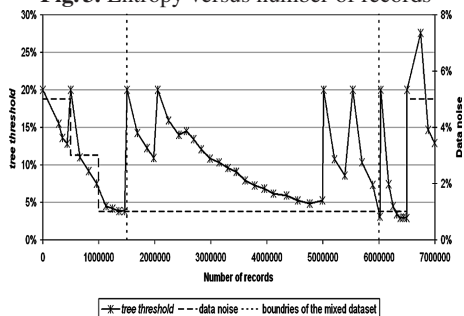
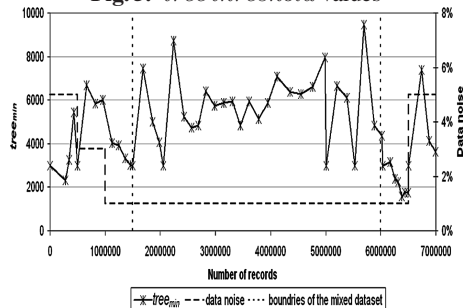
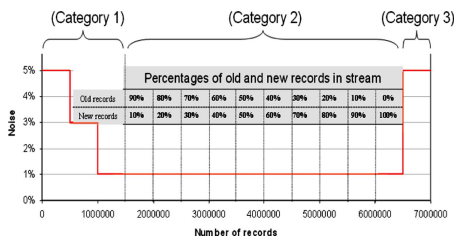

 Fig. 5. $tree\ threshold$ values

 Fig. 7. $tree_{min}$ values


Fig. 2. Concept drift of the synthetic data

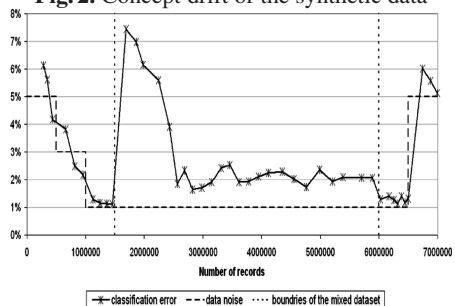


Fig. 4. Classification error of the forest

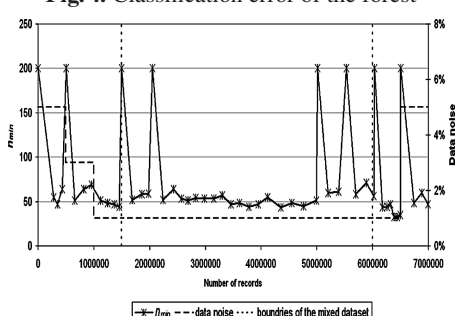
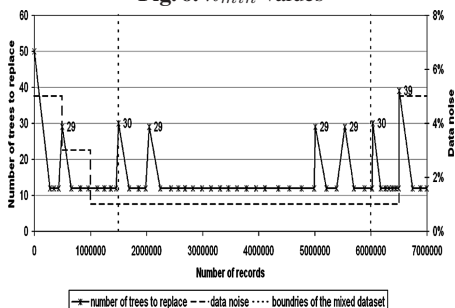

 Fig. 6. n_{min} values


Fig. 8. Number of trees to replace

Figure 8 shows that 50 trees are grown initially, then 25% of the trees (12 trees) are normally replaced, unless a concept drift is detected, where the number of replaced trees increases based on the change significance. For the first six drift points, the number of trees to be replaced varies between 29 and 30 trees, which shows that the significance of the changes are almost equal. The number replaced trees for the last drift is 39. This is because the value of the entropy difference for this drift is the highest.

6 Related Work

Decision trees based on Hoeffding bounds have been widely used for stream classification [1, 2, 5, 13]. Trees ensembles have also been used for stream classification [3–5].

Many of these algorithms are designed for only two class problems. Single decision tree algorithms typically require either significant processing time or memory to handle concept changes [2, 4]. CVFDT [2] handles concept changes by growing an alternative tree for each node. A node is replaced with its alternative tree whenever it records poor classification accuracy. Fan proposed an algorithm that depends on learning four models from combining old and new data [4]. The best model that represents the current data is selected for classification.

Ensemble classifiers have better utilization of data and are more accurate than single classifiers. Many of them are, however, tested only on two-class problems.

7 Conclusion

This paper has presented the *Dynamic Streaming Random Forests* algorithm, a self-adjusting stream-classification algorithm that handles evolving streams using an entropy-based change-detection technique. The algorithm extends the Streaming Random Forests algorithm [6]. It gives the expected behavior when tested on synthetic data with concept drift; the concept drift points were detected; the parameters were automatically adjusted, and the classification error was approximately equal to the noise in the data.

References

1. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2000)
2. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data mining, pp. 97–106 (2001)
3. Chu, F., Wang, Y., Zaniolo, C.: An adaptive learning approach for noisy data streams. In: Proceedings of the IEEE International Conference on Data Mining, pp. 351–354 (2004)
4. Fan, W.: A systematic data selection to mine concept-drifting data streams. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining, pp. 128–137 (2004)
5. Zhu, X., Wu, X., Yang, Y.: Dynamic classifier selection for effective mining from noisy data streams. In: Proceedings of the IEEE International Conference on Data Mining, pp. 305–312 (2004)

6. Abdulsalam, H., Skillicorn, D.B., Martin, P.: Streaming random forests. In: Proceedings of the International Database Engineering and Applications Symposium, pp. 225–232 (2007)
7. Breiman, L.: Random forests. Technical Report (1999), www.stat.berkeley.edu
8. Vorburger, P., Bernstein, A.: Entropy-based concept shift detection. In: Proceedings of the International Conference on Data Mining, pp. 1113–1118 (2006)
9. Hoeffding, W.: Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association* 58(1), 13–30 (1963)
10. Shannon, C.E.: A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* 5(1), 3–55 (2001)
11. Dasu, T., Krishnan, S., Venkatasubramanian, S., Yi, K.: An information-theoretic approach to detecting changes in multi-dimensional data streams. Technical Report (2005)
12. Melli, G.: Scds-a synthetic classification data set generator. Simon Fraser University, School of Computer Science (1997)
13. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining, pp. 523–528 (2003)

On a Parameterized Antidivision Operator for Database Flexible Querying

Patrick Bosc and Olivier Pivert

IRISA-ENSSAT, Technopole Anticipa, BP 80518
22305 Lannion Cedex, France
{Patrick.Bosc,Olivier.Pivert}@enssat.fr

Abstract. In this paper, we introduce an algebraic relational operator called antidivision and we describe a range of interpretations that can be attached to this operator in the context of databases with fuzzy relations (i.e., relations that contain weighted tuples).

Keywords: Relational databases, Antidivision, Fuzzy relations.

1 Introduction

The idea of extending usual Boolean queries with preferences has become a hot topic in the database community. One of the advantages of this approach is to deliver discriminated answers rather than flat sets of elements. Fuzzy sets are a natural means to represent preferences and many works have been undertaken to define queries where fuzzy predicates can be introduced inside user queries. The objective of this article is to illustrate the expressiveness of fuzzy sets with a certain type of queries, that we call *antidivision queries*, in the context of regular databases. Like other operators, the regular antidivision is not flexible at all and small variations in the data may lead to totally different results. Using fuzzy relations counter to a certain extent this behavior by taking into account the notion of graded membership (of a tuple to a relation). First, let us make clear what we mean by antidivision. Let r be a relation of schema $R(X, A)$ and s a relation of schema $S(B, Y)$, with A and B compatible (sets of) attributes. We call antidivision the operator $\nmid$ defined the following way:

$$r[A \nmid B]s = \{x \mid \forall b \in s[B], (x, b) \notin r\}.$$

In other words, an antidivision query $r[A \nmid B]s$ retrieves the X -values present in relation r which are associated in r with *none* of the B -values present in s . In the following, in order to simplify the formulas and with no loss of generality, we will assume that the schema of s is $S(B)$. Some examples of (non-fuzzy) antidivision queries are given hereafter:

- a consumers' association aims at assessing some chemical products (e.g. cosmetics) in order to give them quality labels so as to express their level of safety. In this context, an antidivision query could be: retrieve the products which do not contain any noxious component in a proportion higher than 5%.

- the Atomic Energy Research Center aims at finding a site for implanting a nuclear waste processing plant. In this context, an antidivision query could be: retrieve the sites which are at least hundred miles away from any geographic point where the seismic risk is higher than 2 (on a given scale).

It can be noticed that an antidivision is nothing but an antijoin (denoted hereafter by $\bowtie$) followed by a projection over X:

$$r[A \bowtie B]s = (r \bowtie s)[X].$$

We call this operator antidivision by analogy with the relation between a semijoin and an antijoin: a division query $r[A \div B]s$ (resp. a semi-join query $r \ltimes s$) retrieves the X-values which are associated in r with *all* of the B-values present in s (resp. the tuples from r which join with *at least one* tuple from s) while an antidivision query $r[A \bowtie B]s$ (resp. an antijoin query $r \bowtie s$) retrieves the X-values associated with *none* of the B-values from s (resp. the tuples from r which join with *none* of the tuples from s).

Our purpose is of course not to introduce a superfluous algebraic operator but to show that the concept of antidivision seen as an atomic operator allows to reach a wide range of useful semantics when one moves from regular relations to fuzzy ones.

The remainder of the paper is organized as follows. In section 2, we deal with the possible formulations of the antidivision operator in a regular database context in both relational algebra and SQL. Section 3 is devoted to the antidivision in the context of databases involving fuzzy relations (i.e., relations which contain weighted tuples). We first give some basic notions concerning fuzzy relations and fuzzy queries, then we point out the different semantics that can be attached to the antidivision operator in such a context. The conclusion recalls the main contributions and mentions some perspectives for future work.

2 Antidivision of Regular Relations

In the framework of the relational algebra, an antidivision can be expressed using a difference ($-$), a join ($\bowtie$) and two projections as:

$$r[X] - (r \bowtie s)[X] \quad (1)$$

In SQL, a possible formulation is:

select X from r where X not in (select X from r, s where r.A = s.B).

Example 1. Let us consider the relations *prod* which describes the composition of some chemical products and *nox* which gathers the identifications of noxious components. Let us consider the query “retrieve the products which do not contain any noxious component in a proportion higher than 5%”. Let us suppose that *nox* = { c_1, c_2, c_5 } and *prod* is:

prod	p	c	prop
	p ₁	c ₁	3
	p ₁	c ₂	4
	p ₁	c ₃	93
	p ₂	c ₁	9
	p ₂	c ₄	91
	p ₃	c ₂	8
	p ₃	c ₆	92

This query can be expressed as: $((\text{Prod} : \text{prop} > 5)[p, c])[c \not\parallel c] \text{Nox}$
and its result is $\{p_1\} \blacklozenge$

Another vision of the antidiagnosis in an SQL-like language can be based on an inclusion:

select X from r group by X having set(A) includes_none (select A from s)

where the Boolean predicate includes_none is defined as:

$\text{includes\_none}(E, F) \equiv (E \cap F) = \emptyset \equiv (E \subseteq \text{cp}(F)) \equiv (F \subseteq \text{cp}(E))$.

where $\text{cp}(E)$ denotes the complement of set E.

This vision corresponds to the following definition of the antidiagnosis:

$$r[A \not\parallel B]s = \{x \mid s \cap \Omega(x) = \emptyset\} = \{x \mid s \subseteq \text{cp}(\Omega(x))\} = \{x \mid \Omega(x) \subseteq \text{cp}(s)\} \quad (2)$$

where $\Omega(x) = (r : X = x)[A]$ i.e., is the set of A-values associated with x in r.

3 Antidiagnosis of Fuzzy Relations

In this section, we first recall some basic notions related to fuzzy relations and fuzzy queries, before defining the antidiagnosis operator in a context of fuzzy relations.

3.1 About Fuzzy Relations and Fuzzy Queries

Let us recall that fuzzy set theory [11] aims at representing sets whose boundaries are not sharp. A fuzzy set F defined on a domain X is associated with a membership function μ_F from X into the unit interval [0, 1]. The closer to 1 the membership degree $\mu_F(x)$, the more x belongs to F. The support $S(F)$ and the core $C(F)$ of a fuzzy set F are defined respectively as the following two crisp sets:

$$\begin{aligned} S(F) &= \{x \in X \mid \mu_F(x) > 0\} \\ C(F) &= \{x \in X \mid \mu_F(x) = 1\} \end{aligned}$$

In the database domain, fuzzy set theory can serve as a basis for defining a flexible querying approach [5]. The key concept is that of a fuzzy relation, i.e., a relation designed as a fuzzy subset of Cartesian products of domains. Thus, any such fuzzy relation r can be seen as made of weighted tuples, denoted by μ/t , where μ expresses

the extent to which tuple t belongs to the relation, i.e., is compatible with the concept conveyed by r . Of course, since regular databases are assumed to be queried, initial relations (i.e., those stored in the database) are special cases of fuzzy relations where all the tuple weights are equal to 1.

Example 2. Let us consider a database with the relation employee(num, name, salary, age, living-city). From a given initial extension of this regular relation, it is possible to get the intermediate fuzzy relation fy-emp shown in Table 1 containing those employees who are “fairly young”. It is assumed that the membership function associated with the flexible predicate “fairly young” is defined as follows: $\mu_{fy}(x) = 0$ if $age \geq 45$, $\mu_{fy}(x) = 1$ if $age \leq 30$, linear in between. It can be noticed that no element is a full member of the fuzzy relation fy-emp since no employee reaches the maximal degree 1. In the fuzzy relation obtained, only the tuples t such that $\mu_{fy}(t) > 0$ appear. ♦

Table 1. The extension of the relation fy-emp

num	name	salary	age	living-city	degree
76	martin	12500	40	New-York	0.3
26	tanaka	12000	37	Chiba	0.4
12	smith	12000	39	London	0.4
55	lucas	13000	35	Miami	0.8

The regular relational operations can be extended to fuzzy relations by considering fuzzy relations as fuzzy sets on the one hand and by introducing gradual predicates in the appropriate operations (selections and joins especially) on the other hand. A definition of the division of fuzzy relation can be found in [4]. For more details about query language aspects, the reader may refer to [3] where a fuzzy SQL-like language is described.

3.2 Principle and Formulation of the Antidivision of Fuzzy Relations

Starting from expression (1), and denoting by res the relation resulting from the antidivision query, one gets, for an element x present in relation r , the degree:

$$\begin{aligned}
 \mu_{res}(x) &= \min(\mu_{\text{support}(r[X])}(x), 1 - \mu_{\text{proj}(r \bowtie s, X)}(x)) \\
 &= \min(1, 1 - \max_{a \in s} \top(\mu_s(a), \mu_r(x, a))) \\
 &= \min_{a \in s} (1 - \top(\mu_s(a), \mu_r(x, a)))
 \end{aligned} \tag{3}$$

where $\top$ denotes a triangular norm generalizing the conjunction, e.g., min or product. As to the expression based on an inclusion, it becomes:

$$r[A \not\bowtie B]s = \{\mu/x \mid x \in \text{support}(r[X]) \text{ and } \mu = \text{Inc}(s, \text{cp}(\Omega(x)) > 0\} \tag{4}$$

where $\text{Inc}(s, \text{cp}(\Omega(x)))$ denotes the degree of inclusion ($\in [0, 1]$) of s in $\text{cp}(\Omega(x))$. The graded inclusion indicator Inc can be defined the following way [1]:

$$\text{Inc}(E, F) = \min_{x \in X} (\mu_E(x) \rightarrow \mu_F(x)) \quad (5)$$

where $\rightarrow$ denotes a *fuzzy implication operator*, i.e., a mapping from $[0, 1]^2$ into $[0, 1]$. There are several families of fuzzy implications, notably R-implications [7]:

$$p \rightarrow_R q = \sup_{u \in [0, 1]} \{u \mid \top(u, p) \leq q\}$$

It is possible to rewrite these implications as:

$$p \rightarrow_R q = 1 \text{ if } p \leq q, f(p, q) \text{ otherwise}$$

where $f(p, q)$ expresses a degree of satisfaction of the implication when the antecedent (p) exceeds the conclusion (q). The implications of Gödel ($p \rightarrow_{G\ddot{o}} q = 1$ if $p \leq q$, q otherwise), Goguen ($p \rightarrow_{Gg} q = 1$ if $p \leq q$, q/p otherwise) and Lukasiewicz ($p \rightarrow_{Lu} q = 1$ if $p \leq q$, $1 - p + q$ otherwise) are the three most used R-implications and they are obtained resp. with the norms $\top(x, y) = \min(x, y)$, $\top(x, y) = xy$ and $\top(x, y) = \max(x + y - 1, 0)$.

As to S-implications [7], they generalize the (usual) material implication $p \Rightarrow q = ((\text{not } p) \text{ or } q)$ by:

$$p \rightarrow_s q = \perp(1 - p, q)$$

The minimal element of this class, namely Kleene-Dienes' implication obtained with $\perp = \max$ expresses the inclusion of the support of E in the core of F (1 is reached then). This is also the case for Reichenbach's implication (obtained with the norm product).

Let us discuss the impact of the type of implication (R- or S-) on the semantics of the antidivision. The degrees in the divisor (relation s) act as:

- importance levels if Kleene-Dienes' implication is used; in this case, the higher the degree attached to an element a of s , the more the degree attached to $\langle a, x \rangle$ in r impacts the final degree attached to x in the result; the complement of the degree attached to a (i.e., $1 - \mu_s(a)$) corresponds to a guaranteed satisfaction level;
- thresholds with any R-implication; here, the higher the degree attached to an element a of s , the smaller the degree attached $\langle a, x \rangle$ in r must be so as to get a final degree attached to x equal to 1; when the degree attached to $\langle a, x \rangle$ in r is higher than 1 minus the degree attached to a in s , a penalty is applied, which varies with the R-implication considered.

Example 3. Let us come back to the context of example 1 and consider relations prod and nox again. This time, these relations are supposed to be fuzzy in order to express that a component can be more or less noxious and that the proportion of a component in a chemical product can be more or less important. Let us consider the query "retrieve the products which do not contain any highly noxious component in a significant proportion".

prod				nox		
	p	c	μ		c	μ
	p ₁	c ₁	0.3		c ₁	0.8
	p ₁	c ₂	0.85		c ₂	0.3
	p ₁	c ₃	1		c ₄	0.1
	p ₂	c ₁	1		c ₅	0.6
	p ₂	c ₄	0.7		c ₆	0.4
	p ₃	c ₂	1			
	p ₃	c ₆	0.9			

The fuzzy term “significant” can be defined for instance as $\mu_{\text{sig}}(x) = 0$ if $x \leq 3$, $\mu_{\text{sig}}(x) = 1$ if $x \geq 7$, linear in-between. This fuzzy term is used to obtain the relation prod above by means of a fuzzy selection applied on a regular relation of schema (p, c, proportion) such as that from Example 1. The degrees in relation nox are supposed to be specified explicitly (the divisor relation can even be given in extension in the query). Let us consider the extensions of prod and nox given above. The antidivision query can be expressed as:

(Prod [p, c]) [c $\nVdash$ c] Nox

With Gödel’s implication, one gets the result: $\{0.15/p_1\}$ since:

$$\mu(p_1) = \min(0.7, 0.15, 1) = 0.15, \mu(p_2) = \min(0, 1) = 0, \mu(p_3) = \min(0, 0.1) = 0.$$

With Goguen’s implication, one gets $\{0.5/p_1\}$ since:

$$\mu(p_1) = \min(7/8, 0.5, 1) = 0.5, \mu(p_2) = \min(0, 1) = 0, \mu(p_3) = \min(0, 0.25) = 0.$$

With Lukasiewicz’ implication, one gets $\{0.85/p_1, 0.2/p_2, 0.7/p_3\}$ since:

$$\mu(p_1) = \min(0.9, 0.85, 1) = 0.85, \mu(p_2) = \min(0.2, 1) = 0.2, \mu(p_3) = \min(0.7, 0.7) = 0.7.$$

With Kleene-Dienes’ implication, one gets $\{0.7/p_1, 0.2/p_2, 0.6/p_3\}$ since:

$$\mu(p_1) = \min(0.7, 0.7, 1) = 0.7, \mu(p_2) = \min(0.2, 0.9) = 0.2, \mu(p_3) = \min(0.7, 0.6) = 0.6.$$

◆

Now, let us give a semantic justification of expression (4). It is important to notice that if an R-implication is used in (5), one loses the equivalence valid in the Boolean case between $\text{Inc}(s, \text{cp}(\Omega(x)))$ and $\text{Inc}(\Omega(x), \text{cp}(s))$. Indeed, with an R-implication, the truth value of $(p \rightarrow_R q)$ is not equal to $[(\text{not } q) \rightarrow_R (\text{not } p)]$ in general. Now:

$$\text{Inc}(s, \text{cp}(\Omega(x))) = \min_{a \in s} (\mu_s(a) \rightarrow_R 1 - \mu_{\Omega(x)}(x, a))$$

$$\text{Inc}(\Omega(x), \text{cp}(s)) = \min_{a \in \Omega(x)} (\mu_{\Omega(x)}(a) \rightarrow_R 1 - \mu_s(a))$$

On the other hand, the equivalence between $\text{Inc}(s, \Omega(x))$ and $\text{Inc}(\Omega(x), \text{cp}(s))$ is preserved by S-implications.

In the case of an R-implication, the “correct” choice for defining the antidivision is thus to use $\text{Inc}(s, \text{cp}(\Omega(x)))$ – as in expression (4) – and not $\text{Inc}(\Omega(x), \text{cp}(s))$. Indeed, the expected behavior is that the degrees attached to the elements of the divisor act as thresholds, and not the opposite. Finally, we have :

$$\mu_{r[A \nparallel B]s}(x) = \text{Inc}(s, \Omega(x)) = \min_{a \in s} (\mu_s(a) \rightarrow 1 - \mu_r(x, a)) \quad (6)$$

Dubois and Prade [6] have shown that R-implications and S-implications can be expressed using a common format, i.e.:

$$p \rightarrow q = 1 - \text{cnj}(p, 1 - q)$$

where cnj denotes a triangular norm $\top$ when the implication is an R-implication, and a non-commutative conjunction ncc when it is an S-implication. For example, the operators ncc associated with Gödel's and Goguen's implications are respectively:

$$\text{ncc}_{\text{Gö}}(x, y) = 0 \text{ if } x + y \leq 1, y \text{ otherwise}$$

$$\text{ncc}_{\text{Gg}}(x, y) = 0 \text{ if } x + y \leq 1, (x + y - 1)/x \text{ otherwise.}$$

Hence, we get the generic expression for the antideviation of fuzzy relations:

$$\mu_{r[A \nparallel B]s}(x) = \min_{a \in s} (1 - \text{cnj}(\mu_s(a), \mu_r(x, a))) \quad (7)$$

which generalizes (3) by taking into account non-commutative conjunctions.

Remark. Expression (4) – whose interpretation rests on formula (7) – also generalizes the definition of the antideviation based on an intersection, i.e.:

$$R[A \nparallel B] S = \{x \mid s \cap \Omega(x) = \emptyset\} \quad (8)$$

Indeed, from this expression, it comes:

$$\begin{aligned} \mu_{r[A \nparallel B]s}(x) &= 1 - \mu_{\cap}(s, \Omega(x)) \\ &= 1 - \max_{a \in s} \top(\mu_s(a), 1 - \mu_r(x, a)) \\ &= \min_{a \in s} (1 - \top(\mu_s(a), \mu_r(x, a))) \end{aligned}$$

which is nothing but formula (3).

In order to obtain the generic semantics that we propose for the antideviation in relational algebra, it would be necessary to parameterize the Cartesian product by the conjunction operator. On the one hand, this would not be very easy to do for an end-user (it is not obvious how to choose the right conjunction operator to get the desired threshold-based or importance-based behavior) and, on the other hand, this raises a semantic difficulty since the Cartesian product is by nature a symmetrical operator (but it would not stay so if it were based on a non-commutative conjunction).

In an SQL-like language, the most simple solution is to parameterize the operator *includes_none* introduced above by the fuzzy implication desired. We get an expression of the form:

```
select x from r
group by x
having set(A) includes_none fuzzy implication (select B from s)
```

where $\text{includes\_none}_{\text{fuzzy implication}}(E, F) = \text{Inc}_{\text{fuzzy\_implication}}(F, \text{cp}(E))$.

4 Conclusion

In this paper, we have introduced the concept of an antidivision operator, which, in the classical relational framework, corresponds to a non-primitive operator since it can be expressed by means of an antijoin, a projection and a difference. Seeing this operator as an atomic operator becomes particularly interesting when one moves to the framework of fuzzy relations. We have provided a generic definition for the antidivision operator, based on a graded inclusion, which captures a wide range of semantics.

Among the perspectives for future work, it would be worthy dealing with the optimization of antidivision queries both in the regular relational database model and the fuzzy extension of this model. In particular, it would be interesting to study whether some optimization mechanisms proposed for antijoin queries, such as those described in [10], could be adapted to the processing of antidivision queries.

Another extension of this work concerns the application of the antidivision operator proposed here to the context of information retrieval. In different information retrieval models, it is indeed possible to specify inside a user query a set of (possibly weighted) unwanted keywords [8, 9]. We thus believe that the antidivision operator would be a well suited tool for interpreting the “negative part” of a query, in the same way that a fuzzy division operator can be used to interpret its “positive part” (i.e., the set of required keywords), as described in [2].

References

1. Bandler, W., Kohout, L.: Fuzzy power sets and fuzzy implication operators. *Fuzzy Sets and Systems* 4, 13–30 (1980)
2. Bordogna, G., Bosc, P., Pasi, G.: Fuzzy inclusion in database and information retrieval query interpretation. In: *Proc. ACM SAC 1996*, pp. 547–551 (1996)
3. Bosc, P., Pivert, O.: SQLf: a relational database language for fuzzy querying. *IEEE Transactions on Fuzzy Systems* 3, 1–17 (1995)
4. Bosc, P., Pivert, O., Rocacher, D.: About quotient and division of crisp and fuzzy relations. *Journal of Intelligent Information Systems* 29, 185–210 (2007)
5. Bosc, P., Prade, H.: An Introduction to the Treatment of Flexible Queries and Uncertain or Imprecise Databases. In: Motro, A., Smets, P. (eds.) *Uncertainty Management in Information Systems*, pp. 285–324. Kluwer Academic Publishers, Dordrecht (1997)
6. Dubois, D., Prade, H.: A theorem on implication functions defined from triangular norms. *Stochastica* 8, 267–279 (1984)
7. Fodor, J.: On fuzzy implication operators. *Fuzzy Sets and Systems* 42, 293–300 (1991)
8. Lee, J.H., Kim, W.Y., Kim, M.H., Lee, Y.J.: On the evaluation of Boolean operators in the extended Boolean retrieval framework. In: *Proc. SIGIR 1993*, pp. 291–297 (1993)
9. Pasi, G.: A logical formulation of the Boolean model and of weighted Boolean models. In: *Workshop on Logical and Uncertainty Models for Information Systems (LUMIS 1999)*, pp. 1–11 (1999)
10. Rao, J., Lindsay, B.G., Lohman, G.M., Pirahesh, H., Simmen, D.E.: Using EELs, a practical approach to outerjoin and antijoin reordering. In: *Proc. ICDE 2001*, pp. 585–594 (2001)
11. Zadeh, L.A.: Fuzzy sets. *Inf. Control*, vol. 8, pp. 338–353 (1965)

Providing Explanations for Database Schema Validation

Guillem Rull^{*,**}, Carles Farré^{**}, Ernest Teniente^{**}, and Toni Urpi^{**}

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya
1-3 Jordi Girona, Barcelona 08034, Spain
{grull, farre, teniente, urpi}@lsi.upc.edu

Abstract. We propose a new method for database schema validation that provides an explanation when it determines that a certain desirable property of a database schema does not hold. Explanations are required to give the designer a hint about the changes of the schema that are needed to fix the problem identified. Our method is an extension of the CQC method, which has been shown successful for testing such properties.

1 Introduction

Database schema validation is related to check whether a database schema correctly and adequately describes the user intended needs and requirements. The correctness of the data managed by database management systems is vital to the more general aspect of quality of the data and thus their usage by different applications. A well-known approach to this problem is aimed at checking whether the database schema satisfies desirable properties such as schema satisfiability, query liveness, non-redundancy of constraints, etc.

An important drawback of previous research in this area is that none of the methods proposed to deal with this problem [3,5,9] is able to provide explanations when a certain property does not hold. Therefore, the designer has to consider the full database schema to identify the required schema changes that would fix the problem.

As an example, assume the database schema includes the following two tables:

```
CREATE TABLE Category (  
  name    char(10)  PRIMARY KEY,  
  salary  real      NOT NULL,  
  CONSTRAINT chMinSal CHECK (salary >= 50000),  
  CONSTRAINT chMaxSal CHECK (salary <= 30000) )  
  
CREATE TABLE Employee (  
  ssn     int       PRIMARY KEY,  
  name    char(30)  NOT NULL,  
  catName char(10)  NOT NULL,  
  CONSTRAINT chCatName CHECK (catName <> 'ceo'),  
  CONSTRAINT fkCat FOREIGN KEY (catName) REFERENCES Category(name) )
```

* This work was supported in part by Microsoft Research through the European PhD Scholarship Programme.

** This work was supported in part by the Ministerio de Educación y Ciencia under project TIN2005-05406.

The previous schema may not contain any tuple. The reason is that it is impossible to insert a category since it should have a salary lower than 30000 and higher than 50000. Moreover, since employees must always belong to categories (as stated by the *fkCat* constraint) it is also impossible to insert any employee in the previous database.

Previous methods allow determining that the schema is not satisfiable but they do not give any hint about the reasons that motivate this problem. Taking into account that this could be just a small part of the schema, it may be very hard for the designer to identify the modification that would arrange the problem.

A possible solution may be to use *black-box* techniques [1,6] to compute an explanation, which is understood as a set of constraints responsible for the non-satisfaction of the property. However, these techniques require several executions of the method used to test the property. As schemas become larger, a faster way to perform this computation is needed.

In this paper, we adopt the same definition of explanations as [1,6], but we follow a *glass-box* approach, which is aimed at computing an explanation with a single execution of the method at the same time that it checks whether the tested property holds. We extend the CQC method [4] for this purpose. In the previous example, a single execution of the method we propose in this paper would provide the explanation $\{chMinSal, chMaxSal, fkCat\}$. In addition, the modifications of the CQC method we propose here result also in a substantial efficiency improvement since they reduce the search space required to find the solution to the tested property.

In some cases, the explanation provided by our method may be non-minimal. An explanation is minimal if there is no proper subset of it that is also an explanation. If we were interested in minimal explanations, we could obtain them through a black-box strategy by executing our method as many times as constraints the non-minimal explanation has. Clearly, since the new method is more efficient than the original one and the number of constraints taken into account is never greater than the constraints in the schema (being usually much lower), our approach also improves efficiency of previous black-box techniques [1,6] for obtaining a minimal explanation.

Next section reviews basic concepts. Section 3 describes the CQC_E method, our proposal to draw explanations. Conclusions are given in Section 4.

2 Base Concepts

A *database schema* is a tuple (DR, IC) where DR is a finite set of deductive rules and IC a finite set of constraints. A *deductive rule* has the form

$$p(\bar{X}) \leftarrow r_1(\bar{X}_1) \wedge \dots \wedge r_n(\bar{X}_n) \wedge \neg r_{n+1}(\bar{Y}_1) \wedge \dots \wedge \neg r_m(\bar{Y}_s) \wedge C_1 \wedge \dots \wedge C_t.$$

Symbols p, r_i are *predicates*. Tuples $\bar{X}_i, \bar{Y}_i$ contain *terms*, which are either variables or constants. The terms in tuple $\bar{X}$ are distinct variables. Each C_i is a *built-in literal* in the form of $t_1 \theta t_2$, where t_1 and t_2 are terms and operator θ is $<, \leq, >, \geq, =$ or $\neq$. Atom $p(\bar{X})$ is the *head* of the rule, and $r_i(\bar{X}_i), \neg r_i(\bar{Y}_i)$ are positive and negative *ordinary literals* (those that are not built-in). Rules must be *safe*, that is, every variable that occurs in $\bar{X}, \bar{Y}_i, C_i$ must also appear in some $\bar{X}_j$. Predicates that appear in the head of a rule are *derived predicates*, also called *IDB* predicates. The rest are *base predicates*, also called *EDB* predicates.

We define a derived *inconsistency predicate* Ic_i for each *constraint* in IC . A database instance *violates* a constraint $Ic_i \leftarrow L_1 \wedge \dots \wedge L_k$ if Ic_i is *true*, i.e. if there is some ground substitution σ that makes $(L_1 \wedge \dots \wedge L_k)\sigma$ true.

In this paper, we assume that schema validation properties are expressed in terms of a goal $G = \leftarrow L_1 \wedge \dots \wedge L_m$ and a set of conditions to enforce $F \subseteq IC$ [3]. In this way, we say that a schema validation property is *satisfiable* if there is at least one database instance that makes G true and does not violate any integrity constraint in F . An *explanation* for the non-satisfaction of a schema validation property is a set of integrity constraints $E \subseteq F$ such that (G, E) is not satisfiable.

3 The Approach

The main aim of our approach is to perform satisfiability tests for schema validation properties expressed in the formalism stated above, in such a way that (1) if the property is satisfiable, we provide a concrete database instance in which such a property holds; otherwise, (2) if the property is not satisfiable, we provide an explanation.

Reference [3] showed how to use the CQC method [4] to validate highly expressive database schemas (featuring integrity constraints, negations and comparisons). However, the CQC method does not provide any kind of explanation when a schema validation test fails. In this paper, we propose an extension of the CQC method that provides such an explanation. We refer to this extension as the *CQC_E method*.

Roughly, the original CQC method performs validation tests by trying to construct a database instance for which the tested property holds. The method uses different *Variable Instantiation Patterns (VIPs)* [4], according to the syntactic properties of the database schema considered in each test, to instantiate the ground EDB facts to be added in the database. Adding a new fact may cause the violation of some constraints. When a violation is detected, some previous decisions must be reconsidered to explore alternative ways to reach a solution (e.g., reinstantiate a variable with another constant). In any case, the CQC method does not prescribe any particular execution strategy for the generation of the different alternatives.

The extension we propose in this paper is to define an execution strategy that explores only those alternatives that are indeed relevant for reaching the solution. In order to do this, we need to modify the internal mechanisms of the CQC method to gather the additional information that is required to detect which alternatives are relevant. If none of these alternatives leads to a solution, this same information will be used to build one explanation: the explanation of why this execution has failed.

3.1 Example

Let us consider the example presented in the introduction, expressed here in the logical formalism required by our method. Due to space reasons, we consider neither the primary keys nor the attribute *Employee.name*. However, these modifications do not affect the computed explanation.

$$\begin{aligned} DR &= \{isCat(X) \leftarrow Cat(X, S)\} \\ IC &= \{Ic_1 \leftarrow Emp(X, Y) \wedge Y = 'ceo', Ic_2 \leftarrow Emp(X, Y) \wedge \neg isCat(Y), \\ &\quad Ic_3 \leftarrow Cat(X, S) \wedge S > 30, Ic_4 \leftarrow Cat(X, S) \wedge S < 50\} \end{aligned}$$

Suppose we want to check whether schema validation property ($G \leftarrow \text{Emp}(X, Y), Ic$) is satisfiable, that is, whether the *Employee* table may have at least one tuple in its extension. Fig. 1 shows a CQC_E -derivation that tries to construct an EDB to prove that this property is satisfiable. Each row in the figure corresponds to a CQC_E -node that contains the following information (columns): (1) The goal to attain: the literals that must be made true by the EDB under construction. (2) The conditions to be enforced: the set of conditions that the constructed EDB is required to satisfy. (3) The extensional database (EDB) under construction. (4) The conditions to be maintained: a set containing those conditions that must remain satisfied until the end of the CQC_E -derivation. (5) The set of constants used so far.

The transition between an ancestor CQC_E -node and its successor is performed by applying a CQC_E -expansion rule to a selected literal (underlined in Fig. 1) of the ancestor CQC_E -node (see Section 3.2).

The first two steps shown in Fig. 1 instantiate variables X and Y from literal $\text{Emp}(X, Y)$ in order to obtain a ground fact to be added to the EDB. The constants used to instantiate the variables are determined according to the corresponding Variable Instantiation Patterns (VIPs) [4] and their data type (int, real or string). A label is attached to the constant occurrences, indicating the node where they were introduced. Step 3 inserts the instantiated literal to the EDB under construction. Label 3 is attached to the new tuple to keep record of which node was responsible for its insertion. After this step, we get a node with an empty goal, i.e. []. However, the work is not done yet, since we must ensure that the four constraints are not violated by the current EDB. Steps 4 and 5 evaluate constraint Ic_1 , which is violated.

The analysis of a violation consists in finding those ancestor CQC_E -nodes in the current derivation that take a decision whose reconsideration may help to avoid, *repair*, the violation. Each one of these CQC_E -nodes is a *repair* for the violated constraint. The set of repairs for Ic_1 is recorded in the failed CQC_E -node 5 where constraint Ic_1 was violated. One way to repair this violation is change the value of constant ceo^2 in order to make $ceo^2 = ceo$ false. The label 2 attached to constant ceo indicates that this constant was used in the expansion of CQC_E -node 2 to instantiate a certain variable. Thus, we can backtrack to node 2 and try another instantiation for variable Y . This means node

Goal to attain	Conditions to enforce	EDB	Conditions to maintain	Used constants	Node ID
$\leftarrow \text{Emp}(X, Y)$	$\{Ic_1, Ic_2, Ic_3, Ic_4\} = C_0$	$\emptyset$	C_0	$\{50, 30, ceo\}$	1
1:A2.1 $\leftarrow \text{Emp}(0^1, Y)$	$\{Ic_1, Ic_2, Ic_3, Ic_4\}$	$\emptyset$	C_0	$\{50, 30, ceo, 0\}$	2
2:A2.1 $\leftarrow \text{Emp}(0^1, ceo^2)$	$\{Ic_1, Ic_2, Ic_3, Ic_4\}$	$\emptyset$	C_0	$\{50, 30, ceo, 0\}$	3
3:A2.2 []	$\{Ic_1 \leftarrow \text{Emp}(X, Y) \wedge Y = ceo, Ic_2, Ic_3, Ic_4\}$	$\{\text{Emp}(0^1, ceo^2)^3\}$	C_0	$\{50, 30, ceo, 0\}$	4
4:B2 []	$\{Ic_1 \leftarrow [\text{Emp}(0^1, ceo^2)^3 \wedge] \underline{ceo^2 = ceo}, Ic_2, Ic_3, Ic_4\}$	$\{\text{Emp}(0^1, ceo^2)^3\}$	C_0	$\{50, 30, ceo, 0\}$	5
5:Failed derivation					

Fig. 1. Example of CQC_E -derivation

Goal to attain	Conditions to enforce	EDB	Conditions to maintain	Used constants	Node ID
$\leftarrow \underline{Emp}(0^1, Y)$	$\{Ic_1, Ic_2, Ic_3, Ic_4\}$	$\emptyset$	C_0	$\{50, 30, \text{ceo}, 0\}$	2
6:A2.1					
$\leftarrow \underline{Emp}(0^1, a^2)$	$\{Ic_1, Ic_2, Ic_3, Ic_4\}$	$\emptyset$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	6
7:A2.2					
$\square$	$\{Ic_1 \leftarrow \underline{Emp}(X, Y) \wedge Y = \text{ceo}, Ic_2, Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	7
8:B2					
$\square$	$\{Ic_1 \leftarrow [\underline{Emp}(0^1, a^2)^6 \wedge] a^2 = \text{ceo}, Ic_2, Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	8
9:B5					
$\square$	$\{Ic_2 \leftarrow \underline{Emp}(X, Y) \wedge \neg \text{isCat}(Y), Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	9
10:B2					
$\square$	$\{Ic_2 \leftarrow [\underline{Emp}(0^1, a^2)^6 \wedge] \neg \text{isCat}(a^2), Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	10
11:B3					
$\leftarrow \underline{\text{isCat}}(a^2)^{10}$	$\{Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	11
12:A1					
$\leftarrow \underline{\text{Cat}}(a^2, S)^{11}$	$\{Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	12
13:A2.1					
$\leftarrow \underline{\text{Cat}}(a^2, 50^{12})^{11}$	$\{Ic_3, Ic_4\}$	$\{Emp(0^1, a^2)^6\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	13
14:A2.2					
$\square$	$\{Ic_3 \leftarrow \underline{\text{Cat}}(X, S) \wedge S > 30, Ic_4, Ic_1, Ic_2\}$	$\{Emp(0^1, a^2)^3, \text{Cat}(a^2, 50^{12})^{13}\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	14
15:B2					
$\square$	$\{Ic_3 \leftarrow [\underline{\text{Cat}}(a^2, 50^{12})^{13} \wedge] 50^{12} > 30, Ic_4, Ic_1, Ic_2\}$	$\{Emp(0^1, a^2)^3, \text{Cat}(a^2, 50^{12})^{13}\}$	C_0	$\{50, 30, \text{ceo}, 0, a\}$	15
16:Failed derivation					

Fig. 2. An alternative CQC_E-(sub)derivation

2 is one of the *repairs* for the violation, so node 2 is included in the set of repairs of node 5. Other possible way to repair the violation is avoid the insertion of tuple $Emp(0^1, \text{ceo}^2)^3$ to the EDB. Label 3 indicates that this tuple was inserted in order to satisfy the literal $Emp(0^1, \text{ceo}^2)$ from the goal of node 3. The only possible way to avoid this insertion is by means of avoiding the presence of this literal in the goal. However, as the literal comes from the original goal (note there is no label attached to it), the insertion of the tuple to the EDB cannot be avoided. Therefore, the set of repairs of node 5 is $\{2\}$.

With this information into account, the method will try to construct an alternative CQC_E-(sub)derivation to achieve the initial goal, which will be rooted at CQC_E-node 2 (the repair of node 5). Moreover, in order to keep track of what has happened in the failed derivation, node 2 will record the set of repairs of node 5 together with the *explanation* of why that derivation failed, that is, the set $\{Ic_1\}$.

Fig. 2 shows an alternative CQC_E-derivation rooted at node 2. Steps 6, 7, 8 of this new derivation are similar to steps 2, 3 and 4, but step 6 uses a fresh constant 'a' to instantiate variable Y. Step 9 selects literal $a^2 = \text{ceo}$. Since such a comparison is false, Ic_1 is not violated now, and so, it is removed from the set of conditions to enforce.

Steps 10 and 11 deal with referential constraint Ic_2 , which introduces a new (sub)goal: $\text{isCat}(a^2)$. To achieve it, tuple $\text{Cat}(a^2, 50^{12})^{13}$ is added to the EDB (step 14), but this addition violates constraint Ic_3 (step 16).

As before, the analysis of the violation is performed. In this case, the set of repairs, recorded in node 15, is $\{12, 10\}$. The intuition is that the violation was originated by the instantiation of variable S in node 12, and that this instantiation was required to achieve the (sub)goal introduced by node 10.

The method will try to construct another alternative (sub)derivation rooted at CQC_E -node 12. Any derivation starting from node 12 will fail because each possible instantiation for variable S in $\text{Cat}(a^2, S)$ will lead to the violation of either Ic_3 or Ic_4 , with $\{12, 10\}$ as the set of repairs in any case. Therefore, the method marks CQC_E -node 12 as failed. Its explanation is $\{Ic_3, Ic_4\}$, and the set of repairs is $\{10\}$. The method will visit now this node 10. This node enforces referential constraint Ic_2 , and so, leads to the violation of constraints Ic_3 and Ic_4 . Since there is not an alternative (sub)derivation rooted at node 10, the method marks this node as failed. The explanation for this failure is the explanation of its only (sub)derivation plus the referential constraint Ic_2 , i.e. $\{Ic_2, Ic_3, Ic_4\}$. The set of repairs of node 10 is the empty set. Therefore, there is no point in reconsidering any previous decision, so the method ends without being able of constructing an EDB that satisfies the initial goal, and returns $\{Ic_2, Ic_3, Ic_4\}$ as the set of integrity constraints that explains such a failure (the explanation indicated in the introduction). Note that since node 2 does not belong to the set of repairs of node 10, the explanation for the failed derivation in Fig. 1, recorded at node 2, is discarded and not included in the final explanation.

3.2 Formalization

Let $S = (DR, IC)$ be a database schema, $G_0 = \leftarrow L_1 \wedge \dots \wedge L_n$ a goal, and $F_0 \subseteq IC$ a set of constraints to enforce, where G_0 and F_0 characterize a certain schema validation property in the terms defined in [3]. A CQC_E -node is a 5-tuple of the form $(G_i, F_i, D_i, C_i, K_i)$, where G_i is a goal to attain; F_i is a set of conditions to enforce; D_i is a set of ground EDB atoms, an EDB under construction; C_i is the whole set of conditions that must be maintained; and K_i is the set of constants appearing in DR, G_0, F_0 and D_i .

A CQC_E -tree is inductively defined as follows:

1. The tree consisting of the single CQC_E -node $(G_0, F_0, \emptyset, F_0, K)$ is a CQC_E -tree.
2. Let T be a CQC_E -tree, and $(G_n, F_n, D_n, C_n, K_n)$ a leaf CQC_E -node of T such that $G_n \neq []$ or $F_n \neq \emptyset$. Then the tree obtained from T by appending one or more descendant CQC_E -nodes according to a CQC_E -expansion rule applicable to $(G_n, F_n, D_n, C_n, K_n)$ is again a CQC_E -tree.

It may happen that the application of a CQC_E -expansion rule on a leaf CQC_E -node $(G_n, F_n, D_n, C_n, K_n)$ does not obtain any new descendant CQC_E -node to be appended to the CQC_E -tree because some necessary constraint defined on the CQC_E -expansion rule is not satisfied. In such a case, we say that $(G_n, F_n, D_n, C_n, K_n)$ is a *failed* CQC_E -node. Each branch in a CQC_E -tree is a CQC_E -derivation consisting of a (finite or infinite) sequence $(G_0, F_0, D_0, C_0, K_0), (G_1, F_1, D_1, C_1, K_1), \dots$ of CQC_E -nodes. A CQC_E -derivation is *successful* if it is finite and its last (leaf) CQC_E -node has the form $([], \emptyset, D_n, C_n, K_n)$. A CQC_E -derivation is *failed* if it is finite and its last (leaf) CQC_E -node is failed. A CQC_E -tree is *successful* when at least one of its branches is a successful CQC_E -derivation. A CQC_E -tree is *finitely failed* when each one of its branches is a failed CQC_E -derivation.

```

ExpandNode( $T$ : CQCE-tree,  $N$ : CQCE-node): Boolean
  if  $N$  is a solution node then  $T.solution := N$ ;  $B := true$ 
  else
     $B := false$ 
    Apply one CQCE-expansion rule  $R$ .
    if  $children(N, T) = \emptyset$  then HandleLeaf( $T, N$ )
    else
       $U := children(N, T)$ 
      while  $\exists M \in U \wedge \neg B$ 
        if ExpandNode( $T, M$ ) then  $B := true$ 
        else if  $N \notin M.repairs$  then  $N.repairs := M.repairs$ ;  $N.explanation := M.explanation$ ;  $U := \emptyset$ 
        else
          if  $R$  is A1-rule or A2.1-rule then HandleDecisionalNode( $T, N$ )
          else /* $R$  is B3-rule*/ HandleSelectionOfConstrWithNegs( $T, N$ )
           $U := U - \{M\}$ 
      return  $B$ 

```

```

HandleLeaf( $T$ : CQCE-tree,  $N$ : CQCE-node)
  if  $N.selectedLiteral$  is from  $N.goal$  then
     $N.repairs := RepairsOfGoalComparison(N.selectedLiteral, T)$ ;  $N.explanation := \emptyset$ 
  else /* $N.selectedLiteral$  is from  $N.selectedCondition$ */
     $N.repairs := RepairsOfc(N.selectedCondition, T, N)$ 
    Let us assume  $N.selectedCondition$  defines predicate  $Ic_i$ .
    if there is a constraint  $Ic$  defining predicate  $Ic_i$  in  $root(T).conditionsToEnforce$  then
       $N.explanation := \{Ic\}$ 
    else /* $N.selectedCondition$  appeared as a result of a negative literal in the goal*/
       $N.explanation := \emptyset$ 

```

```

HandleSelectionOfConstrWithNegs( $T$ : CQCE-tree,  $N$ : CQCE-node)
  Let  $children(N, T) = \{M\}$ ; Let us assume  $N.selectedCondition$  defines predicate  $Ic_i$ .
   $N.repairs := M.repairs - \{N\}$ 
  if there is a constraint  $Ic$  defining predicate  $Ic_i$  in  $root(T).conditionsToEnforce$  then
     $N.explanation := M.explanation \cup \{Ic\}$ 
  else
     $N.explanation := M.explanation$ 

```

```

HandleDecisionalNode( $T$ : CQCE-tree,  $N$ : CQCE-node)
   $N.explanation := \emptyset$ ;  $N.repairs := \emptyset$ 
  for each node  $C \in children(N, T)$ 
     $N.explanation := N.explanation \cup C.explanation$ ;  $N.repairs := N.repairs \cup (C.repairs - \{N\})$ 

```

Fig. 3. Formalization of the CQC_E-tree exploration process

Fig. 3 shows the formalization of the CQC_E-tree exploration process. $ExpandNode(T, N)$ is the main algorithm, which generates and explores the subtree of T that is rooted at N . The CQC_E method starts with a call to $ExpandNode(T, N_{root})$ where T contains only the initial node $N_{root} = (G_0, F_0, \emptyset, F_0, K)$. Constants, literals and constraints in N_{root} are unlabeled. If the CQC_E method constructs a successful derivation, $ExpandNode(T, N_{root})$ returns “true” and $T.solution$ pinpoints its leaf CQC_E-node. On the contrary, if the CQC_E-tree is *finitely failed*, $ExpandNode(T, N_{root})$ returns “false” and $N_{root}.explanation \subseteq F_0$ is an explanation for the unsatisfiability of the tested schema validation property.

$RepairsOfGoalComparison$ and $RepairsOfc$ (called by $HandleLeaf$ in Fig. 3) return the corresponding set of repairs for the case in which the violation is in the goal and in a condition to enforce, respectively. Due to space reasons, we refer to the full

version of the paper [7] for the formalization of these algorithms and of the CQC_E -expansion rules. The intuition has already been given in Section 3.1.

4 Conclusions

We have proposed the CQC_E method, an extension of the CQC method [4] for database schema validation, aimed at providing the database designer with an explanation for why a given database schema does not satisfy a certain desirable property.

The CQC_E method computes one explanation with a single execution, at the same time that it checks whether the tested property holds. This addresses an important drawback of previous research because none of the existing methods for schema validation [3,5,9] provides any kind of explanation when the tested property fails.

We have implemented the CQC_E method in a database schema validation tool: SVT_E [2]. An experimental evaluation comparing one single execution of this implementation with one single execution of the original CQC method (as implemented in [8]) can be found in the full version of the paper [7]. Those experiments have shown that the modifications of the CQC method proposed here to obtain an explanation result in a significant efficiency improvement.

References

1. Bailey, J., Stuckey, P.J.: Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. In: PADL, pp. 174–186 (2005)
2. Farré, C., Rull, G., Teniente, E., Urpí, T.: SVT_E A Tool to Validate Database Schemas giving Explanations. In: DBTest 2008 Workshop (to appear, 2008)
3. Farré, C., Teniente, E., Urpí, T.: A New Approach for Checking Schema Validation Properties. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 77–86. Springer, Heidelberg (2004)
4. Farré, C., Teniente, E., Urpí, T.: Checking Query Containment with the CQC Method. *Data Knowl. Eng.* 53(2), 163–223 (2005)
5. Halevy, A.Y., Mumick, I.S., Sagiv, Y., Shmueli, O.: Static Analysis in Datalog Extensions. *J. ACM* 48(5), 971–1012 (2001)
6. Rull, G., Farré, C., Teniente, E., Urpí, T.: Computing Explanations for Unlively Queries in Databases. In: CIKM 2007, pp. 955–958 (2007)
7. Rull, G., Farré, C., Teniente, E., Urpí, T.: Providing Explanations for Database Schema Validation. Research Report LSI-08-14-R, Universitat Politècnica de Catalunya (2008), <http://www.lsi.upc.edu/dept/techreps/techreps.html>
8. Teniente, E., Farré, C., Urpí, T., Beltrán, C., Gañán, D.: SVT: Schema Validation Tool for Microsoft SQL-Server. In: VLDB 2004, pp. 1349–1352 (2004)
9. Zhang, X., Özsoyoglu, Z.M.: Implication and Referential Constraints: A New Formal Reasoning. *IEEE Trans. Knowl. Data Eng.* 9(6), 894–910 (1997)

Temporal Conformance of Federated Choreographies

Johann Eder^{1,2} and Amirreza Tahamtan²

¹ Alpen-Adria University of Klagenfurt, Dept. of Information Systems, A-9020
Klagenfurt, Austria

`eder@isys.uni-klu.ac.at`

² University of Vienna, Dept. of Knowledge and Business Engineering,
Rathausstrasse 19/9, A-1010 Vienna, Austria

`amirreza.tahamtan@univie.ac.at`

Abstract. Web service composition is a new way for implementing business processes. In particular, a choreography supports modeling and enactment of interorganizational business processes consisting of autonomous organizations. Temporal constraints are important quality criteria. We propose a technique for modeling temporal constraints in choreographies and orchestrations, checking whether the orchestrations satisfy the temporal constraints of a choreography and compute internal deadlines for the activities in an interorganizational workflow.

Keywords: Web Services, Composition, Choreographies, Orchestration, Temporal Conformance.

1 Introduction

A major step toward integration of web services into organizations is their composition, enabling single web services be composed to an orchestration and choreographies describing the collaboration of orchestrations. Temporal aspects are important quality criteria in business processes. Temporal constraints are envisioned as part of the negotiations for setting up choreographies. It must be ensured that activities are performed in a timely manner and right information is delivered to right activities at the right time such that overall temporal restrictions are satisfied. Choreographies and orchestrations may have deadlines which specify the latest time point in which they must finish execution. Temporal conformance checking assists organizations to provide services with a higher QoS and reduces the process costs as violation of temporal constraints leads to costly exception handling mechanisms [1].

Federated Choreographies have been introduced in [2] as a hierarchical architecture for a consistent modeling of choreographies and orchestrations. Central to the model is the notion of conformance. Structural [3], temporal, data flow, and messaging conformance are different aspects.

Here we propose an algorithm for checking of temporal constraints of federated choreographies and generate a temporal execution plan. Based on this, one can

decide whether execution of the model leads to temporal exceptions and necessary modifications can be done. The algorithm works in a distributed manner and there is no need for a central role. A choreography defines the interaction among partners, accessible activities and which activities this partner has to execute. Because of the distributed functionality of the algorithm, one partner may need data from another partner to process locally. A partner can request for data which is only associated to its accessible activities. Such activities are defined in the choreography and are public to all partners, i.e. data provider has not to reveal its private data but only required data for interaction. This enables partners to hide their internal process logic whilst allow for interaction. Temporal conformance checking has a build-time and a run-time aspect. At build time we check whether all orchestrations meet the temporal restrictions given by the choreographies. At run-time, execution is monitored to allow for predictive and pro-active time management, i.e. to diagnose potential violations of temporal constraints early enough such that counter-measures still can be taken.

Temporal aspects of web services have been studied in [4,5,6]. [4] uses temporal abstractions of protocols for compatibility and replaceability analysis based on a finite state machine formalism. [5,6] exploit an extension of timed automata for modeling time properties of web services. Our approach can cover cases modeled in these works and additionally allows for explicit deadlines. This work extends previous works by addressing the conformance and verification problem and provides an a priori execution plan at build-time (both best and worst cases) consisting of valid execution intervals for all activities with consideration of the overall structure and temporal restrictions. The calculated execution plans can be monitored at run-time.

2 Federated Choreographies

There are mainly two ways for modeling choreographies: by protocols between orchestrations and by abstract processes [7]. Protocols have the advantage of flexibility and that only a minimum of information about processes is disclosed. They have the disadvantage that the overall process is never explicitly modeled, a drawback for process awareness. Abstract processes model the choreography as a (global) process which integrates the disclosed views of the participating processes [8]. They have the disadvantage that all partners share the global process, even, if subsets of partners have closer relationships [9,10]. *Federated choreographies* [2] overcome this disadvantage by allowing abstract processes with different level of abstraction, organized in a hierarchy. The abstract processes describe a business process in various level of detail. This approach is more flexible than typical compositional approaches and provides advantages like a better business secrecy, extendability and uniform modeling. The main idea of this approach is presented in fig. 1. The upper layer is composed of shared choreographies. A choreography may support another one, i.e. the former is a partial extension of the latter. The supporting choreography is an extended subset (of activities) of the supported choreography. The support relationship

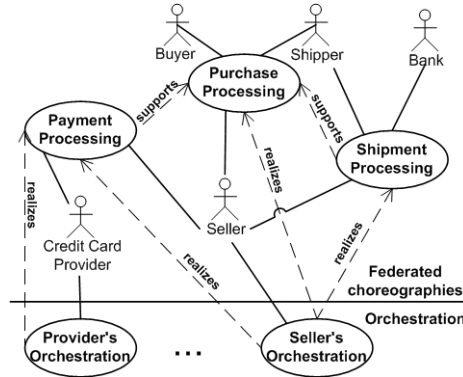


Fig. 1. Federated choreographies

is acyclic. A choreography which is only supported and realized by other nodes and in turn supports no choreography is called *the global choreography*. A node refers to a choreography and/or orchestration. Informally, the global choreography captures the core of the business process and its supporting choreographies reflect how its parts are carried out in reality. The bottom layer consists of realizing orchestrations. Each partner provides its own internal realization of relevant parts of the choreographies. The presented approach is fully distributed. Each partner has local models of all choreographies in which it participates. Additionally, each partner holds and runs its own model of the orchestration. There is no need for a centralized coordination. For a detailed discussion refer to [2,3].

Both nodes are modeled as workflows. To avoid the complex metamodel of [2], this paper is based on a simplified process model. A generic workflow model is used in this paper as a structure for representing temporal information. A workflow is a collection of activities and the dependencies between them. Activities correspond to individual steps in a process. Dependencies determine the execution sequence of activities and the data flow. Activities can be executed sequentially, in parallel and conditionally. Consequently, a workflow can be represented by a directed acyclic graph, where nodes correspond to activities and edges to dependencies. All activities have a unique name and two corresponding events. An event is either start of an activity (denoted a_s for an activity a) or its end (denoted a_e for an activity a). In this paper we model the relationship between a supporting and a supported choreography simply by event correspondence. $e_1 \equiv e_2$ denotes that event e_1 corresponds to event e_2 . Note that e_1 and e_2 may belong to different nodes.

To represent time information, the workflow model is augmented with the following temporal types: time points, durations and deadlines. We refer to such a graph as timed graph (TG) [1] (See Fig. 3 and 4)

3 Temporal Conformance

Amongst other conformance issues temporal conformance is a key requirement of the federated choreographies. It should be ensured that the flow of information and tasks is done in a timely manner with consideration of the structural dependencies. In addition it must be checked that no deadline is violated.

3.1 Prerequisites

The concepts used for calculation of temporal plans come from the field of operations research. Two kinds of constraints are used:

- **Implicit constraints** are derived implicitly from the structure of the process.
- **Explicit constraints**, e.g. assigned deadlines, can be set or enforced explicitly.

All activities of a node have durations. In this work deterministic values for durations are used. We are aware that activity durations may vary. However, we use fixed values for clarification of the concepts. We calculate an interval in which an activity can execute as described in [11]. This interval is delimited by *earliest possible start*(*eps*) and *latest allowed end*(*lae*). *eps* is the earliest point in time in which an activity can start execution and *lae* the latest point in time in which an activity can finish execution without violating the deadline. Both *eps* and *lae* are calculated for *best case* and *worst case*. The best case is given, if the shortest path is executed and the worst case when the longest path is executed.

3.2 The Proposed Approach

Fig. 1 illustrates the starting point of the algorithm. To make the presentation less complex we assume that only one global choreography exists.

The calculation of a node's TG is based on iteratively delimiting the initial intervals of activities because of implicit and explicit constraints. In addition, other nodes with a link may also impose a restriction on the TG because of additional activities present in them which may further tighten the interval. A link identifies a dependency between two nodes and is either a support or realize relationship. A valid execution interval is calculated when all constraints are considered: implicit, explicit constraints and links. The conformance condition must always be met i.e. $eps + d$ must be less or equal to its *lae* in both best and worst cases.

One important issue to consider is when one node has multiple incoming links as depicted in fig. 2. The numbers beside the arrows show the order of execution. The method *assignValuesTo* is described in subsection 3.3. Temporal values are assigned from *G* to *S1* (1), after recalculations at *S1*, they are assigned to *G* (2). An assignment may change the values of the target node. This cycle is repeated for *S2* (3,4). If *S2* again modifies *G*, the most recently modified values may again

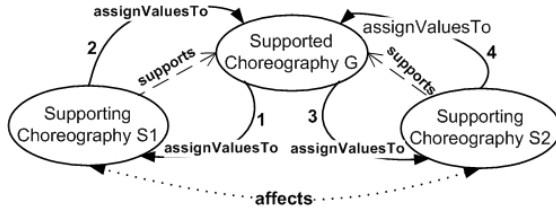


Fig. 2. Supported choreography with multiple incoming links

impose a restriction on $S1$. In other words, two or more nodes with the same supported/realized node may affect each other indirectly even with no direct link. This cycle of assignment-recalculation must be iterated for all supporting and realizing nodes of a choreography as long as a stable state is reached i.e. no values are changed after an assignment. Change of a TG can be propagated in both directions, i.e. along incoming and outgoing links. After change of the values of a node G , all of its incoming and outgoing links are marked and the recalculated values are propagated for all links with a source or target node G .

3.3 Methods

Following notations are used in the methods: $a.bc.eps$ and $a.bc.lae$ denote the best case *eps* and *lae*. $a.wc.eps$ and $a.wc.lae$ represent these values for the worst case. $a.d$ duration of an activity a . $a.pred$ and $a.succ$ set of predecessors and successors of an activity a respectively. $a.pos$ position of an activity a in the TG. a_s denotes the start-event of an activity a and a_e its end-event. $G.deadline$ deadline of a TG G . $d.max$ maximum allowed duration of a node. Upper case letters represent the nodes and lower case letters the activities.

initialize(G)

This method initializes the *eps* and *lae*-values of a node to 0 and ∞ respectively. The reason is that *eps* can only become greater and *lae* smaller.

calculate($G, G.deadline$)

This method takes as input a node and the output is the calculated TG for best and worst cases.

When recalculating a node's TG, existing *eps*(*lae*) is replaced by the recently calculated *eps*(*lae*), if the calculated values are greater(smaller) than existing values. This method is used for pre-calculation of TGs as well as for recalculation of a TG after assignment from another node.

assignValuesTo(G, H)

This method assigns values from one node to another. It uses event correspondence for assignment. Correspondence of start events is used for assignment of *eps* and correspondence of end events for *lae*.

checkConformance(G)

The above method checks if the conformance condition is fulfilled (See section 3.2)

Calculation of Timed Graphs and Temporal Conformance Checking

`temporalConformanceFederation()`

```

-initialization and precalculation-
conf := true
initialize( $C_g$ )
calculate( $C_g$ )
conf := checkConformance ( $C_g$ )
  for all directly and indirectly supporting choreographies
  and realizing orchestrations  $G$  of  $C_g$  in a topological
  order {
    initialize( $G$ )
    change := assignValuesTo( $C_g$ ,  $G$ )
    if change = true
       $G.deadline := G.first.eps + G.d.max$ 
      calculate ( $G$ )
    endif
    change: = AssignValuesTo( $G$ ,  $C_g$ )
    if change = true
      calculate( $C_g$ )
      conf := checkConformance( $C_g$ )
      mark all incoming and outgoing edges of  $C_g$ 
    endif
  }
-recalculation and conformance checking-
Repeat {
  select randomly a marked edge  $e$  such that  $G$  is the
  supported choreography and  $H$  the supporting
  choreography or realizing orchestration
  change: = AssignValuesTo( $G$ ,  $H$ )
  if change = true
    calculate  $H$ 
    conf := checkConformance ( $H$ )
    mark all incoming and outgoing edges  $\in H$ 
  endif
  change: = AssignValuesTo( $H$ ,  $G$ )
  if change = true
    calculate  $G$ 
    conf := checkConformance ( $G$ )
    mark all incoming and outgoing edges  $\in G$ 
  endif
  unmark  $e$  }
Until (all edges are unmarked  $\vee$  conf = false)

```

In the initialization and precalculation phase after initialization of the global choreography, its TG is calculated without considering other nodes. That means only implicit and explicit constraints are considered. Maximum duration $d.max$ is considered for calculating the deadline of other nodes than the global choreography which is the maximum duration during which a workflow can execute whereas a deadline denotes a point in time. Like deadlines, $d.max$ is given a

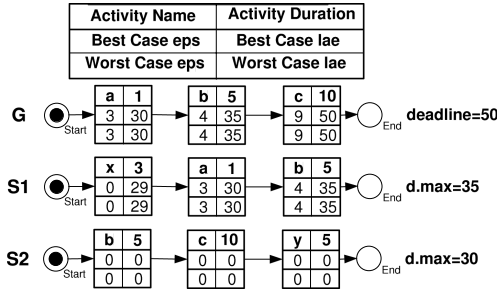


Fig. 3. After steps 1 and 2

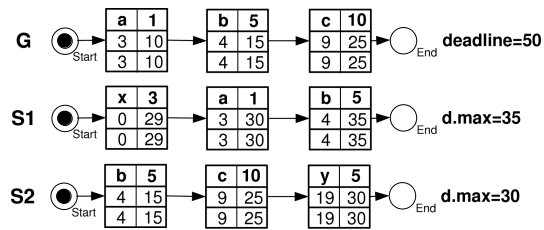


Fig. 4. After steps 3 and 4

priori. It suffices in this phase to assign the values to each node only once. These values just serve as initial values for further calculations. A variable *change* indicates the change of a TG. If *change* becomes true all incoming and outgoing links of the corresponding node are marked. Start and target node of each marked link must be revisited and recalculated if any value is changed. Multiple marks on an edge has no additional effect.

The recalculation and conformance checking phase consists of recalculation of the precalculated TGs in the previous phase. For all marked edges, the cycle of assignment-recalculation is repeated until a stable state is reached or the conformance condition is violated. A stable state is reached if all marked edges are unmarked. Figures 3 and 4 show by a numeric example how TGs are calculated. Because of space limitations a simple example is chosen. The dependency between nodes and the steps of this example are described in fig. 2. Note that fig. 3 and fig. 4 demonstrate only one cycle. Note that this cycle must be iterated as long as a stable state is reached.

We have implemented a prototype as proof of concept. The workflow specifications of nodes together with deadlines, *d.max* and links are read as input. The tool calculates the execution plan for all nodes and checks if the conformance condition is met.

3.4 Proof of Termination

We informally prove that the algorithm terminates. Algorithm terminates in two cases: if there are no marked edges or the conformance condition is violated. Because the number of edges is finite, in a finite number of steps the stable state is reached. If such a stable state does not exist, after a finite number of steps the conformance condition is violated. The reason is with each iteration, if changes are made, the *lae* becomes smaller and the *eps* greater until $eps + a.d > lae$.

4 Conclusions

Temporal quality criteria play an important role in business processes. We proposed a technique for modeling and checking of temporal constraints of choreographies and orchestrations. A time plan is generated for each choreography and orchestration representing valid execution intervals for their activities. This time plan is used at run-time for monitoring purposes and allows for pro-active time management to avoid temporal failures. The algorithm is fully distributed such that there is no need to compromise the autonomy of partners in interorganizational workflows.

Acknowledgments. This work is partly supported by the Commission of the European Union within the project WS-Diamond in FP6. STREP.

References

1. Panagos, E., Rabinovich, M.: Predictive workflow management. In: Proc. of the 3rd Int. Workshop on Next Generation Information Technologies and Systems (1997)
2. Eder, J., Lehmann, M., Tahamtan, A.: Choreographies as federations of choreographies and orchestrations. In: Proc. of CoSS 2006 (2006)
3. Eder, J., Lehmann, M., Tahamtan, A.: Conformance test of federated choreographies. In: Proc. of IESA 2007 (2007)
4. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On temporal abstractions of web service protocols. In: Proc. of CAiSE Forum (2005)
5. Kazhamiakin, R., Pandya, P., Pistore, M.: Timed modelling and analysis in web service compositions. In: Proc. of ARES 2006 (2006)
6. Kazhamiakin, R., Pandya, P., Pistore, M.: Representation, verification, and computation of timed properties in web service compositions. In: Proc. of ICWS 2006 (2006)
7. Andrews, T., et al.: Business process execution language for web services (bpel4ws). ver. 1.1. BEA, IBM, Microsoft, SAP, Siebel Systems (2003)
8. Banerji, A., et al.: Web services conversation language (wscl) 1.0. Technical report, W3C (2002)
9. Peltz, C.: Web services orchestration and choreography. IEEE Computer 36(10), 46–52 (2003)
10. Dijkman, R., Dumas, M.: Service-oriented design: A multi-viewpoint approach. Int. J. Cooperative Inf. Syst. 13(4), 337–368 (2004)
11. Eder, J., Panagos, E., Rabinovich, M.: Time Constraints in Workow Systems. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, Springer, Heidelberg (1999)

Relational Database Migration: A Perspective

Abdelsalam Maatuk, Akhtar Ali, and Nick Rossiter

School of Computing, Engineering & Information Sciences, Northumbria University,
Newcastle upon Tyne, UK

Abstract. This paper presents an investigation into approaches and techniques used for database conversion. Constructing object views on top of a Relational DataBase (RDB), simple database integration and database migration are among these approaches. We present a categorisation of selected works proposed in the literature and translation techniques used for the problem of database conversion, concentrating on migrating an RDB as source into object-based and XML databases as targets. Database migration from the source into each of the targets is discussed in detail including semantic enrichment, schema translation and data conversion. Based on a detailed analysis of the existing literature, we conclude that an existing RDB can be migrated into object-based/XML databases according to available database standards. We propose an integrated method for migrating an RDB into object-based/XML databases using an intermediate Canonical Data Model (CDM), which enriches the source database's semantics and captures characteristics of the target databases. A prototype has been implemented, which successfully translates CDM into object-oriented (ODMG 3.0 ODL), object-relational (Oracle 10g) and XML schemas.

1 Introduction

Object-oriented and Web technologies have become mainstream due to their productivity, flexibility and extensibility [9,10]. The dominance of traditional RDB and its limitation to support the benefits provided by these new technologies motivate its migration into Object-Oriented DataBase (OODB), Object-Relational DataBase (ORDB) and XML [1,2,11,12]. This paper aims to provide an investigation into the problem of DataBase Migration (DBM), to review various techniques and proposals, to identify their differences, and to assess the impact of existing literature and how it shapes current and future research in this area. We focus on the case where the input is an RDB and the outputs are OODB, ORDB and XML. Hence, we do not cover the inverse of the process (e.g., migrating OODB into RDB). Many proposals exist in the literature for handling data stored in RDBs through Object-Oriented (OO)/XML interfaces, i.e., Object-to/from-Relational (OR) and Xml-to/from-Relational (XR) mapping, connecting an existing RDB into non-RDB system that might be conceptually different, and migrating an RDB into other databases [9,15,8,16,19]. New requirements of database systems determine which technique is most suitable to adopt.

Database application migration is a process in which all components (i.e., schema, data, application programs, queries and update operations) of a source

database application are converted into their equivalents in a target database environment. However, application programs and queries conversion is a software engineering job and is, therefore, out of the scope of this paper, i.e., we assume that DBM includes schema translation and data conversion. A schema of an existing data model can be translated into an equivalent target schema expressed in the target data model through applying a set of mapping rules. The translation of a source schema to a target schema consists of two sub-phases. The first one, called DataBase Reverse Engineering (DBRE), aims to recover the conceptual schema, e.g., Entity Relationship Model (ERM), which expresses explicit and implicit data semantics of the source schema. Explicit semantics involves relation, attributes, keys and data dependencies. It is necessary to extract extra semantics that are not expressed explicitly in RDBs (e.g., relationships). The second phase, called DataBase Forward Engineering (DBFE), aims to obtain the target physical schema from the conceptual schema obtained in the first phase. However, source schema can be translated directly to a target one without intermediate representation [8]. An expert user or a tool might be required to provide missing semantics or to refine the result to exploit the target database concepts [16,8]. Data Conversion is a process of converting data from the source into the target database. Data stored as tuples in an RDB are converted into complex objects/literals in object-based databases or elements in XML document. This involves unloading and restructuring relational data, and then reloading it into a target database in order to populate the schema generated earlier during the schema translation process [9].

The remainder of this paper is organised as follows. Section 2 surveys current approaches and techniques related to database conversion. In Section 3, DBM proposals, the focus of this paper, are discussed in detail. Section 4 concludes the paper.

2 Approaches and Techniques

2.1 Conversion Approaches

There are three approaches related to database conversion. The first approach is for handling data stored in RDBs through OO/XML interfaces. Connecting an existing RDB to a conceptually different database system is the second approach. The third approach is migrating an RDB into a target database. The first and second approaches deal with schema translation, whereas in the third approach, both schema and data are completely migrated into a target database.

Viewing Objects on Top of RDBs: Data may be required to be processed in object/XML form and stored in relational form based on the concept of object for programs and RDB for persistence. A single object might be represented by several tuples in several tables, therefore, joining these tables is required for queries. However, converting these objects to tabular forms to be stored in and retrieved from RDB systems leads to a semantic gap between two different paradigms, which is known as the OR impedance mismatch. To avoid this, developers have to write huge amount of code to map objects in programs into tuples in an

RDB, which might be time-consuming to write and execute. Another solution is via using OR mapping middleware, which is a software layer that links OO Programming Languages (OOPs) concepts to data stored in RDBs through ODBC or JDBC drivers. Similarly, RDBs data can be published as XML documents using special declarative languages to be exchanged over the Web, with which users see views that can be queried using XML query languages. However, mapping using middleware requires schema mapping time.

Database Integration: A connection could be established between RDBs and other databases allowing the applications built on top of new DBMS accessing both relational and object/XML DBMSs giving an impression that all data are stored in one database. This presents a simple level of DataBase Integration (DBI). This is achieved using a special type of software called *Gateways*, which support connectivity between DBMSs and do not involve the user in SQL and RDB schema. Hence, queries and operations are converted into SQL and the results are translated into target objects. Most commercial DBMSs provide flexibility on gateways construction among heterogeneous databases. The difference between Gateways and mapping tools is that, in Gateways, objects are persistently stored in the new target database system, whereas in the mapping, objects are created and handled in the normal way but are stored in an RDB. However, in both approaches old data, stored in an RDB, is retained.

Database Migration: Migration of an RDB into its equivalents is accomplished in the literature for two databases. The first database is an RDB, the *source*, and the second database, the *target*, represents the result of DBM process. The process is performed with or without the help of an Intermediate Conceptual Representation (ICR), e.g., ERM. The input source schema is enriched semantically and translated into a target schema. Generally, relations and attributes are translated into equivalent targets. Foreign Keys (FKs) may be replaced by another domain or relationship attributes. Relationships can be extracted by analysing data dependencies or database instances. Data stored in the source database is converted into the target database. FKs realise relationships between tuples, which are converted into value-based or object identifier references. The challenge in this process is that data of one relation may be converted into a collection of literal/references rather than into one corresponding type. This is because of the heterogeneity between the concepts and structures of source and target data models.

2.2 Translation Techniques

Existing techniques can be classified into two types: Source-to-Target (ST) and Source-to-Conceptual-to-Target (SCT).

ST Technique: This type of technique translates a physical source code into an equivalent target. However, as the target schema is generated using one-step mapping with no ICR for enrichment, this technique usually results in an ill-designed database as some of the data semantics are ignored. This approach could take the following forms:

Flat Technique: This technique converts each relation into object class/XML element in target database [9,16]. FKs are mapped into references to connect objects. However, the flattened form of RDBs is preserved in the generated database, with which object-based model features and the hierarchical form of XML model are not exploited. This means that the target database is semantically weaker and of a poorer quality than the source. Moreover, creating too many references cause degraded performance during data retrieval.

Clustering Technique: This technique is performed recursively by grouping entities and relationships together starting from atomic entities to construct more complex entities until the desired cluster is achieved, which is labelled with the strong entity name [19]. However, this technique may lead to complex structures, data redundancy and is prone to error in translation.

Nesting Technique: This technique uses the iterated mechanism of *nest* operator to generate a nested target structure from relational inputs [7]. The target is extracted from the best possible nesting outcome. However, the technique has some limitations, e.g., mapping each table separately and ignoring integrity constraints. Besides, the process is quite expensive, as it needs all tuples of a table to be scanned repeatedly to get the best possible nesting.

SCT Technique: This type of technique enriches a source schema by semantics that might not have been clearly expressed in it and their interrelationships. Then, the schema is translated from logical into conceptual through recovering the domain semantics and making them explicit. Finally, the results are represented as a conceptual schema, which can be translated into the target effectively. In this way the technique results in a good well-designed target database. Inferring conceptual schema from a logical RDB schema has been extensively studied by many researchers based on analysing schema, data and queries. Chiang *et al.* presented a method for extracting an Extended ERM (EERM) from an RDB [5] through derivation and evolution of key-based inclusion dependencies. Alhajj developed algorithms for identifying candidate keys to locate FKs in an RDB using data analysis [1]. Andersson extracts a conceptual schema by investigating equi-join statements [3]. The approach uses a join condition and the *distinct* keyword for attribute elimination during key identification.

3 Migrating RBD into OODB/ORDB/XML

This section discusses proposals for migrating RDBs into OODB/ORDB/XML databases. Table 1 shows a comparison of some proposals showing input, output, technique used, data semantics, prerequisites and features of DBM process.

Migrating RBD into OODB: Several methods have been proposed for migrating RDBs into OODBs without using an ICR [16,4,8,9]. Premerlani and Blaha propose a procedure for mapping an RDB schema into an OMT schema [16]. They produce an initial schema and determine Primary Keys (PKs) and FKs by resolving synonyms and homonyms. Then, horizontally partitioned classes are refined, and relationships are identified using keys evaluation. Fahrner

Table 1. RDB migration (prerequisites, features, input and output databases)

Proposal	ST	DC	Tec	Data Semantics					Input	Prerequisites	Features		Output		
				AS	AG	IN	RI	OP			SA	UI	OODB	ORDB	XML
[9]	✓	✓	ST	✓	×	✓	×	×	RDB	FD,ID,ED	×	H	✓	×	×
[8]	✓	×	ST	✓	✓	✓	✓	✓	RDB	keys,FD,ID,3NF	✓	H	✓	×	×
[2]	✓	✓	SCT	✓	✓	✓	×	×	RDB	keys,DD,Ins	×	L	✓	×	×
[15]	✓	×	ST	✓	✓	×	×	×	ERM	ERM	×	H	✓	×	×
[16]	✓	×	ST	✓	✓	✓	×	✓	RDB	keys,non-3NF	×	H	✓	×	×
[4]	✓	×	ST	✓	✓	✓	×	×	RDB	FD,ID,ED,non-3NF	×	H	✓	×	×
[18]	✓	×	ST	✓	✓	×	✓	✓	UML	UML class diagram	×	-	×	✓	×
[14]	✓	×	ST	✓	✓	✓	✓	✓	UML	UML class diagram	✓	-	×	✓	×
[10]	✓	✓	SCT	✓	✓	✓	✓	×	RDB	PKs,FKs	✓	L	×	×	✓
[12]	✓	✓	ST	✓	×	×	×	×	EERM	FD,ID	✓	H	×	×	✓
[6]	✓	×	SCT	✓	✓	✓	✓	×	RDB	3NF	✓	H	×	×	✓
[11]	✓	✓	SCT	✓	✓	×	×	✓	RDB	FD,MVD,JD,TD	✓	L	×	×	✓
[7]	✓	×	ST	✓	✓	×	×	✓	RDB	PKs,FKs	✓	H	×	×	✓

ST: Schema Translation DC: Data Conversion MVD: Multi-valued Dependency TD: Transitive Dependency Ins: Data instances FD: Functional Dependency ED: Exclusion Dependency ID: Inclusion Dependency JD: Join Dependency UI: User Interaction SA: Standard Adoption L: Low consideration H: High consideration Tec: Technique AS: Association AG: Aggregation IN: Inheritance RI: Referential Integrity OP: Optimization ✓: Yes ×: No

and Vossen propose a method in which an RDB schema is normalised to 3NF, enriched by semantics using data dependencies and translated into an ODMG-93 OODB schema [8]. Moreover, the resulting schema is then restructured (by the user) with respect to OO paradigm options, e.g., binary relationship relations are eliminated and integrity constraints are mapped into class methods. Castellanos *et al.* present a method that improves an RDB schema semantically (by analysing the schema and data) and converts it into an object-based schema, called BLOOM schema [4]. Narasimhan *et al.* propose a procedure for mapping an ERM into an OO schema [15]. The approach suggests creating a separate constraint class as a subclass for each of OODB classes. Yan and Ling present a method that produces an OODB schema from an RDB using clustering technique [19]. A cluster of relations is identified from a main relation and its component/subclass relations, which are not participating in relationships with other relations. Besides, the method proposes generating OIDs for identified objects by concatenating the key of each tuple with the relation name. Alhajj and Polat re-engineer an RDB into an OODB using an RID graph as an ICR [2]. The graph, which is similar to EERM is derived and optimised for identifying relationships. Finally, RDB tuples are migrated into objects in OODB.

Migrating RDB into ORDB: A number of researchers have considered exploiting user-defined types in Oracle 8_i/9_i and SQL3 from conceptual models [18,14]. The logical structure of an ORDB schema is achieved by creating object-types using UML, based on which tables are created to store data. Multi-valued attributes are defined using arrays. An association relationship is mapped using REF/collection of REFs. An inheritance is defined using FKs or REF types in Oracle 8_i and using the UNDER clause in Oracle 9_i/SQL3 [14]. Although most ORDB concepts are presented in these proposals, they are aimed at producing an ORDB schema from conceptual models rather than DBM. However, if a DBM

process uses a conceptual model as an ICR then these proposals could be useful in schema translations.

Migrating RDB into XML: Fong and Cheung introduce a method, in which data semantics are extracted from an RDB into an EERM, which is then mapped into an XSD graph. An XML logical schema is extracted from the XSD graph [10]. The authors suggest mapping FKs into element hierarchy, which may cause redundancy when an element has a relationship with more than one element. Kleiner and Lipeck translate an ERM to DTD [12]. However, some data semantics cannot be represented, e.g., the limitation of DTD in specifying composite keys. Vela and Marcos propose an approach for extending UML to represent an XML Schema in graphical notation, which has a unique equivalence with XML Schema [20]. Du *et al.* propose translation rules for converting an enriched RDB schema into a semi-structured model, called ORA-SS, which is then translated into XML Schema [6]. However, they adopt an exceptionally deep clustering technique, which is prone to errors. Fong *et al.* propose a procedure to translate RDB views into XML documents [11]. The approach de-normalises an RDB into joined tables and translates them Document Object Models (DOMs), which are integrated into one DOM, which is then mapped into an DTD schema. Based on the generated DTD schema and data dependencies, each tuple of the joined tables is loaded into an instance in DOM and then transformed into an DTD document. Lee *et al.* present two algorithms, called NeT and CoT, to translate an RDB schema to DTD using a language named XSchema [7]. The CoT algorithm is proposed to remedy the drawbacks of NeT, e.g., the mapping of each table separately and not taking into account integrity constraints.

4 Discussion

In this paper, we have presented a survey of existing approaches and techniques used for database conversion. Our investigation into DBM problem shows that different proposals have different focuses. Each proposal has some assumptions to facilitate the process, which might be a point of limitations or a drawback. While existing works for migrating into OODBs focus on schema translation using ST techniques, we note that most works for migrating to XML are following SCT techniques, focusing on generating a DTD schema and data. Moreover, all researches on the generation of ORDBs are focused on design rather than migration. It could be concluded, based on our analysis of the literature, that there are several areas in need of more attention for migrating RDBs to object-based/XML databases.

Due to focusing on schema rather than data, proposals either ignore data loading or assume working on virtual target databases and data remain stored in RDBs. Moreover, there are still shortcomings in implementation of loading an RDB data to more than one environment. Using middleware may lead to slow performance making the process expensive at run-time because of dynamic mapping of tuples to complex objects. However, using object-based DBMSs and

native-XML, objects can be stored and retrieved directly without any need for translation layers, hence saving development time and improving performance.

Some semantics (e.g., inheritance) have not been considered during DBM. ERM and DTD do not support inheritance. Despite UML's ability to model data semantics such as aggregation and inheritance, UML is still weak to handle the hierarchical structure of the XML data model [10]. UML should be extended by adding new stereotypes to specify ORDB and XML models features [14,20]. Although generalization/specialization and categorization could be realized in an RDB, they have been either ignored or briefly mentioned without delving into its different types, e.g., union and multiple inheritance, and its constraints. Translating inheritance relationships from RDBs to object-based/XML databases needs more attention. Standard adoption is essential for more portability and flexibility. In the ODMG 3.0 model, referential integrity is maintained via inverse references. SQL4 has an ability to address complex objects in ORDBs. Compared to DTD, XML Schema offers a much more extensive set of data types, and provides a powerful referencing, nesting and inheritance mechanisms of attributes and elements.

Most of the existing proposals and techniques generate a database that is either flat relational or has a deep level of clustering/nesting. It would be desirable to avoid the flattened form and reduce clustering levels of objects structure to the lowest in order to increase utilisations of advantages that target models provide and to avoid undesirable redundancy. This requires preservation of semantics of the source database and relocating them into an ICR, which takes into account the relatively richer data model of the target database.

The Way Forward: The existing work does not provide a solution for more than one target database or for either schema or data conversion. Besides, none of the existing proposals can be considered as a method for migrating an RDB into an ORDB. Several challenges could arise when a DBM process aims at several target databases, which are fundamentally different and have different design characteristics. An integrated method, which deals with migration from RDB to OODB/ORDB/XML covering both schema and data is not yet in existence. We propose a complete method [13], which is able to preserve the structure and semantics of an existing RDB in a CDM, to generate OODB/ORDB/XML schemas, and to find an effective way to load data into target databases without lose or unnecessary redundancies. The method is superior to the existing proposals as it can produce three different output databases. Besides, the method exploits the range of powerful features that target data models provide such as ODMG 3.0, SQL4, and XML Schema. A system architecture is designed and a prototype has been implemented, which resulted successfully in target databases.

References

1. Alhajj, R.: Extracting the Extended Entity-Relationship Model from a Legacy Relational Database. *Info. Syst.* 28, 597–618 (2003)
2. Alhajj, R., Polat, F.: Reengineering Relational Databases to Object-Oriented: Constructing the Class Hierarchy and Migrating the Data. In: *WCRE 2001*, pp. 335–344 (2001)

3. Andersson, M.: Extracting an Entity Relationship Schema from a Relational Database through Reverse Engineering. In: 13th Int. Conf. on the ER Approach, pp. 403–419 (1994)
4. Castellanos, M., Salto, F., García-Solaco, M.: Semantically Enriching Relational Databases into an Object Oriented Semantic Model. In: Karagiannis, D. (ed.) DEXA 1994. LNCS, vol. 856, pp. 125–134. Springer, Heidelberg (1994)
5. Chiang, R.H., Barron, T.M., Storey, V.C.: Reverse Engineering of Relational Databases: Extraction of an EER Model from a Relational Database. *Data Knowl. Eng.* 12, 107–142 (1994)
6. Du, W., Lee, M., Ling, T.W.: XML Structures for Relational Data. In: WISE (1), pp. 151–160 (2001)
7. Lee, D., Mani, M., Chiu, F., Chu, W.W.: NeT and CoT: Translating Relational Schemas to XML Schemas using Semantic Constraints. In: CIKM, pp. 282–291 (2002)
8. Fahrner, C., Vossen, G.: Transforming Relational Database Schemas into Object-Oriented Schemas according to ODMG 1993. In: Ling, T.-W., Vieille, L., Mendelson, A.O. (eds.) DOOD 1995. LNCS, vol. 1013, pp. 429–446. Springer, Heidelberg (1995)
9. Fong, J.: Converting Relational to Object-Oriented Databases. *SIGMOD Record* 26, 53–58 (1997)
10. Fong, J., Cheung, S.K.: Translating Relational Schema into XML Schema Definition with Data Semantic Preservation and XSD Graph. *Info. & Soft. Tech.* 47, 437–462 (2005)
11. Fong, J., Wong, H.K., Cheng, Z.: Converting Relational Database into XML Documents with DOM. *Info. & Soft. Tech.* 45, 335–355 (2003)
12. Kleiner, C., Lipeck, U.W.: Automatic Generation of XML DTDs from Conceptual Database Schemas. *GI Jahrestagung* (1), 396–405 (2001)
13. Maatuk, A., Ali, A., Rossiter, N.: A Framework for Relational Database Migration. *TR* (2008), <http://computing.unn.ac.uk/staff/cgma2/papers/RDBM.pdf>
14. Marcos, E., Vela, B., Cavero, J.M.: A Methodological Approach for Object-Relational Database Design using UML. *Soft. and Syst. Modeling* 2, 59–75 (2003)
15. Narasimhan, B., Navathe, S.B., Jayaraman, S.: On Mapping ER Models into OO Schemas. In: 12th int. Conf. on the Entity-Relationship Approach, vol. 823, pp. 402–413 (1993)
16. Premerlani, W.J., Blaha, M.R.: An Approach for Reverse Engineering of Relational Databases. *Communications of the ACM* 37, 42–49 (1994)
17. Soutou, C.: Inference of Aggregate Relationships through Database Reverse Engineering. In: Ling, T.-W., Ram, S., Li Lee, M. (eds.) ER 1998. LNCS, vol. 1507, pp. 135–149. Springer, Heidelberg (1998)
18. Urban, S.D., Dietrich, S.W., Tapia, P.: Succeeding with Object Databases: Mapping UML Diagrams to Object-Relational Schemas in Oracle 8, pp. 29–51. John Wiley and Sons, Ltd, Chichester (2001)
19. Yan, L., Ling, T.W.: Translating Relational Schema with Constraints into OODB Schema. In: The IFIP WG 2.6 Database Semantics Conf. on Interoperable Database Systems, vol. A-25, pp. 69–85 (1993)
20. Vela, B., Marcos, E.: Extending UML to Represent XML Schemas. In: CAiSE Short Paper Proceedings (2003)
21. Zhang, X., Zhang, Y., Fong, J., Jia, X.: Transforming RDB Schema to Well-structured OODB Schema. *Info. & Soft. Tech.* 41, 275–281 (1999)

DB-FSG: An SQL-Based Approach for Frequent Subgraph Mining*

Sharma Chakravarthy and Subhesh Pradhan

IT Laboratory & Department of Computer Science and Engineering
The University of Texas at Arlington, Arlington, TX 76019
{sharma@cse.uta.edu, subhesh_kp}@yahoo.com

Abstract. Mining frequent subgraphs (FSG) is one form of graph mining for which only main memory algorithms exist currently. There are many applications in social networks, biology, computer networks, chemistry and the World Wide Web that require mining of frequent subgraphs. The focus of this paper is to apply relational database techniques to support frequent subgraph mining. Some of the computations, such as duplicate elimination, canonical labeling, and isomorphism checking are not straightforward using SQL. The contribution of this paper is to efficiently map complex computations to relational operators. Unlike the main memory counter parts of FSG, our approach addresses the most general graph representation including multiple edges between any two vertices, bi-directional edges, and cycles. Experimental evaluation of the proposed approach is also presented in the paper.

1 Introduction

Frequent subgraphs (FSG) is one form of graph mining. However, for FSG mining there currently exist only main memory algorithms [4]. There are many applications in social networks, biology, computer networks, chemistry and the World Wide Web that require mining of frequent subgraphs over large data sets. These main memory algorithms do not scale very well for large data sets. Hence, there is a need for developing scalable algorithms for frequent subgraph mining. An SQL-based approach [9,5] is one way of doing that by exploiting the buffer management and optimization techniques already provided and fine tuned in a RDBMS. However, applying limited representation and computations provided by a RDBMS for graph mining is not trivial. Representation of a graph, generation of larger subgraphs, checking for exact and inexact matches of subgraphs using relational representation and operators is one of the contributions of this paper.

The remainder of the paper is organized as follows. The different graph mining algorithms that motivated the development of a SQL-based approach for frequent subgraph mining is discussed in section 2. An overview of DB-FSG algorithm

* This work was supported, in part, by Air Force grant F30602-01-2-0570 and NSF (grants EIA-0216500, IIS-0326505, MRI 0421282, and IIS 0534611).

for FSG is provided in section 3. The design issues related to DB-FSG algorithm is detailed in section 4. Experimental results are discussed in section 5. Finally, conclusions and future work are discussed in section 6.

2 Related Work

Subdue [2] is one of the earliest graph mining algorithms that detects the best substructure using the minimum description length principle [8]. It also mines for interesting concepts, detect anomalies, and similarities between graph structures. FSG [4] and others [10,3] are main memory algorithms that mine graph sets to discover frequent subgraphs. FSG uses canonical labeling to determine subgraph isomorphism. It considers an undirected graph representation without multiple edges (between two vertices) or cycles. Hence, FSG cannot mine over general forms of directed graphs, graphs with multiple edges, and cycles. gSpan [11] is another frequent subgraph mining approach which uses depth first search and generates lesser candidate items than FSG. The depth-first traversal and book keeping requires special data structures and is not clear how it can be mapped using relational operators.

DB-Subdue [1] and HDB-Subdue [6] (SQL-based versions of Subdue) detect interesting subgraphs that compress a graph (or a forest) maximally using the minimum description length (or MDL) principle. HDB-Subdue handles multiple edges, cycles, and hierarchical reduction to deal with a general graph. However, HDB-Subdue did not support mining over a set of input graphs to discover frequent subgraphs.

3 Overview of DB-FSG

Normally, graphs are represented as a set of edges and vertices. DB-FSG represents graphs using two relations: i) a vertex table and ii) an edge table which store the vertices and the edges of the graph, respectively. For the set of graphs shown in Figure 1(a), the corresponding *vertex* and *edge* tables are shown in Figures 1(b) and 1(c), respectively. Graph Id (in short GID) attribute in the tables helps to identify the edges and vertices belonging to the same graph.

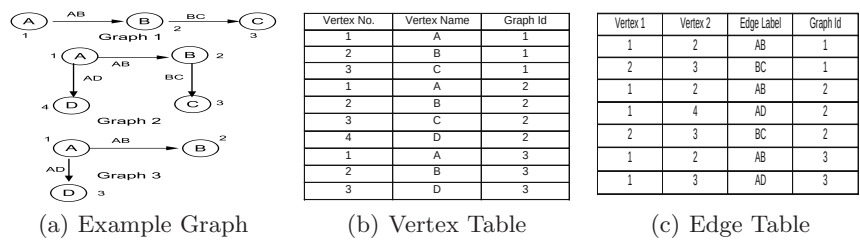


Fig. 1. Sample Graph and Corresponding Vertex and Edge Table

Vertex 1	Vertex 2	Edge No.	Edge Label	Vertex 1 Name	Vertex 2 Name	Graph Id
1	2	1	AB	A	B	1
2	3	2	BC	B	C	1
1	2	3	AB	A	B	2
2	3	4	BC	B	C	2
1	4	5	AD	A	D	2
1	2	6	AB	A	B	3
1	3	7	AD	A	D	3

Fig. 2. Oneedge Table

As the edge table does not contain information about vertex labels, tuples of edge table cannot represent substructures of size one. Hence, we create a new relation called *oneedge* by joining the vertex and the edge tables as shown in Figure 2. The *oneedge* table will contain all the instances of substructures of size one as tuples. For a one-edge substructure, the edge direction is always from the first vertex to the second vertex. Hence, there are no attributes in the *oneedge* table which specify the direction. For a higher edge substructure, we introduce connectivity attributes to denote the direction of edges between the vertices of the substructure. The *oneedge* table is the base table that will be used for generating higher size substructures. For each edge in the *oneedge* table, we assign a unique identifier called the edge number.

We need to systematically generate subgraphs of increasing size in all the input graphs and obtain the count for the isomorphic substructures across the graphs. To expand a one-edge substructure to a two-edge substructure, we join *oneedge* relation with itself on matching vertices. To ensure that the expansion is done within the same graph, we impose a constraint that the GID (each graph has an id termed GID) of both one-edge substructures should be same. We term the resulting two-edge substructure table as *instance_2*. In general, substructures of size *i* are generated by joining *instance_(i-1)* relation with *oneedge* relation. In order to avoid expansion of instances on edges that are already present (remember that our approach unlike FSG handles multiple edges and cycles), we impose the rule that the new edge being added should not have the same edge number as the edge already present in the substructure instances. In case of substructures that have two or more edges, we would need attributes to denote the direction of the edges. The From and To (F and T for short) attributes in the *instance_n* table serve this purpose. An *n*-edge substructure is represented by *n*+1 vertex numbers, *n*+1 vertex labels, *n* edge numbers, *n* edge labels, and *n* From and To pairs. In general, *6n*+3 attributes are needed to represent an *n*-edge substructure. Though edge numbers are part of every *instance_n* table, owing to the space constraint, we will be showing it only in sections where they are necessary.

4 Details Of DB-FSG Approach

DB-FSG algorithm is shown below. Some of the major aspects - *new substructure instance representation, unconstrained expansion and pseudo duplicate*

elimination, canonical label ordering and frequency counting and substructure pruning are elaborated further. For a comprehensive description, refer to [7].

Algorithm 1. DB-FSG Algorithm

```

1: Create oneedge (instance1) table by joining vertex table and edge table
2: Remove the edges with instance count less than support from the oneedge table
3: for n=2 to MaxSize do
4:   Join instance(n-1) with oneedge table to generate instancen
5:   Eliminate pseudo duplicates from instancen table
6:   Canonically order instancen table on vertex labels
7:   Project distinct vertex label, edge label and gid to obtain one instance per substructure for
   each graph and store in distn table.
8:   Group distn table by vertex label and edge label to obtain substructures and its count
9:   Retain only the instances of substructure satisfying support and store it in instancen table
10:  If there are no instances of substructure satisfying support then stop
11: end for
  
```

The algorithm starts by creating one-edge substructure instance by joining vertex table and edge table as shown in the step 1 of algorithm 1. The *oneedge* instances with frequency less than the user specified support value are pruned. The remaining one edge instances are expanded to two-edge instances and the two-edged substructure instances having frequency less than the support value are pruned. As shown in steps 3 to 11 of the algorithm 1, the expansion and pruning of sub-graphs continues till user specified MaxSize is reached or until the subgraphs cannot be expanded any further. Due to the unconstrained expansion, the same substructure may be generated in many ways. Hence, pseudo duplicate elimination is required to remove such duplicates as mentioned in step 5 of the algorithm 1. Also, due to unconstrained expansion similar substructure instances may be generated in different order. Hence, canonical ordering is performed in step 6 of the algorithm 1 to identify such substructures instances. Similarly, to get the correct frequency of the substructures, substructure counting and pruning is done in steps 7, 8 and 9 of the algorithm 1.

DB-FSG [7] represents a substructure as a tuple of a relation. The repeating vertex number of a cycle or a multiple edge is marked by ‘0’ and the corresponding vertex label is marked by ‘-’, respectively. The marking of repeating vertices avoids redundant expansion on the same vertex. This form of representation can represent most general forms of a graph including cycles and multiple edges.

However, this representation is not sufficient to represent a set of graphs. For example, this representation cannot represent a set of graphs shown in In DB-FSG, we need to distinguish between graphs in which the same substructure

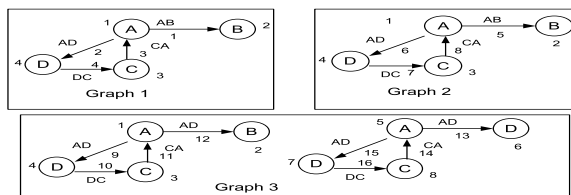


Fig. 3. DB-FSG graph representation

appears. Hence, we have added one more attribute (Graph ID or GID) to denote which graph a substructure belongs to. Each graph is assigned a unique GID and all the substructures belonging to same graph will have the same GID. New substructure instance representation of size two for Figure 3 is shown in table 1.

Table 1. DB-FSG instances

V1	V2	V3	V4	VL1	VL2	VL3	VL4	EL1	EL2	EL3	GID	F1	T1	F2	T2	F3	T3
1	2	3	4	A	B	C	D	AB	AD	CA	1	1	2	3	1	1	4
1	3	4	0	A	C	D	-	AD	DC	CA	1	1	2	2	3	3	1
1	2	3	4	A	B	C	D	AB	AD	CA	2	1	2	3	1	1	4
1	3	4	0	A	C	D	-	AD	DC	CA	2	1	2	2	3	3	1
1	2	3	4	A	D	C	D	AB	AD	CA	3	1	2	3	1	1	4
1	3	4	0	A	C	D	-	AD	DC	CA	3	1	2	2	3	3	1
5	6	7	8	A	D	C	D	AB	AD	CA	3	1	2	3	1	1	4
5	6	7	0	A	C	D	-	AD	DC	CA	3	1	2	2	3	3	1

Unconstrained expansion generates all possible substructures in an arbitrary graph input. However, this unconstrained expansion also results in the same instance to be generated in different order (will be termed pseudo duplicates). In order to identify same instances that grew in different order, we have implemented pseudo duplicate elimination by constructing an edge code that is unique to an instance. We have introduced a new attribute called `ecode` in `instance_n` table. This attribute will store edge code of each instance in the table. Then by comparing `ecodes`, we can identify and remove pseudo duplicates more efficiently. Details can be found in [7].

4.1 Canonical Ordering

In order to identify two similar substructure instances, vertex labels and the connectivity attributes need to be used (unlike vertex numbers or edge numbers for pseudo duplicate elimination). If two instances have same vertex labels and edge directions, then they can be identified as similar (or isomorphic) instances. In SQL, we can identify similar substructures only if the vertex labels and connectivity map of each tuple is canonically ordered. Since databases do not allow rearrangement of columns (only rows by using `group by` and `order` clauses), to obtain canonical ordering, we have to transpose the rows of each substructure into columns, sort and reconstruct them to get the canonical order. To facilitate construction of canonically ordered `instance_n` table, we introduce an additional attribute called `ID` in unordered `instance_n` table. Each instance (tuple) in the `instance_n` table should have unique `ID` for which `rownum` is used as `ID` value. Table 2 shows `instance.2` table for Fig 3 before canonical ordering.

Owing to the table space constraints, canonical ordering of only the second and third instance are shown below. We project the vertex numbers and vertex names from the instance table and insert them row wise into a relation called `unsorted` as shown in Fig 4(a). We also include the position in which the vertex occurs in the original instance. To differentiate between the vertices of different instances, we

Table 2. Before Canonical Ordering

ID	V1	V2	V3	VL1	VL2	VL3	EL1	EL2	GID	F1	T1	F2	T2
1	1	3	4	A	C	D	AD	DC	1	1	3	3	2
2	1	4	3	A	D	C	AD	DC	2	1	2	2	3
3	3	4	1	C	D	A	DC	AD	3	2	1	3	1

carry the Id value from the instance table onto the unsorted table. Next, we sort the table on Id and vertex label and insert it into a table called Sorted as shown in Fig 4(b) with its New attribute pointing to the new position of the vertex and the attribute Old pointing to the old position of the vertex. Similarly, the

Id	V	VL	Pos
2	1	A	1
2	4	D	2
2	3	C	3
3	3	C	1
3	4	D	2
3	1	A	3

(a) Unsorted Table

Id	V	VL	Old	New
2	1	A	1	1
2	4	D	2	2
2	3	C	3	3
3	3	C	1	1
3	4	D	2	2
3	1	A	3	3

(b) Sorted Table

Id	EL	F	T
2	AD	1	2
2	DC	2	3
3	DC	2	1
3	AD	3	1

(c) Old_Ext Table

Fig. 4. Canonical Ordering Intermediate Tables

connectivity attributes are also transposed into a table called Old_Ext as shown in Fig 4(c). Since the sorting on vertex numbers has changed its position, we need to update the connectivity attributes to reflect this change. Therefore, we do a 3 way join of Sorted and Old_Ext tables on the Old attribute of the Sorted table to get the updated connectivity attributes which we call New_Ext as in Tab 3. Next, we sort the New_Ext table on Id and the attributes F (From vertex) and T (Terminating vertex). Since, we also need ecode and GID attributes for

Table 3. New_Ext

Id	EL	F	T
2	AD	1	3
2	DC	3	2
4	DC	3	2
4	AD	1	3

Table 4. Sorted_Ext

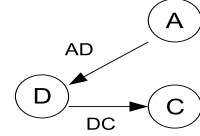
Id	EL	F	T
2	AD	1	2
2	DC	3	2
4	AD	1	3
4	DC	3	2

expansion of the instances, the ecode and GID attribute were also transposed to tables called label_ecode_n and label_GID_n. To differentiate between the GID and ecode of different instances, we carry the Id value from the instance table onto the respective tables.

Now, we have the ordered vertex as well as connectivity map tables. Hence, we can do a $2n+3$ way join (where n is current substructure size) of $n+1$ Sorted tables, label_GID_n table, label_ecode_n table and n Sorted_Ext tables to reconstruct the original instance in a canonical order. Table 5 shows the substructures after canonically ordering vertex numbers and connectivity attributes.

Table 5. Instance table - After canonical ordering

V1	V2	V3	VL1	VL2	VL3	EL1	EL2	GID	F1	T1	F2	T2
1	3	4	A	C	D	AD	DC	1	1	3	3	2
1	3	4	A	C	D	AD	DC	2	1	3	3	2
1	3	4	A	C	D	AD	DC	3	1	3	3	2



(a) Graph with multiple instances

V1	V2	V3	VL1	VL2	VL3	EL1	EL2	GID	F1	T1	F2	T2
1	3	4	A	C	D	AD	DC	1	1	3	3	2
1	3	4	A	C	D	AD	DC	2	1	3	3	2
1	3	4	A	C	D	AD	DC	3	1	3	3	2
5	7	8	A	C	D	AD	DC	3	1	3	3	2
.	.	.	.	.	.	.	.	.	.	.	.	.

(b) Instance table with multiple instances

Fig. 5. Multiple Instances of same substructure

4.2 Frequency Counting and Substructure Pruning

A graph may have many instances of the same substructure. For example, if we consider substructure in Fig 5(a), it has two instance in graph 3 of Fig 3. Fig 5(b) shows the instances of substructure in Fig 5(a). If we count the frequency of the substructure from the instance table, it will give a count of four. Even though, the correct frequency count across the graph set is three. Hence, to obtain the correct frequency of a substructure in the graph set, we need to include only one instance per substructure within a graph. However, we need to preserve all instances of a substructure that satisfy the support condition for future expansion. In order to get one instance per substructure of size n , we project *distinct* vertex labels, edge labels, connectivity map and GID and store it into `dist_n` table. Then, a GROUP BY operation on vertex labels, edge labels and connectivity map in `dist_n` table will provide the correct frequency of each substructure. Since, the substructures having less frequency than support value will not contribute to future expansion, we can store only those substructures that satisfies the support value in `sub_fold_n` table. Then, we can prune the `instance_n` table by removing instances of substructures that are not in `sub_fold_n` table.

5 Experimental Analysis

The experiments were conducted on a Linux machine (running on dual processors with 2.4 GHz CPU speed and 2 GB memory) of the Distributed and Parallel Computing Cluster at UTA (DPCC@UTA) using Oracle 10g.

For the comparison of DB-FSG with FSG, we performed experiments on data sets containing 100K - 300K graphs, with each graph containing 30 to 50 edges and 30 to 50 vertices. The support value was set to 1% and the maximum substructure size (MaxSize) was set to 5. When we tried to compare DB-FSG with FSG, it was observed that FSG was not able to detect all the frequent

patterns in the input graphs and failed to detect all the embedded subgraphs. Also, FSG does not handle multiple edges and cycles. Hence, the results are not discussed here. Other main memory FSG systems

Other set of experiments were performed to analyze the performance of the DB-FSG algorithm for different data sets, for various support values, and for different types of graphs (that is, simple graph without cycles and multiple edges, graph with cycles, and graph with multiple edges). Each graph in the data set has 40 edges and 40 vertices. The data sets are varied from 50K of graphs (2 million vertices and 2 million edges in the data set) to 300K of graphs (12 million vertices and 12 million edges in the data set). The frequent substructures embedded in the data set had support value of 3% and 4%. The parameters used for the set of experiments were MaxSize - 5, Support - 1%. Fig 6(a) gives the

Type Graph	Time (seconds)		
	Simple	Cycles	Multiple Edges
50K	391.23	431.74	560.8
100K	1510.4	1516.365	1735.8
150K	2572.61	2313.04	2639.49
200K	3680.08	3233.4	3535.39
250K	4663.78	4387.89	4590.78
300K	5692.28	5297.8	5604.93

(a) Performance of DB-FSG on different graphs

Graph Support	Time (seconds)		
	100K	200K	300K
1%	1892.89	3912.11	6088.9
3%	1679.05	3697.68	5722.07
5%	1516.64	3280.12	5374.15
7%	1064.64	2224.03	4323.82

(b) Performance of DB-FSG for varying support

Fig. 6. Summary of Experiments

processing time required by DB-FSG on data sets containing graphs without cycles and multiple edges, graphs with cycles and graphs with multiple edges. The experimental results showed that the processing time of algorithm increases linearly as the size of the data set grows. The number of substructures instances discovered in data set containing graph with cycles are lesser than the graphs without cycles and graphs with multiple edges. Hence, the processing time of the data sets containing cycles was less than the graphs without cycles and graphs with multiple edges.

Then, we conducted experiments to analyze the performance of the algorithm for varying support value. The frequent substructures embedded in the data sets had support value of 1%, 3%, 5% and 7%. We varied the support value from 1% to 7% (keeping the MaxSize as 5) in order to evaluate the performance of the DB-FSG on those data sets. Figure 6(b) gives shows the relation of support value with the processing time. The experimental results showed that the processing time decreased as the support value increased. For greater support value, more substructures will be pruned in earlier iterations of the algorithm. Hence, less processing time is required. The number of substructure instances retained for 7% of support value will be lesser than the number of instances retained for support value of 1% in each expansion iteration. Hence, the processing time for each steps DB-FSG like substructure expansion, pseudo duplicate elimination, canonical ordering and substructure counting and pruning will require lesser time for user defined support value of 7% than for 1%.

6 Conclusions

In this paper, for the first time, we have applied relational database approach for frequent subgraph mining. The graph representation used in this paper can represent the most general form of graph including graphs with cycles and multiple edges (between two vertices). Our approach addresses all aspects of frequent subgraph mining – from candidate generation to pseudo duplicate elimination to canonical ordering – all using standard SQL. Our experimental results show that this approach is highly scalable for very large data sets whereas main memory approaches are likely to fail. Currently, we are further optimizing the efficiency of the algorithm.

References

1. Chakravarthy, S., Beera, R., Balachandran, R.: Database approach to graph mining. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 341–350. Springer, Heidelberg (2004)
2. Cook, D.J., Holder, L.B.: Graph-based data mining. *IEEE Intelligent Systems* 15(2), 32–41 (2000)
3. Inokuchi, A., Washio, T., Motoda, H.: Complete mining of frequent patterns from graphs: Mining graph data. *Mach. Learn.* 50(3) (2003)
4. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM 2001: Proc. of the 2001 IEEE International Conference on Data Mining, Washington, DC, USA, pp. 313–320. IEEE Computer Society, Los Alamitos (2001)
5. Mishra, P., Chakravarthy, S.: Performance evaluation and analysis of k-way join variants for association rule mining. In: James, A., Younas, M., Lings, B. (eds.) BNCOD 2003. LNCS, vol. 2712, pp. 95–114. Springer, Heidelberg (2003)
6. Padmanabhan, S.: HDB-Subdue, a relational database approach to graph mining and hierarchical reduction. Master's thesis, CSE Dept., U T Arlington (2004)
7. Pradhan, S.: A Relational Database Approach to Frequent Subgraph (FSG) Mining. Master's thesis, The University of Texas at Arlington (August 2006), <http://itlab.uta.edu/ITLABWEB/Students/sharma/theses/Pra06MS.pdf>
8. Rissanen, J.: Stochastic Complexity in Statistical Inquiry Theory. World Scientific Publishing Co., Singapore (1989)
9. Sarawagi, S., Thomas, S., Agrawal, R.: Integrating mining with relational database systems: Alternatives and implications. In: SIGMOD Conference, pp. 343–354 (1998)
10. Washio, T., Motoda, H.: State of the art of graph-based data mining. *SIGKDD Explor. Newsl.* 5(1), 59–68 (2003)
11. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM 2002: Proc. of the 2002 IEEE Int. Conf. on Data Mining, pp. 721–731 (2002)

Efficient Bounds in Finding Aggregate Nearest Neighbors

Sansarkhuu Namnandorj, Hanxiong Chen, Kazutaka Furuse, and Nobuo Ohbo

Dept.Computer Science, University of Tsukuba, Tennoudai 1-1-1, Tsukuba, Ibaraki 305, Japan
{sansar, chx, furuse, ohbo}@dblab.is.tsukuba.ac.jp

Abstract. Developed from Nearest Neighbor (NN) queries, Aggregate Nearest Neighbor (ANN) queries return the object that minimizes an aggregate distance function with respect to a set of query points. Because of the multiple query points, ANN queries are much more complex than NN queries. For optimizing the query processing and improving the query efficiency, many ANN queries algorithms utilizes pruning strategies, with or without an index structure. Obviously, the pruning effect highly depends on the tightness of the bound estimation. In this paper, we figure out a property in vector space and develop some efficient bound estimations for two most popular types of ANN queries. Based on these bounds, we design the indexed and non-index ANN algorithms, and conduct experimental studies. Our algorithms show good performance, especially for high dimensional queries, for both real dataset and synthetic datasets.

1 Introduction

ANN Queries are novel forms of NN queries. Given two sets of points in spatial database P and Q , $P = \{p_1, p_2, \dots, p_N\}$ is the static source dataset and $Q = \{q_1, q_2, \dots, q_n\}$ is the query points set. As defined in [1], the aggregate distance between a data point and query points set Q can be expressed by $\text{adist}(p, Q) = f(|pq_1|, |pq_2|, \dots, |pq_n|)$, where $|pq_i|$ is the Euclidean distance between point p and q_i . Given a set P of static points, ANN Queries retrieves the data point p in P having minimal aggregate distance to Q . For ANN, different function f gives ANN different meaning. For example, if f is the *sum* function, then the ANN queries will find the point p with minimal total aggregate distance to Q . And if f is the *max* function, then the ANN query will return the point p which minimizes the maximum distance to the points in Q . ANN query now is becoming more and more important in many domains, such as clustering, outlier detection[3], GIS and mobile computing applications[8].

Now NN and ANN problems are reasonably well solved for low dimensional applications. A wide variety of solutions indexed by spatial access methods (SAM) such as the R-tree[2] and the R*-tree[9] have been proposed. Though most of them work well in low dimensional space, many studies have shown that traditional indexing methods fail in high dimensional space. According to [6], NN search could be meaningless in high dimensional spaces due to the well-known curse of dimensionality. In order to compute as few distances as possible, some pruning strategies are always used to optimize the query processing in ANN queries [1,7,8]. Some metrics are utilized to prune the search space for queries optimization. [10] uses an ellipse to approximate the extent of

all query points, and the distance or MBR derived from the ellipse is used to prune intermediate index nodes during search via the R*-tree. Obviously a tight bound enables filtering out more data which cannot belong to the answer set. Our task is to find better bounds for ANN queries.

As defined in [1], there can be many versions of ANN queries by the definition of aggregate distance, and MQM may be a general solution. However, we believe that the most useful versions are (weighted) *sum* ANN query and *maximum* ANN query. Some functions can be easily solved instead of using MQM. For example, *minimum* ANN query is answered by finding the NN point of each query point and returning the smallest one among them, in cost of $|Q|$ times that of a single NN query.

In vp-ANN [4], Luo *et al* proposed two projection-based, non-indexing pruning strategies for *sum* ANN query processing. It also assumes that all query points can fit in main memory and only consider the *sum* function. Though the efficiency for *sum* ANN queries in low dimensional data is not so good as index-based methods, it performs efficiently for high dimensional datasets. However, this method also suffers from the disadvantage that the bound becomes loose when query points are distributed hence many data points remain after the filtering phase.

Therefore, we limit to *sum* ANN and *maximum* ANN, figure out a more effective bound and develop efficient algorithms.

2 Efficient *sum* and *maximum* ANN Algorithms

In this section, we first give some denotations and the definition of the problem we are to solve. We then figure out the theoretic tight bounds, based on which our efficient algorithms are successfully proposed. Given a query object q and a non-negative radius r , an r -neighbor search (r -N search) of q is to retrieve the objects in the r -neighbor of q , $r\text{-N}(q, r) = \{p | p \in P \text{ and } d(q, p) < r\}$. We call this region by r -neighbor region. Due to space limitation, we omitted the detail implementation of r -N because it is almost straightforward with spatial index.

Sum ANN Query and Maximum ANN Query: If the aggregate distance of ANN query, f , is defined as *sum* function, we call it *sum* ANN query. *Sum* ANN query returns the data object that minimizes sum of distances to each query points. Similarly it is a *maximum* ANN query when f is *maximum* function. *Maximum* ANN query returns the data object that minimizes the maximum distance to query points.

The Theoretic Bounds

For an arbitrary point set Q in a vector space, the following property holds.

Lemma 1. *Given a point set $Q = \{q_1, q_2, \dots, q_n\} \in \mathbb{R}^d$, and let G be the geometric centroid point of Q . For an arbitrary point $X \in \mathbb{R}^d$, if $r \leq |GX|$, then $r \cdot n \leq \text{sum}(X, Q)$.*

Proof. By the property of vector space, for each point q_i in Q : $\overrightarrow{GX} = \overrightarrow{Gq_i} + \overrightarrow{q_iX}$. Summing up i , we have $n \cdot \overrightarrow{GX} = \sum_{i=1}^n \overrightarrow{Gq_i} + \sum_{i=1}^n \overrightarrow{q_iX} = \sum_{i=1}^n \overrightarrow{q_iX}$

The last equation holds because $\sum_{i=1}^n \overrightarrow{Gq_i} = 0$ due to the fact that G is the geometric centroid of Q . Considering the assumption $r \leq |GX|$, finally we have

$$n \cdot r \leq n \cdot |GX| = \left| \sum_{i=1}^n \overrightarrow{q_i X} \right| \leq \sum_{i=1}^n |\overrightarrow{q_i X}| = \text{sum}(X, Q) \quad (1)$$

Lemma 2. Given a point set $Q = \{q_1, q_2, \dots, q_n\} \in \mathbb{R}^d$, and an arbitrary point $X \in \mathbb{R}^d$. If there exists any $q_i \in Q$ which satisfies $0 \leq r \leq |Xq_i|$, then $r \leq \max(X, Q)$.

Proof.

$$r \leq |Xq_i| \leq \max(|Xq_1|, |Xq_2|, \dots, |Xq_n|) = \max(X, Q) \quad (2)$$

By Lemma 1, we know that given a certain point G then we can find sum ANN only from within the circle having radius r , centered at G . Similarly, by Lemma 2, we can find max ANN from the intersection of such circles having radius r , centered at q_i . Utilizing the above properties, we can develop more efficient algorithms for *sum* and *maximum* ANN queries.

r-NQC Method for Sum ANN Query

By using Lemma 1, we can prune a lot of data points from the target of *sum* ANN searching as in the algorithm shown in Figure 1.

```

Algorithm r-NQC( $Q$ : Set of the query points)
/*  $r = \infty$ ;  $bestDist = \infty$ ;  $bestANN = null$  */
1.   $G \leftarrow$  the geometric centroid of  $Q$ .
2.  for each point  $p$  in  $P$  do
3.      if  $dist(p, G) \leq r$ 
4.          if  $adist(p, Q) < bestDist$ 
5.               $bestDist = adist(p, Q)$ ;  $bestANN = p$ ;  $r = bestDist/n$ 
6.  end for
7.  return  $bestANN$ 

```

Fig. 1. Algorithm of the r-NQC method for *sum* ANN

In the algorithm r-NQC, any data point fails to satisfy Lemma 1 is discarded immediately (line 3). Otherwise if the aggregate distance of the point is better than best ANN found so far, then the best ANN and the circle radius r is updated (line 5). In other words, the condition of Lemma 1 is dynamically updated, which refines the bound.

Indexed r-NQC Method for Sum ANN Query

There is another idea that combines Lemma 1 with index structure to search a highly desired area and prune the other search space. To make the area as smaller as possible, it is better to search around a point which has a high probability to be the best ANN initially. By experience, the nearest neighbor point of the geometric centroid of Q in dataset (say, $vp \in P$) can be a good choice. Let its average distance to each point of Q be r , then by Lemma 1, only those points nearer than r can be answer. The best one in the remaining set is the answer of ANN. If no one satisfies the condition, then vp will be

Algorithm indexed r-NQC(Q : Set of the query points)

1. $G \leftarrow$ the geometric centroid of Q
2. $bestANN = NN(G)$ // the NN of G as current best candidate
3. $bestDist = adist(bestANN, Q)$; $r = bestDist/n$
4. $R \leftarrow r-N(G, r)$ // r-N search
5. for each point p in set R do
6. if $adist(p, Q) < bestDist$
7. $bestDist = adist(p, Q)$; $bestANN = p$
8. return $bestANN$

Fig. 2. Algorithm of the indexed r-NQC method for *sum* ANN

the best ANN. The following Figure 2 gives the detail of the algorithm. In the algorithm indexed r-NQC, NN query and r-N query is called once. If the dataset is indexed, then many efficient methods proposed for NN queries can be used. By using the idea of the Branch-and-Bound method ([5]), we can perform both NN and r-N queries efficiently. If the dataset is indexed (for example, by R-tree), in the algorithm indexed r-NQC, we can use the r-N method (line 2) and Branch-and-Bound method (line 4).

r-NQP Method for *Maximum* ANN Query

By Lemma 2, we want to give a proper r and figure out the intersection region, R . However, checking whether a point is included in R costs the same as calculating its aggregate distance. One way to reduce the cost for calculate region R is to take the region R as an intersection of only two regions which are as far away from each other as possible. Figure 3 shows the algorithm mentioned above. In algorithm r-NQP, the region R is taken as the intersection of $C(q_1, r)$ and $C(q_2, r)$ where q_1 and q_2 are far away from each other (line 1 and 2), and r is dynamically updated while checking data points.

Indexed r-NQP for *Maximum* ANN Query

To utilize index structures in r-NQP, similarly to indexed r-NQC, we need to find a vp corresponding to Lemma 2. This time, the data point nearest to the center of the MBR

Algorithm r-NQP(Q : query point set)

/ $r = \infty$; $bestDist = \infty$; $bestANN = null$; q_0 is a random element of Q */*

1. point $q_1 \leftarrow$ the farthest point from q_0 in Q .
2. point $q_2 \leftarrow$ the farthest point from q_1 in Q .
3. for each point p in P do
4. if $dist(p, q_1) < r$ and $dist(p, q_2) < r$
5. if $adist(p, Q) < bestDist$
6. $bestANN = p$; $bestDist = adist(p, Q)$; $r = bestDist$
7. end for
8. return $bestANN$

Fig. 3. Algorithm of the r-NQP method for *max* ANN

Algorithm indexed r-NQP(Q : query point set)

1. calculate point G as the center of MBR of Q
2. $bestANN = NN(G)$ // nearest neighbor of G
3. $bestDist = adist(bestANN, Q)$; $r = bestDist$
4. point $q_1 \leftarrow$ the farthest point from $bestANN$ in Q .
5. point $q_2 \leftarrow$ the farthest point from q_1 in Q .
6. set $C_1 = r-N(q_1, r)$; and $C_2 = r-N(q_2, r)$;
7. set $R = C_1 \cap C_2$ // intersection of C_1 and C_2
8. for each point p in set R do
9. if $adist(p, Q) < bestDist$
10. $bestDist = adist(p, Q)$; $bestANN = p$
11. end for
12. return $bestANN$

Fig. 4. Algorithm of the indexed r-NQP method for *max* ANN

of Q is a good choice. This is because, if the aggregate distance is defined as *maximum* function, then the aggregate centroid of Q will be the center of MBC¹ of Q . Calculating the center of MBC costs order of n^2 . But calculating the center of MBR costs only order of n , and it is near from the center of MBC in many cases.

Now it is ready to utilize R-tree structure to build the r-NQC method as shown in Figure 4.

3 Experiments

All the experiments are run with Intel Pentium(R) 4 CPU 2.33GHz PC, 1GB main memory. The query set is memory-resident, and the dataset is indexed by R-tree for MBM, indexed r-NQC and indexed r-NQP. We used workloads of 40 queries. The query points were generated randomly in the workspace of P , and distributed uniformly in a MBR of Q . We compare the CPU cost of our methods against MBM and Projection-Based method, studying the following cases of variation. That is, area of MBR M of Q , query set size n , and number of dimension of both synthetic and real dataset P . The synthetic high dimension datasets have 100,000 uniform points generated randomly in $[0, 1]$. The real dataset (68,040 points, 9-dimensional) is the image features extracted from the Corel image collection available at <http://kdd.ics.uci.edu/>.

First, we study the effect of query size n , fixing the ratio of query range over the data range, M , to 0.08 in 2-dimensional dataset. In Figure 5(a), the CPU costs of MBM, r-NQC and indexed r-NQC increase similarly as well, but the CPU cost of Projection-Based increases rapidly with increasing of n . This is because the candidate region of Projection-Based includes many data points, and has to calculate so many aggregate distances. Calculating aggregate distance costs order of n . For other methods, their candidate points number are affected by n slightly, that is, the influence of n is smaller than Projection-Based. As in Figure 5(b), the CPU costs of r-NQP and indexed r-NQP are almost constant. Because while the M is fix, the largest distance between query points

¹ Minimum Bounding Circle: the smallest circle covering Q .

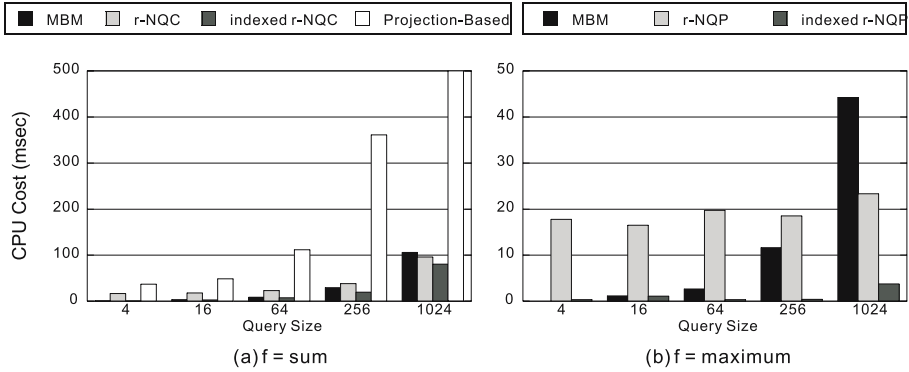


Fig. 5. CPU Cost vs Query Size (Dimension=2, MBR of Query=0.08)

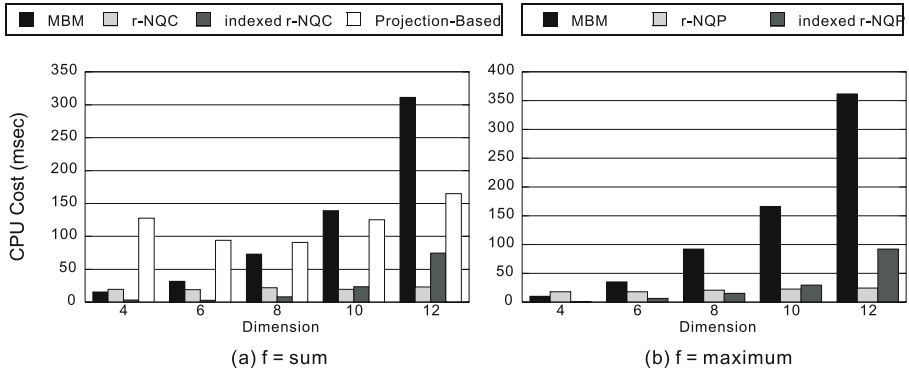


Fig. 6. CPU Cost vs Dimension (Query size=64, MBR of Query=0.16)

is constant, therefore, area of candidate region ($C(q_1, r) \cap C(q_2, r)$) will be constant, too.

In order to measure the effect of dimensionality of dataset, we set $n = 64$ and $M = 0.16$ as shown in Figure 6. Figure 6(a) shows that the CPU cost of r-NQC is constantly low with increasing the number of dimensions. If the number of dimensions is small, indexed r-NQC is fastest, but for high dimensional dataset indexed r-NQC gets slowly due to the NN search and r-N search on R-tree. It shows that using R-tree for NN or r-N searching on high-dimensional dataset is meaningless. But indexed r-NQC improves the problem with high-dimensional datasets as an index-based method. In Figure 6(b), there are same result as Figure 6(a). Non-index method r-NQP is constantly fast for any dimensional dataset, and index-based methods MBM and indexed r-NQP increase their CPU costs with increasing of the number of dimension, especially MBM increases more sharply.

The experimental result on real dataset is shown in Figure 7 for *sum* and Figure 8 for *max*. As shown in the figures, our methods outperforms MBM method in almost all

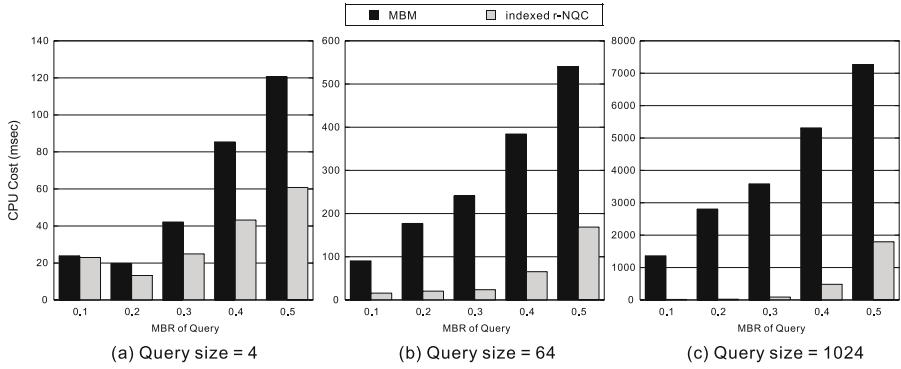


Fig. 7. CPU Cost vs MBR of Query for *sum*. Query sizes n are (a):4, (b):64, and (c):1024

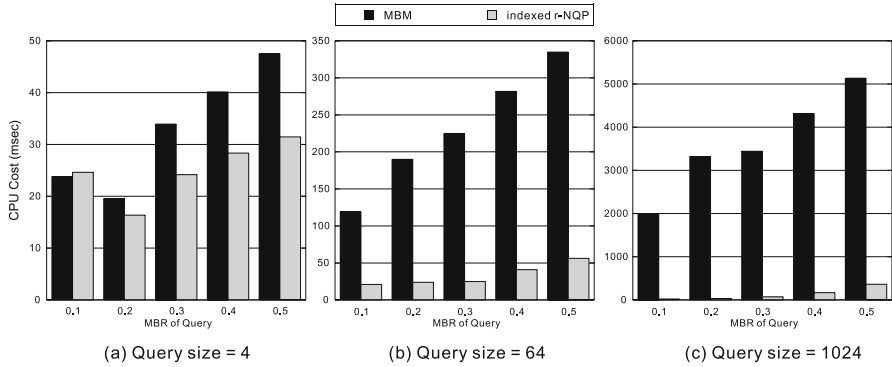


Fig. 8. CPU Cost vs MBR of Query for *max*. Query sizes n are (a):4, (b):64, and (c):1024

cases. When the query size n is small, their CPU costs are similarly low, but when n increases, the CPU cost of MBM increases much rapidly with increasing of M . In contrast, our methods, especially indexed r-NQC for *maximum* ANN queries only increases slightly when n becomes larger.

4 Conclusion

ANN Queries are the extension of NN queries but are more complex because of multiple query points. Based on the coordinate space, we consider the most popular cases of *sum* function and *maximum* function in ANN queries. Assuming that Q is memory-resident we described and analyzed r-NQC method for *sum* ANN queries and r-NQC method for *maximum* ANN queries, and compared them with index-based method MBM and non-index method Projection-Based. Our strategy finds the proper search region and enables efficient pruning of irrelevant ones. The experimental results demonstrate that the methods we proposed perform as good as the index-based method MBM in low

dimensional datasets. In high dimensional space, our methods perform much better than other methods. Our non-index methods improved the problems of non-index methods which have been proposed so far. Moreover, we improved the problems of index-based methods by utilizing the spatial index structure R-tree. In the future, we intend to explore the case of other ANN functions to give a general solution. Also, theoretical analysis on the pruning effect of the new bound against various distributions and dimensionality of data, is under going.

References

1. Papadias, D., Tao, I., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial datasets. *ACM Trans. Database Syst.* 30(2), 529–576 (2005)
2. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: *SIGMOD 1984*, pp. 47–57 (1984)
3. Aggarwal, C.C., Yu, P.S.: Outlier Detection for High Dimensional Data. In: *SIGMOD 2001*, pp. 37–46 (2001)
4. Luo, Y., Chen, H., Furuse, K., Ohbo, N.: Efficient Methods in Finding Aggregate Nearest Neighbor by Projection-Based Filtering. In: Gervasi, O., Gavrilova, M.L. (eds.) *ICCSA 2007, Part III. LNCS*, vol. 4707, pp. 821–833. Springer, Heidelberg (2007)
5. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest Neighbor Queries. In: *SIGMOD 1995*, pp. 71–79 (1995)
6. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is Nearest Neighbors Meaningful? In: *Proc. of the Int. Conf. Database Theories*, pp. 217–235 (1999)
7. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group Nearest Neighbor Queries. In: *ICDE 2004*, pp. 301–312 (2004)
8. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate Nearest Neighbor Queries in Road Networks. *IEEE TKDE* 17(6), 820–833 (2005)
9. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In: *SIGMOD 1990*, pp. 322–331 (1990)
10. Li, H., Lu, H., Huang, B., Huang, Z.: Two ellipse-based pruning methods for group nearest neighbor queries. In: *GIS*, pp. 192–199 (2005)

A Grid-Based Multi-relational Approach to Process Mining

Antonio Turi, Annalisa Appice, Michelangelo Ceci, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona, 4 - 70126 Bari - Italy
{turi,appice,ceci,malerba}@di.uniba.it

Abstract. Industrial, scientific, and commercial applications use information systems to trace the execution of a business process. Relevant events are registered in massive logs and process mining techniques are used to automatically discover knowledge that reveals the execution and organization of the process instances (cases). In this paper, we investigate the use of a multi-level relational frequent pattern discovery method as a means of process mining. In order to process such massive logs we resort to a Grid-based implementation of the knowledge discovery algorithm that distributes the computation on several nodes of a Grid platform. Experiments are performed on real event logs.

1 Introduction

Many information systems, such as Workflow Management Systems, ERP systems, Business-to-business systems and Firewall systems trace behavior of running processes by registering relevant events in massive logs. Events are described in a structured form that includes properties of cases and activities. A case represents the process instance which is being handled, while an activity represents the operation on the case. Information on timestamp and on the person executing the event (performer) is available in the logs. Both activities and performers may belong to different categories. Event logs are stored in multi-terabyte warehouses and sophisticated data mining techniques are required to process this huge amount of data and extract knowledge concerning the execution and organization of the recorded processes. This huge amount of data is the main concern of research in process mining whose aim is to discover a description or prediction of real process, control, organizational, and social structures [10].

Process mining poses several challenges to the traditional data mining tasks. In fact, data stored in event logs describe objects of different type (cases, activities and performers) which are naturally modeled as several relational data tables, one for each object type. Foreign key constraints express the relations between these objects. This (relational) data representation makes necessary distinguishing between the reference objects of analysis (cases) and other task-relevant objects (activities and performers), and to represent their interactions. Another challenge is represented by the temporal autocorrelation. Events are temporally related according to a timestamp. This means that the effect of a

property at any event may not be limited to the specific event. Furthermore, activities and performers are generally organized in hierarchies of categories (e.g. the performer of operations on a text file can be a writer or a reader). By descending or ascending through a hierarchy, it is possible to view the same object at different levels of abstraction (or granularity). Finally, reasoning is the process by which information about objects and their relations (e.g. operator of indirect successor) are used to arrive at valid conclusions regarding the object relations [7]. This source of knowledge cannot be ignored in the search.

Currently, many algorithms [2,1,11,3] have dealt with several of these challenges and some of them are integrated into the ProM framework [12]. Anyway, to the best of our knowledge, methods of process mining neither support a multi-level analysis nor use inferential mechanisms defined within a reasoning theory. Conversely, the multi-relational data mining method SPADA [5] offers a sufficiently complete solution to all the challenges posed by the process mining tasks in descriptive case. However, SPADA is not applicable in practice. Indeed, frequent pattern discovery is a very complex task, particularly in the multi-relational case [5]. In addition SPADA, similarly to most of the multi-relational data mining algorithms, operates with data in main memory, hence it is not appropriate for processing massive logs. Advantages of the multi-relational approach in facing the challenges of the process mining justify the attempt of resorting to the computational power of distributed high-performance environments (e.g., computational Grids [4]) to mitigate the complexity of the relational frequent pattern discovery on massive event logs.

In this paper, we present G-SPADA, an extension of SPADA, which discovers approximate multi-level relational frequent patterns by distributing exact computation of locally frequent multi-level relational patterns on a computational Grid and then by post-processing local patterns in order to approximate the set of the globally frequent patterns as well as their supports. Distributing relational frequent pattern discovery on a Grid poses several issues. Firstly, relational data must be divided in data subsets and each subset has to be distributed on the Grid. Split must take into account relational structure of data, that is, each data split must includes a subset of reference objects and the task-relevant objects to reconstruct all interactions between them. Secondly, it is necessary a framework for building the Grid applications utilizing the power of distributed computation and storage resources across the Internet. Finally, processing local patterns to approximate global ones requires a way of combining distinct sets of patterns into a single one and obtaining an estimate of the global support.

2 Multi-level Relational Frequent Pattern Discovery

The multi-level relational pattern discovery task is formally defined as follows: *Given*: a set S of reference objects, some sets R_k , $1 \leq k \leq m$ of task-relevant objects, a background knowledge BK which includes hierarchies H_k on the objects in R_k and domain knowledge in form of rules, a deductive database D that is formed by an extensional (D_E) part where properties and relations of reference

objects and task-relevant objects are expressed in derived ground predicates and an intensional part (D_I) where domain knowledge in BK is expressed in form of rules, M granularity levels in the descriptions (1 for the highest), a set of granularity ψ_k which associate each object in H_k with a granularity level to deal with several hierarchies at once, a threshold $minsup[l]$ for each granularity level l ($1 \leq l \leq M$), *Find*, for each granularity level l , the *frequent*¹ relational patterns which involve properties and relations of task relevant-objects at level l of H_k .

The relational formalization of the task of frequent pattern discovery is based on the idea that each unit of analysis (or example) $D[s]$ includes a reference object $s \in S$ and all the task-relevant objects of R_k which are (directly or indirectly) related to s according to some foreign key path in D . The frequency (support) of a pattern is based on the number of units of analysis, i.e., reference objects, covered by the pattern. An example of relational pattern is:

Example 1. Let D_E be the extensional database described in Example 1. A possible relational pattern P1 on D is in the form:

P1: *case(A), activity(A,B), is_a(B,activity), before(B,C), is_a(C,activity), description(C,workinprogress), user(B, D), is_a(D, performer)* [72.25%]

P1 expresses the fact that a process A is formed by two sequential activities, namely B and C , the performer of B is generic. The support is 72.5%.

By taking into account hierarchies on task-relevant objects, relational patterns can be discovered at multiple level of granularity.

Example 2. Let us consider two level hierarchies defined on performers and activities defined in the followings:

administrator, user $\rightarrow$ *performer*; *namemaker, delete, workflow* $\rightarrow$ *activity*

P2 is a finer-grained relational pattern than P1 obtained by descending one level in hierarchies. P2 is in the form:

P2: *case(A), activity(A,B), is_a(B,namemaker), before(B,C), is_a(C,workflow), description(C,workinprogress), is_a(D, administrator)* [62.5%]

P2 provides better insight than P1 on the nature of B , C and D .

In SPADA [5], multi-level relational frequent patterns are discovered according to the *levelwise* method [6] that is based on a breadth-first search in the lattice of patterns spanned by θ -subsumption [8] generality order ($\succeq_\theta$).

3 G-SPADA

Similarly to Partition [9], G-SPADA splits a dataset into several partitions to be processed independently. It approximates the multi-level relational frequent pattern discovery by means of a three stepped strategy. In the first step, the set of original N reference objects is partitioned into n approximately equally-sized subsets ($n \ll N$). Each partition includes a subset of the reference objects and the set of task-relevant objects. In the second step, the frequent pattern

¹ With support greater than $minsup[l]$.

computation is parallelized and distributed on n nodes of a Grid platform, one node for each partition. In this way, G-SPADA generates n parallel executions of SPADA at the same time and retrieves local patterns which are frequent in at least one of the data partition. In the third step, G-SPADA approximates the set of globally frequent patterns by merging patterns discovered at the nodes.

The basic idea in approximating the global patterns is that each globally frequent pattern must be locally frequent in at least k partitions of the original dataset. In the case k is set to 1, this guarantees that the union of all local solutions is a superset of the global solution. However, a merge step with $k = 1$ may generate several false positives, i.e. patterns that result locally frequent but globally infrequent. Hence, value of k should be adequately tuned between 1 and n in order to find the best trade-off between false positive and false negative frequent patterns. The merge step also attempts to approximate values of support for the global patterns starting from the local values of support.

3.1 Relational Data Partitioning

G-SPADA pre-processes the deductive database of logs and completes the description explicitly provided for each example (D_E) with the information that is implicit in the domain knowledge (D_I). An example of this saturation step is:

Example 3. Let us consider the deductive database:

case(c1). case(c2). activity(c1,a1). activity(c1,a2). activity(c1,a3). activity(c2,a4). time(a1,10). time(a2,25). time(a3,29). time(a4,13). description(a1,create). ...
before(A,B):-activity(C, A1),activity(C, A2), time(A1,T1), A1≠ A2, time(A2,T2), T1<T2, not(activity(C, A), A≠ A1, A≠ A2, time(A,T), T1<T, T<T2).

By performing the saturation step, the following predicates are made explicit in the database: *before(a1,a2). before(a2,a3).*

Saturation precedes data partitioning. In this way, redundant inferences are prevented for properties and relations of task-relevant objects shared from two or more reference objects belonging to different data partitions.

Data partitioning is performed by randomly splitting the set of reference objects in n approximately equal-sized partitions such that the union of the partitions is the entire set of reference objects. These data partitions are enriched by adding the ground predicates which describe properties and relations of the reference objects falling in the partition at hand. Subsequently, properties and relations of task-relevant objects related to reference objects according to some foreign key path are also added to the partition.

3.2 Distributing Computation on Grid

Each dataset partition is shipped along with the G-SPADA pattern discovery algorithm to computation nodes on Grid using gLite² middleware. This is done

² gLite (<http://glite.web.cern.ch/glite/>) is a next generation middleware for Grid computing which provides a framework for building Grid applications.

by submitting parametric jobs described in JDL (Job Description Language) through the CLI (command line interface). Submission of jobs on Grid are divided in several steps: (i) Authenticate on a UI (user interface) through PKI based authentication system with proxy credentials (GSI); (ii) Prepare the jobs (JDL, shell script to automate procedure, input file); (iii) Upload (Stage-in) a set of dataset; (iv) Submit a relative parametric job; (v) Check/wait results; (vi) Finally, once the job is executed on Grid, we get the output (Stage-out) files containing the frequent pattern sets along with their support for each sample.

3.3 Computing Approximate Global Frequent Patterns

The n sets of local frequent patterns are collected from the computation nodes of the Grid platform and then merged to approximate the set of global patterns.

For each local pattern discovered in at least k data partitions ($1 \leq k \leq n$), G-SPADA derives an approximate of the global support by averaging the support values collected on the partitions where the pattern is found to be frequent. The check that the same local pattern occurs in different partitions is based on an equivalence test between two patterns under θ -subsumption, which corresponds to performing a double θ -subsumption test ($P \succeq_{\theta} Q$ and $Q \succeq_{\theta} P$). Local patterns occurring in less than k partitions are filtered out. The global frequent patterns obtained following this merge procedure approximate the original frequent patterns which can be possibly mined on the entire dataset.

An example of approximate global process pattern is:

*case(A), activity(A,B), is_a(B,namemaker), before(B,C), is_a(C,workflow),
description(C,workinprogress)* [7, 72.5%]

which describes the order of execution between two activities, namely B and C , in the process A . B is a name-maker activity while C is a workflow activity. In addition, C is described as work in progress. 7 means that this pattern is found in 7 partitions (sample-level support), while 72.5% indicates the macro average support obtained by averaging the support values computed on the 7 samples.

4 Experimental Results

Experiments are performed by processing event logs provided by THINK3 Inc³ in the context of the TOCAI project⁴. THINK3 is a global player in Cad and Plm market whose mission is to help manufacturers optimizing their entire product development processes. G-SPADA is run on the deductive database that is obtained by boiling down the event logs from January 1st to February 28th, 2006 and considering as domain knowledge the definition of the “before” predicate. In the experiments, each case (process instance) traced in the logs is considered as a whole and multi-level relational patterns are discovered from traced business processes. These patterns capture the possible relation between the order of activities and the properties of their performers.

³ <http://www.think3.com/en/default.aspx>

⁴ <http://www.dis.uniroma1.it/~tocai/index.php>

activity [59578]	performer [73]
+ -- workflow [52929]	+ -- administrator [3]
+ -- o1,o2,o4,o5,o7,...	+ -- mueller,cma,admin
+ -- namemaker [6689]	+ -- user [70]
+ -- o0,o3,o6,o9,o11,o14,...	+ -- altendorfer,amaeder,andra,...
+ -- deleteEnt [230]	
+ -- o392,o439,o476, ...	
+ -- prpDelete [122]	
+ -- o11828,o11829,o11830,...	
+ -- prpModify [1]	
+ -- o54318	
+ -- cast [7]	
+ -- o1609,o1672,o1673,o8299,o8300,...	

Fig. 1. Three-level hierarchies on activity and performer

4.1 Data Description

Data trace the behavior of 21,256 instances of a business process recorded in the period under analysis. A case is traced by registering its events. Each event describes the activity executed within a case and the activity performer. The activities correspond to six tasks (or classes of operations), that is, workflow, namemaker, deleteEnt, prpDelete, prpModify and cast while the performers are the category of person (or system) executing the activity, that is, user or administrator. This corresponds to model activities and performers by means of three-level hierarchies (see Figure 1). Each hierarchy is mapped into the three granularity levels thus allowing to deal uniformly with both hierarchies at once. By descending or ascending through the hierarchy, it is possible to view the same activity or performer at different levels of granularity.

For each activity, a text description of the operation is registered in the event logs. This text includes a left part and a right one (“left::right”). The right part is a characterization of the description of the operation provided in the left part. Some examples of descriptions registered in the event logs are: *create::workingprogress*, *t2f::freigabe*, *wip2f::freigabe*.

Thirty-six distinct descriptions are registered in the logs, but several of them share the same left or right part: *k2f::freigabe*, *m2f::freigabe*, *document::doccad*, *document::doccad3d*. By interpreting this structure, the activity descriptions is practically boiled down into two predicates, that is:

leftDescription(activity, text). *rightDescription(activity, text)*.

Finally, each performer is described by the belonging group. Twenty two distinctive groups are registered in the event logs. Performers labeled as users and administrators can possibly belong to the same group.

4.2 Local and Global Multi-level Relational Patterns Discovery

G-SPADA is run on the event logs including 395,404 ground predicates. Reference objects are the cases, while task-relevant objects are the activities and performers. In this way, the description of an activity and of the group of its performer is not limited to the specific event.

Table 1. Number of global frequent patterns discovered by varying k in $[1,20]$

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
#P	428	424	412	412	400	372	372	364	356	356	344	344	314	312	312	311	263	259	259	235

Data of the event logs are split into twenty partitions. The discovery of the local multi-level frequent relational patterns is then distributed on twenty nodes. The Grid infrastructure is required to process the huge amount of data registered in the logs. Indeed, SPADA generates a memory exception when running on the entire dataset. Multi-level relational patterns are discovered at each node with $minsup[l] = 0.2$ ($l = 1, 2$) and $max\_len\_path = 9$ ⁵. Finally, for each level of granularity, global patterns are approximated from the local ones by varying k between 1 and 20. The number of discovered global patterns are reported in Table 1. Obviously, the number of global patterns decreases by increasing k .

Global patterns provide a compact description of the instances of process traced in the logs. They provide a multi-level insight of the order of execution and/or organization of the processes traced in the logs.

At level 1, G-SPADA discovers the global relational pattern P1:

P1: $case(A), activity(A, B), before(B, C), user(B, D), is\_a(B, activity), is\_a(C, activity), is\_a(D, performer), descript(C, release)$. [$k=20, avgSup=63.55\%$]

P1 captures the execution order between two activities (B and C) within a case (A). One activity (B) is performed by a generic performer D , while the other activity (C) is described as a *release* activity. By descending one level of the hierarchies, G-SPADA discovers the finer grained global relational pattern P2:

P2: $case(A), activity(A, B), before(B, C), user(B, D), is\_a(B, workflow), is\_a(C, workflow), is\_a(D, user), descript(C, release)$. [$k=20, avgSup=51.43\%$]

P2 clarifies that the performer C is a user, while B and D are workflow activities. Support of P2 is reconstructed from the support of P2 on the local partitions, that is, P2 covers at least 10935 cases registered in the original event logs.

The pattern P3:

P3: $case(A), activity(A, B), before(B, C), user(B, D), is\_a(C, workflow), is\_a(D, user), descript(B, construction), descript(C, release)$. [$k=4, avgSup=29.06\%$]

is a specialization of P2 under θ -substitution which describes B as a *construction* activity. Obviously, the support of P3 decreases with respect to the support of P2, due to the θ -substitution antimonotonicity of support.

Finally, the relational pattern:

P4: $case(A), activity(A, B), before(B, C), before(C, D), is\_a(B, namemaker), is\_a(C, workflow), is\_a(D, workflow), descleft(C, creation), descleft(D, wip2k)$ [$k=16, avgSup=21.35\%$]

describes the execution order among three sequential activities, namely B , C and D . B is a *namemaker* activity, while C and D are *workflow* activities. C is described as a *creation* activity, while D is described as *wip2k* operation.

⁵ max_len_path is the maximum number of predicates to be included in a pattern.

5 Conclusions

In this paper, we present G-SPADA, an extension of the system SPADA, to discover approximate multi-lever relational frequent patterns in the context of process mining. G-SPADA exploits a multi-relational approach in order to deal with both multiple nature of data stored in event logs and temporal autocorrelation. G-SPADA faces the need of processing massive logs by resorting to a grid based architecture. Experiments on the real event logs allow us to discover interpretable patterns which capture regularities in the execution of activities and the characteristics of the performers of a business process. Such patterns can be used to deploy new systems supporting the execution of business processes or analyzing and improving already enacted business processes.

Acknowledgments

This work is in partial fulfillment of the research objectives of “TOCAL.it” project “Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet”. The authors wish to thank THINK3 Inc. for providing data.

References

1. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
2. Cook, J.E., Wolf, A.L.: Software process validation: Quantitatively measuring the correspondence of a process to a model. *ACM Trans. Softw. Eng. Methodol.* 8(2), 147–176 (1999)
3. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. *IEEE Trans. Knowl. Data Eng.* 18(8), 1010–1027 (2006)
4. Li, T., Bollinger, T.: Distributed and parallel data mining on the grid. In: ARCS Workshops. LNI, vol. 41, pp. 370–379. GI (2004)
5. Lisi, F.A., Malerba, D.: Inducing multi-level association rules from multiple relations. *Machine Learning* 55(2), 175–210 (2004)
6. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* 1(3), 241–258 (1997)
7. Michalski, R.S.: A theory and methodology of inductive learning, pp. 323–348 (1993)
8. Plotkin, G.D.: A note on inductive generalization 5, 153–163 (1970)
9. Savasere, A., Omiecinski, E., Navathe, S.B.: An efficient algorithm for mining association rules in large databases. In: VLDB, pp. 432–444 (1995)
10. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., de Medeiros, A.K.A., Song, M., Verbeek, H.M.W.: Business process mining: An industrial application. *Inf. Syst.* 32(5), 713–732 (2007)

11. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.* 47(2), 237–267 (2003)
12. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)

Extraction of Opposite Sentiments in Classified Free Format Text Reviews

Dong (Haoyuan) Li¹, Anne Laurent², Mathieu Roche², and Pascal Poncelet¹

¹ LGI2P - École des Mines d'Alès, Parc Scientifique G. Besse, 30035 Nîmes, France
{Haoyuan.Li,Pascal.Poncelet}@ema.fr

² LIRMM - Université Montpellier II, 161 rue Ada, 34392 Montpellier, France
{laurent,mroche}@lirmm.fr

Abstract. Most of the previous approaches in opinion mining focus on the classifications of opinion polarities, *positive* or *negative*, expressed in customer reviews. In this paper, we present the problem of extracting contextual opposite sentiments in classified free format text reviews. We adapt the sequence data model to text mining with Part-of-Speech tags, and then we propose a belief-driven approach for extracting contextual opposite sentiments as unexpected sequences with respect to the opinion polarity of reviews. We conclude by detailing our experimental results on free format text movie review data.

1 Introduction

Opinion mining received much attention in finding personal opinions from user generated contents, such as customer reviews, forums, discussion groups, and blogs, where most of the previous approaches concentrate on the classifications of opinion polarities, *positive* or *negative*, in free format text reviews [9,14,2,16,5,8,15]. Although the positive-negative classifications are determinative, the opposite sentiments expressed in classified reviews, within the context of topic, become more and more interesting for decision making.

For instance, about a notebook computer, a positive review may contain the sentences like “however the graphics performance is not enough”, or in a negative review we may also find “anyway this notebook is beautiful”, and such critiques are important to improve the quality of products. However, even sentence-level sentiment classifications [2,5,15] extract the sentences that express the opposite sentiment with the positive-negative connotations different to document-level opinion polarity, such sentences may be not within the same context of the topic about the review.

In this paper, we present a belief-driven approach for extracting contextual opposite sentiments in classified free format text reviews. A training-extracting process is considered: Given a topic context, first a sequential pattern mining algorithm is applied to a set of classified training reviews, in order to generate the contextual models of opinion polarity with respect to current topic. Then, from such contextual models, a belief base is constructed to represent the opinion

polarity by using a dictionary of antonyms¹ of the adjectives contained in the contextual models. Finally, the unexpected sequence mining process proposed in our previous work [7] is performed to the target reviews for extracting all sentences that contradict the belief base, which stand for the contextual opposite sentiments.

The rest of this paper is structured as follows. Section 2 introduces the related work. In Sect. 3 we first formalize the data model, then propose the contextual models and belief base on sentiment polarities, and then present the process of extracting contextual opposite sentiments. Section 4 details our experiments on positive-negative movie-review data. Section 5 is a short conclusion.

2 Related Work

Opinion mining in free format text contents is closely connected with the Natural Language Processing (NLP) problems, where the positive or negative connotation can be annotated by the subjective terms at document-level [9,14,2,8] or sentence-level [2,16,5,15].

In [2], a term frequencies based scoring system is proposed for determining both document- and sentence-level sentiment polarities. The approach proposed in [5] extracts the features of products contained in customer reviews with positive-negative polarities, which can be considered as a sentence-level opinion classification. Compared to our approach, [16] proposed a model for classifying opinion sentences as positive or negative in terms of the main perspective expressed in the opinion of document, which identifies facts and opinions, and can be considered as a contextual approach. Another contextual notion, so called contextual polarity, is proposed in [15], which is determined by the dependency tree of the structure of sentences; in our approach, we use sequential pattern mining to determine the frequent structures of contextual models for sentiment polarity.

Actually, the opinion polarities are often given by the adjectives [3,13]. We use WordNet [4] for determining the antonyms of adjectives required for constructing the belief base, which has been used in many NLP and opinion mining approaches. For instance, in the proposal of [6], WordNet is also applied for detecting the semantic orientation of adjectives.

3 Extracting Contextual Opposite Sentiments

3.1 Data Model

We are given a set of free format text reviews that have been already classified into positive-negative opinion polarities. Each review consists in an ordered list of sentences, and each sentence consists in an ordered list of words. In order to

¹ The antonym dictionary is based on the WordNet project, which can be found at <http://wordnet.princeton.edu/>

involve the words in the context of reviews, the Part-of-Speech tag (PoS tag) introduced in the TreeTagger approach [11] is considered, and a list of such PoS tags is available in [10]. With respect to this list, we do not consider the difference between the different tags of the adjectives (J instead of JJ, JJR and JJS), of the adverbs (R in stead of R, RB, RBR and RBS), of the nouns (N in stead of N, NN, NNS, NP and NPS), and of the verbs (V in stead of V, VB, VBD, VBG, VBN, VBP and VBZ).

A *word*, denoted as w , is a lemma associated with a simplified PoS tag. For example $(be|V)$ is a word where be is a lemma and V is the base tag standing for the verbs. Without loss of generality, we use the wild-card $*$ and a simplified PoS tag for denoting a generalized vocabulary. For example, $(*|V)$ denotes a vocabulary that is a verb. Especially, we use (NEG) for denoting the adverb $(not|R)$, $(n't|R)$, and other negation expressions, so that by default when we say the term *word*, we do not include (NEG) .

Let $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$ be a set of a limited number of distinct words, a *clause*, denoted as s , is an ordered list of words $w_1 w_2 \dots w_k$. The *length* of a clause is the number of words contained in the clause, denoted as $|s|$. For example, $(film|N)(be|V)(good|J)$ is a clause with length 3, in the order $(film|N)$ followed by $(be|V)$ and then followed by $(good|J)$. A word could also be a clause with length 1 if it is reduced to one lemma and its associated PoS tag. An *empty clause* is denoted as $\emptyset$, we have $s = \emptyset \iff |s| = 0$. The *concatenation* of clauses is denoted as the form $s_1 \cdot s_2$.

Within the context of mining sequence patterns [1], a word is an *item* and a clause is a *sequence*. Given two clauses $s = w_1 w_2 \dots w_m$ and $s' = w'_1 w'_2 \dots w'_n$, if there exist integers $1 \leq i_1 < i_2 < \dots < i_m \leq n$ such that $w_i = w'_{j_i}$ for all w_i , then s is a *sub-clause* of s' , denoted as $s \sqsubseteq s'$. If we have $s \sqsubseteq s'$, we say that s is *contained in* s' , or s' *supports* s . If clause s is not contained in any other clauses, then we say that the clause s is *maximal*. For example, the clause $(film|N)(good|J)$ is contained in the clause $(film|N)(be|V)(good|J)$, but is not contained in the clause $(be|V)(good|J)(film|N)$.

A *sentence*, denoted as S , is a maximal clause that is terminated by one of the following symbols “: ; . ? !” in the given text reviews. A *document*, denoted as $\mathcal{D}$, is an ordered list of sentences. Given a document $\mathcal{D}$, the *support* or *frequency* of a clause s , denoted as $\sigma(s, \mathcal{D})$, is the total number of sentences $S \in \mathcal{D}$ that support s . Given a user specified threshold of support called *minimum support*, denoted as min_supp , a clause is *frequent* if $\sigma(s, \mathcal{D}) \geq min\_supp$.

3.2 Contextual Models of Sentiment Polarity

We represent sentiment polarities as rule-format on clauses, that is, $s_\alpha \Rightarrow s_\beta$, where s_α and s_β are two clauses; given a clause s , if we have $s_\alpha \cdot s_\beta \sqsubseteq s$, then we say that the clause s *supports* the rule r , denoted as $s \models r$. We therefore propose a belief system for formalizing the opposite sentiments expressed in classified reviews. A *belief* on clauses, denoted as b , consists of a rule $s_\alpha \Rightarrow s_\beta$ and a semantical constraint $s_\beta \not\sim s_\gamma$, where the clause s_γ is semantically contradicts the clause s_β . We note a belief as $b = [s_\alpha; s_\beta; s_\gamma]$. A belief constrains that if the

clause s_α occurs in a clause s , i.e., $s_\alpha \sqsubseteq s$, then the clause s_β should occur in s after s_β , and the clause s_γ should not occur in s after s_α , that is,

$$[s_\alpha; s_\beta; s_\gamma] \iff s_\alpha \sqsubseteq s \implies s_\alpha \cdot s_\beta \sqsubseteq s \wedge s_\alpha \cdot s_\gamma \not\sqsubseteq s.$$

A clause s that verifies a belief b is *expected*, denoted as $s \models b$; that violates a belief b is *unexpected*, denoted as $s \not\models b$. Given a belief $b = [s_\alpha; s_\beta; s_\gamma]$ and a clause s such that $s_\alpha \sqsubseteq s$, the unexpectedness is considered as:

$$s_\alpha \cdot s_\beta \not\sqsubseteq s \wedge s_\alpha \cdot s_\gamma \sqsubseteq s \implies s \not\models b.$$

Example 1. Given a belief $b = [(be|V); (good|J); (bad|J)]$ and two clauses $s_1 = (be|V)(a|DT)(good|J)(film|N)$, $s_2 = (be|V)(bad|J)(actor|N)$, we have $s_1 \models b$ and $s_2 \not\models b$. $\square$

Let M^+ be the positive sentiment and M^- be the negative sentiment, a sentiment $M \in \{M^+, M^-\}$ can be expressed in documents (denoted as $\mathcal{D} \models M$), sentences (denoted as $S \models M$), clauses (denoted as $s \models M$) or vocabularies (denoted as $v \models M$). In addition, we denote the negation of a sentiment M as $\overline{M}$, so that we have $\overline{M^+} = M^-$ and $\overline{M^-} = M^+$. The negation is taken into account in other text-mining applications (for instance for synonym/antonym extraction process [13]).

Proposition 1. *Given a sentiment $M \in \{M^+, M^-\}$, if a document $\mathcal{D} \models M$, then there exists at least one sentence $S \in \mathcal{D}$ such that $S \models M$; if a sentence $S \models M$, then there exists at least one word $w \sqsubseteq S$ such that $w \models M$ or at least one clause $(NEG)v \sqsubseteq S$ (or $w(NEG) \sqsubseteq S$) such that $w \models \overline{M}$.*

We focus on the sentiments expressed by the sentences that contain adjectives and nouns/verbs, such as “this is a good film”. The sentiment expressed by sentences like “this film is well produced” is currently not considered in our approach. Note that we extract basic words relations without the use of syntactic

Contextual Model	Sentiment Rule	Belief Pattern
J-N model	$(* J) \Rightarrow (* N)$	$[(\overline{*} J); \emptyset; (* N)]$ $[(NEG)(* J); \emptyset; (* N)]$
N-J model	$(* N) \Rightarrow (* J)$	$[(*) N]; (* J); (\overline{*} J)]$ $[(*) N]; (* J); (NEG)(* J)]$
V-J model	$(* V) \Rightarrow (* J)$	$[(*) V]; (* J); (\overline{*} J)]$ $[(*) V]; (* J); (NEG)(* J)]$ $[(*) V](NEG); (\overline{*} J); (* J)]$
J-V model	$(* J) \Rightarrow (* V)$	$[(*) J]; (* V); (* V)(NEG)]$
NEG-J-N model	$(NEG)(* J) \Rightarrow (* N)$	$[(NEG)(\overline{*} J); \emptyset; (* N)]$
N-NEG-J model	$(*) N)(NEG) \Rightarrow (* J)$	$[(*) N](NEG); (* J); (\overline{*} J)]$
V-NEG-J model	$(*) V)(NEG) \Rightarrow (* J)$	$[(*) V](NEG); (* J); (\overline{*} J)]$
J-V-NEG model	$(*) J) \Rightarrow (* V)(NEG)$	$[(\overline{*} J); \emptyset; (* V)(NEG)]$

Fig. 1. Contextual models of sentiment polarity

analysis tools [12] to avoid the silence in the data (i.e. syntactic relations not extracted by the natural language systems).

With the adoption of rules and beliefs, we can extract the contextual information from reviews by finding the most frequent clauses that consist of at adjectives and nouns/verbs by sequential pattern mining algorithms, where the frequent nouns and verbs reflect topic of reviews, and the sentence-level sentiment polarities are expressed by frequent adjectives.

We propose a set of contextual models for constructing the belief base of opinion polarities within the context of review topic, listed in Fig. 1, where the word $(\bar{*}|J)$ stands for each antonym of the word $(*|J)$. Given a review, each sentence violating a belief generated from one of the belief patterns listed in Fig. 1 stands for an opposite sentiment.

3.3 Extracting Contextual Opposite Sentiments

We now introduce the training-extracting process of our approach. Let $\mathcal{V}$ be a set of adjectives expressing the sentiment M , we denote $\bar{\mathcal{V}}$ the set that contains the antonym(s) of each word contained in $\mathcal{V}$. Thus, for each $(*|J) \in \mathcal{V}$, we have $(*|J) \models M$ and $(\bar{*}|J) \in \bar{\mathcal{V}}$. Given a *training document* $\mathcal{D}_L$ such that for each sentence $S \in \mathcal{D}_L$, there exist at least one adjective $(*|J) \in \mathcal{V}$ or there exist (NEG) and at least one adjective $(*|J) \in \bar{\mathcal{V}}$. In order to construct the belief base of contextual models, we first apply a sequential pattern mining algorithm for discovering all maximal frequent clauses from $\mathcal{D}_L$ with respect to a minimum support threshold, denoted as $\mathcal{D}_F$. For each clause $s \in \mathcal{D}_F$, if s verifies a contextual model listed in Fig. 1 with the listing-order, then a set of beliefs can be generated from s corresponding to the belief pattern(s) of each contextual model. A belief base $\mathcal{B}_M$ can therefore be constructed with respect to the topic of reviews.

Example 2. Given a clause $s = (this|DT)(be|V)(a|DT)(good|J)(film|N)$, we have that s supports the J-N and V-J models, and the sentiment rules are $(good|J) \Rightarrow (film|N)$ and $(be|V) \Rightarrow (good|J)$. We have the priority of J-N model is higher than V-J model, so that $(good|J) \Rightarrow (film|N)$ is used for generating beliefs. Let $(bad|J)$ be the antonym of $(good|J)$, we have two beliefs generated: $[(bad|J); \emptyset; (film|N)]$ and $[(NEG)(good|J); \emptyset; (film|N)]$. $\square$

Given a classified review $\mathcal{D}_M$ and a belief base $\mathcal{B}_M$ corresponding to the sentiment polarity M , the procedure of extracting unexpected sentences can be briefly described as follows. For each sentence $S \in \mathcal{D}_M$ and for each belief $b \in \mathcal{B}_M$ such that $b = [s_\alpha; s_\beta; s_\gamma]$, s_α is first matched for improving the performance; if $s_\alpha \sqsubseteq S$, and then if $s_\alpha \cdot s_\beta \not\sqsubseteq S$ and $s_\alpha \cdot s_\gamma \sqsubseteq S$, then S is an unexpected sentence expressing the contextual opposite sentiment $\bar{M}$. A detailed description of the representation of belief base and the unexpected sequence mining process can be found in [7].

4 Experiments

The data sets we use for evaluating our approach are the movie-review data² introduced in [8]. We combined these reviews into two documents $\mathcal{D}^+$ (containing 1,000 positive reviews, 75,740 sentences, and 21,156 distinct words) and $\mathcal{D}^-$ (containing 1,000 negative reviews, 67,425 sentences, and 19,714 distinct words). The two dictionaries $\mathcal{V}^+$ and $\mathcal{V}^-$ are generated from $\mathcal{D}^+$ and $\mathcal{D}^-$, by finding most frequent positive/negative adjectives.

#	Positive	Frequency	Negative	Frequency
1	good	2146	bad	1414
2	great	882	stupid	214
3	funny	441	poor	152
4	special	282	awful	109
5	perfect	244	silly	97
6	beautiful	202	horrible	71
7	nice	184	suck	65
8	entertaining	179	violent	64
9	wonderful	165	sad	56
10	excellent	146	ugly	44

Fig. 2. The dictionaries $\mathcal{V}^+$ and $\mathcal{V}^-$

To not make our experiments too complex, we selected ten most frequent adjectives for each dictionary, listed as Fig. 2. The training documents $\mathcal{D}_L^+$ (contains 1,678 sentences) and $\mathcal{D}_L^-$ (contains 3,842 sentences) are therefore generated from $\mathcal{D}^+$ and $\mathcal{D}^-$ by gathering the sentences containing at least one adjective from $\mathcal{V}^+$ and $\mathcal{V}^-$.

The maximal frequent clauses (standing for $\mathcal{D}_F^+$ and $\mathcal{D}_F^-$) and the sentiment rules (standing for $\mathcal{P}^+$ and $\mathcal{P}^-$) extracted by the sequential pattern mining algorithm are shown in Fig. 3. For instance, with $min\_supp = 0.001$, we find 160 distinct sentiment rules from 572 discovered maximal frequent clauses in positive reviews, however with $min\_supp = 0.01$, only 8 distinct sentiment rules are found from 19 frequent clauses. The 10 most frequent sentiment rules are listed in Fig. 4. The antonym dictionaries for constructing the belief bases are given by WordNet. For respecting the size limit of this paper, we list a small part of the two belief bases in Fig. 5.

In order to analyze the accuracy of our approach, we randomly select a number of beliefs for extracting the sentences that express the sentiment opposite to the documents $\mathcal{D}^+$ and $\mathcal{D}^-$. For instance, as the beliefs listed in Fig. 5, the 5 beliefs of positive sentiment produced totally 304 unexpected sentences, and 236 of them express the negative sentiment; the 5 beliefs of negative sentiment produced totally 136 unexpected sentences, and 97 of them express the positive sentiment. Within these beliefs, the average accuracy is about 74.48%.

² <http://www.cs.cornell.edu/People/pabo/movie-review-data/>

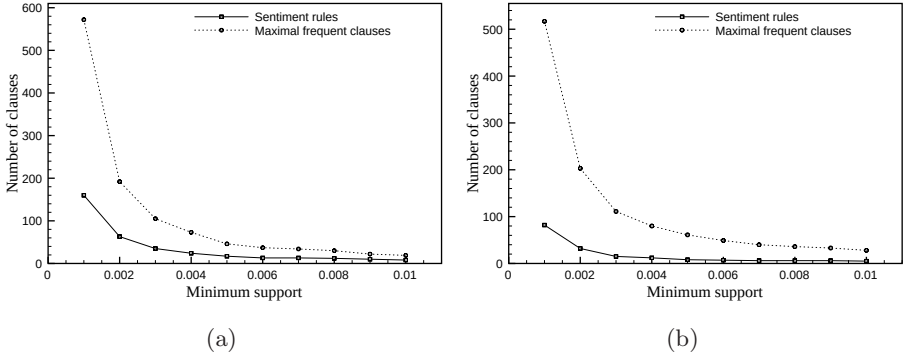


Fig. 3. (a) Maximal frequent clauses and sentiment rules of positive reviews. (b) Maximal frequent clauses and sentiment rules of negative reviews.

Positive Sentiment Rules	Negative Sentiment Rules
$\langle \langle \text{be} V \rangle \rangle \Rightarrow \langle \langle \text{good} J \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{guy} N \rangle \rangle$
$\langle \langle \text{good} J \rangle \rangle \Rightarrow \langle \langle \text{film} N \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{be} V \rangle \rangle$
$\langle \langle \text{good} J \rangle \rangle \Rightarrow \langle \langle \text{be} V \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{movie} N \rangle \rangle$
$\langle \langle \text{good} J \rangle \rangle \Rightarrow \langle \langle \text{performance} N \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{film} N \rangle \rangle$
$\langle \langle \text{good} J \rangle \rangle \Rightarrow \langle \langle \text{movie} N \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{thing} N \rangle \rangle$
$\langle \langle \text{good} J \rangle \rangle \Rightarrow \langle \langle \text{friend} N \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{year} N \rangle \rangle$
$\langle \langle \text{great} J \rangle \rangle \Rightarrow \langle \langle \text{film} N \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{time} N \rangle \rangle$
$\langle \langle \text{great} J \rangle \rangle \Rightarrow \langle \langle \text{be} V \rangle \rangle$	$\langle \langle \text{bad} J \rangle \rangle \Rightarrow \langle \langle \text{dialogue} N \rangle \rangle$
$\langle \langle \text{special} J \rangle \rangle \Rightarrow \langle \langle \text{be} V \rangle \rangle$	$\langle \langle \text{stupid} J \rangle \rangle \Rightarrow \langle \langle \text{be} V \rangle \rangle$
$\langle \langle \text{special} J \rangle \rangle \Rightarrow \langle \langle \text{effect} N \rangle \rangle$	$\langle \langle \text{poor} J \rangle \rangle \Rightarrow \langle \langle \text{be} V \rangle \rangle$

Fig. 4. The 10 most frequent sentiment rules

Belief Base of Positive Sentiment	Belief Base of Negative Sentiment
$[\langle \langle \text{be} V \rangle \rangle; \langle \langle \text{good} J \rangle \rangle; \langle \langle \text{bad} J \rangle \rangle]$	$[\langle \langle \text{not} R \rangle \rangle \langle \langle \text{bad} J \rangle \rangle; \emptyset; \langle \langle \text{guy} N \rangle \rangle]$
$[\langle \langle \text{be} V \rangle \rangle; \langle \langle \text{good} J \rangle \rangle; \langle \langle \text{not} R \rangle \rangle \langle \langle \text{good} J \rangle \rangle]$	$[\langle \langle \text{n't} R \rangle \rangle \langle \langle \text{bad} J \rangle \rangle; \emptyset; \langle \langle \text{guy} N \rangle \rangle]$
$[\langle \langle \text{be} V \rangle \rangle; \langle \langle \text{good} J \rangle \rangle; \langle \langle \text{n't} R \rangle \rangle \langle \langle \text{good} J \rangle \rangle]$	$[\langle \langle \text{bad} J \rangle \rangle; \langle \langle \text{be} V \rangle \rangle; \langle \langle \text{be} V \rangle \rangle \langle \langle \text{not} R \rangle \rangle]$
$[\langle \langle \text{bad} J \rangle \rangle; \emptyset; \langle \langle \text{film} N \rangle \rangle]$	$[\langle \langle \text{bad} J \rangle \rangle; \langle \langle \text{be} V \rangle \rangle; \langle \langle \text{be} V \rangle \rangle \langle \langle \text{n't} R \rangle \rangle]$
$[\langle \langle \text{not} R \rangle \rangle \langle \langle \text{good} J \rangle \rangle; \emptyset; \langle \langle \text{film} N \rangle \rangle]$	$[\langle \langle \text{good} J \rangle \rangle; \emptyset; \langle \langle \text{film} N \rangle \rangle]$
$[\langle \langle \text{n't} R \rangle \rangle \langle \langle \text{good} J \rangle \rangle; \emptyset; \langle \langle \text{film} N \rangle \rangle]$	$[\langle \langle \text{not} R \rangle \rangle \langle \langle \text{bad} J \rangle \rangle; \emptyset; \langle \langle \text{film} N \rangle \rangle]$
.....	

Fig. 5. The belief base for mining unexpected sentences

5 Conclusion

In this paper we present a belief-driven approach that extracts contextual opposite sentiment as unexpected sentences from classified free text reviews. We adapt the sequence data model to text mining with Part-of-Speech tags, so that

the extraction is associated with the semantic property of each word contained in the text reviews, thus the sequence mining techniques can be applied. Our experimental results show that the accuracy of the extracted opposite sentiments is in the acceptable range. Our future work includes to combine the adverbs and the conjunctions (like *however*, *but*) into the extraction process, and to integrate contextual opposite sentiments into document-sentence classifications.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE, pp. 3–14 (1995)
2. Dave, K., Lawrence, S., Pennock, D.M.: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In: WWW, pp. 519–528 (2003)
3. Esuli, A., Sebastiani, F.: PageRanking WordNet synsets: An application to opinion mining. In: ACL, pp. 424–431 (2007)
4. Fellbaum, C.: WordNet: An electronic lexical database. MIT Press, Cambridge (1998)
5. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: KDD, pp. 168–177 (2004)
6. Kamps, J., Mokken, R.J., Marx, M., de Rijke, M.: Using WordNet to measure semantic orientation of adjectives. In: LREC, pp. 1115–1118 (2004)
7. Li, D.H., Laurent, A., Poncelet, P.: Mining unexpected sequential patterns and rules. Technical Report RR-07027 (2007), LIRMM (2007)
8. Pang, B., Lee, L.: A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: ACL, pp. 271–278 (2004)
9. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: EMNLP, pp. 79–86 (2002)
10. Santorini, B.: Part-of-Speech tagging guidelines for the Penn Treebank project. Technical Report MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania (1990)
11. Schmid, H.: Probabilistic Part-of-Speech tagging using decision trees. In: NeMLaP (1994)
12. Sleator, D.D., Temperley, D.: Parsing English with a link grammar. In: 3rd International Workshop on Parsing Technologies (1993)
13. Turney, P.D.: Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 491–502. Springer, Heidelberg (2001)
14. Turney, P.D.: Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In: ACL, pp. 417–424 (2002)
15. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing contextual polarity in phrase-level sentiment analysis. In: HLT/EMNLP (2005)
16. Yu, H., Hatzivassiloglou, V.: Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In: EMNLP, pp. 129–136 (2003)

Navigational Path Expressions on XML Schemas

Federico Cavalieri¹, Giovanna Guerrini¹, and Marco Mesiti²

¹ Università di Genova, Italy
`guerrini@disi.unige.it`

² Università di Milano, Italy
`mesiti@dico.unimi.it`

Abstract. XML Schema is employed for describing the type and structure of information contained in valid XML documents. As for a document, a schema can be navigated and its components can be identified through a path language. In this paper we discuss the drawbacks of using XPath for this purpose and present *XSPath*, a language tailored for specifying path expressions on schemas.

1 Introduction

XML Schema [9] is being extensively used to represent domain specific document structures. Due to the presence of many alternative and repeatable elements, the dimension of the schemas can be considerable, while most valid documents contain a small subset of the elements declared in the schema. For example in the biological domain, MAGE-ML [7] is employed for the representation of different aspects of experiments (like measures, protocols, bio-materials, bio-sequences). The need may arise to retrieve the structure of the bio-materials element, that is, to retrieval components of the schema itself. Moreover, applications can request to Web servers only a given type of data (e.g., only the bio-sequences). Therefore, approaches for the retrieval of document elements based on type/structure constraints expressed through their schemas are needed as well.

Similar issues arise in other domains where languages for the retrieval of schema components as well as facilities of accessing parts of documents on the basis of schema constraints are desired. Queries on schemas obviously play an important role for retrieving information from multiple heterogeneous sources, in both query formulation and query optimization [3,6]. They allow to inspect the schema to obtain a proper formulation of queries. They are also useful in the identification and specification of mappings between schema elements. Also in third party data management [5] schema queries can be useful. The data owner can locally keep the schema of her data and can express queries on the local schema to locally download only a limited parts of remote documents, with positive effects on transfer time. We have personally experimented the usefulness of schema queries in the context of schema evolution [4] for the identification of schema components to be updated.

Because of the hierarchical nature of XML Schema, navigational expressions on the schema structure are a natural means to retrieve its main components

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="mail" type="mailType" />
  <xsd:complexType name="subjectType">
    <xsd:sequence>
      <xsd:element name="address" type="addressType" />
      <xsd:element name="name" type="xsd:string" minOccurs="0" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="mailType">
    <xsd:sequence>
      <xsd:element name="envelope" type="envelopeType" />
      <xsd:element name="body" type="xsd:string" />
      <xsd:element ref="attachment" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="envelopeType">
    <xsd:sequence>
      <xsd:element name="From" type="subjectType" />
      <xsd:element name="To" type="subjectType" maxOccurs="unbounded" />
      <xsd:element name="Date" type="xsd:dateTime" />
      <xsd:element name="Subject" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="header" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="attachment">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice minOccurs="0">
          <xsd:element name="picture" type="xsd:string" />
          <xsd:element name="audio" type="xsd:string" />
          <xsd:element name="movie" type="xsd:string" />
        </xsd:choice>
        <xsd:element name="content" type="xsd:string" minOccurs="0" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:simpleType name="addressType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[^\@]+\@[^\.]+\.[^\.]*" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>

```

Fig. 1. email.xsd schema

(namely, element declarations, complex and simple type definitions), to navigate in complex types structures, and to filter schema elements according to their values for properties like minimal and maximal occurrences of an element.

The adoption of XPath [10] for the specification of navigational expressions on schemas would result in the specification of complex expressions that do not reflect the user expectation in query formulation. Moreover, the occurrence of references to element declarations and the possibility to define the type of an element as global, require to specify expressions over internal links. Navigation through links is, however, not supported in XPath [10]. Suppose, for instance, that we wish to denote the global `mail` element in the schema in Fig. 1. The following XPath expressions could be specified.

- `/xsd:schema/xsd:element[@name="mail"]`. However, this expression would return the `mail` element without the specification of its complex type.

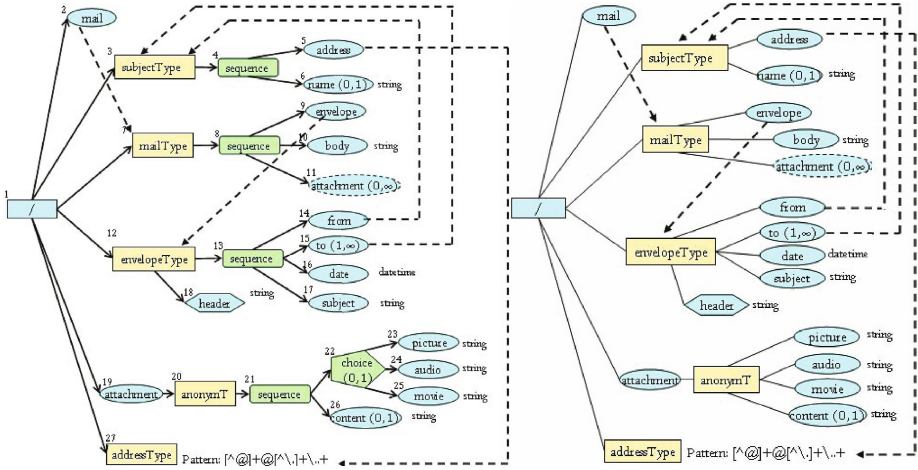


Fig. 2. Low and high level schema representations

– `/xsd:schema/xsd:complexType[@name=../xsd:element[@name="mail"]/@type]`. It would return the complex type of the `mail` element.

They are quite verbose and the intuition is that a simpler expression like `“/mail”` would be preferred. A simple extension of this query like: “find the declaration of the `body` element within `mail`” would make the XPath specification furthermore complicated while we would expect an expression like `“/mail/body”`.

Starting from the requirements for schema querying, broadly discussed in the extended version of this paper [2], a two level graph-based representation of schemas (Sec. 2) is introduced for abstracting the details of schemas when they are not required. We then propose XSPaTh, a path language specifically tailored for XML Schema (Sec. 3) that derives from XPath and allows the specification of path expressions on the two level graph representation of schemas. The paper concludes by discussing how XSPaTh expressions are interpreted (Sec. 4).

2 Schema Representation

Low Level Schema Representation. The representation of a schema is a graph G consisting of a set of nodes and a set of edges among nodes. Nodes of a schema can have one of the following types: **root** representing the `<schema>` root component (a single node of this type must occur in each schema); **element** representing an element; **type** representing XSD types (both global and local), further specialized in **simpleType** and **complexType** types; **attribute** representing an attribute; **operator** representing an operator employed to specify the structure of an element or type. The **node** type is introduced as their generalization [2].

To distinguish different types of nodes in the graphical representation, appropriate symbols and colors are employed [2]. Nodes can be connected through

<i>Step</i>	::= <i>AxisSpec NodeSpec Predicate*</i>
<i>AxisSpec</i>	::= <i>LevelSpec</i> "::" <i>AxisName</i> "::"
<i>LevelSpec</i>	::= "HL" "LL"
<i>AxisName</i>	::= (usual <i>axis</i> . <i>Check</i> [2])
<i>NodeSpec</i>	::= <i>NodeType</i> "(" <i>QName?</i> ")"
<i>NodeType</i>	::= "attribute" "element" "node" "operator" "type"
<i>BasicCond</i>	::= <i>ExistCond</i> <i>PosCond</i> <i>TypeCond</i> <i>DTypeCond</i> <i>PropCond</i> <i>ResCond</i>
<i>ExistCond</i>	::= "exists(" <i>SinglePath</i> ")"
<i>PosCond</i>	::= "position()" <i>CompareOp IndexPos</i>
<i>TypeCond</i>	::= "type()" <i>EqOp</i> (<i>SinglePath</i> <i>BuiltInType</i>) "type() is" ("simple" "complex" "anonym")
<i>DTypeCond</i>	::= "typeDerivedFrom(" <i>SimpleType</i> ")"
<i>PropCond</i>	::= "property(" <i>PropertyName</i> ")" (<i>CompareOp Value</i>)?
<i>ResCond</i>	::= "restriction(" <i>RestrictionName</i> ")" (<i>CompareOp Value</i>)?
<i>IndexPos</i>	::= <i>Number</i> "last()"
<i>EqOp</i>	::= "=" "!="
<i>CompareOp</i>	::= "<=" ">=" "<" ">" <i>EqOp</i>

Fig. 3. Step and Basic conditions specification

direct and *link* edges. Direct edges model the hierarchical structure of the schema whereas link edges represent the structure of an element whose type is global, and a node which is a reference to a global element. The occurrence of link edges makes the schema a graph rather than a tree.

The left side of Fig. 2 reports the graph representation of the `email.xsd` schema in Fig. 1. The symbol for node `attachment`, within the `mailType` node, is dashed to denote that it is a reference/link to the global `attachment` element. Simple types have been reported next to the symbol representing the element whereas dashed lines are used to represent a link edge from an element to its global type. Since schemas are XML documents, they are totally ordered. The node rank is reported in the low level representation of the schema in Fig. 2.

High Level Schema Representation. In navigating schemas, however, we may be not interested in the detailed internal structure of complex types. A higher level representation allows the specification of simpler navigational expressions. The high level representation of a schema is a graph in which nodes of type `operator` are removed. The subelements the operators bind are attached to the corresponding type. The right side of Fig. 2 shows the high level representation of our running example.

3 XPath Specification

The basic building block of XPath is the path expression that consists of a sequence of one or more steps, separated by `/`, and optionally beginning with `/`. The only exception (i.e., zero steps) is the expression `/` which returns the entire schema. XPath expression evaluation is similar to that of XPath expressions.

3.1 Steps

Step specification includes: *axis specification*, *node specification*, and (optionally) *predicates* as reported on top of Fig. 3. The *axis specification* determines the direction toward which nodes should be identified starting from the context node and moving at a given level of abstraction. Axis specification is thus composed of the abstraction level and the axis name (as in XPath). The abstraction level is denoted LL for low level and HL for high level. When the context node is an operator in a complex type, the nodes identified by the axis are within the internal structure of the complex type.

Example 1. Consider the schema in Fig. 2. The following table reports for each context node and axis specification, the identified nodes. To uniquely identify each node, the name of the node is coupled with its rank in the graph. □

Context node	AxisSpec	Identified Nodes
sequence ²¹	LL::child	choice ²² ,content ²⁶
sequence ²¹	HL::child	picture ²³ , audio ²⁴ ,movie ²⁵ ,content ²⁶
picture ²³	LL::parent	choice ²²
picture ²³	HL::parent	anonymT ²⁰
subjectType ³	LL::descendant	sequence ⁴ ,address ⁵ ,name ⁶
subjectType ³	HL::descendant	sequence ⁴ ,address ⁵ , addressType ²⁷ ,name ⁶

Once the axis has been specified, the *node specification* determines the node to be selected in that direction. Selection criteria can be the type of a node and its name. For each node type a function is available to select all corresponding nodes. The `node()` function allows the selection of all nodes independently of their type. Each one of these functions takes an optional `QName` parameter which further refines the selection keeping only nodes of that type having the specified name. Inline type declarations, including anonymous ones, can be addressed using the `type()` function. The qualified name, eventually specified for the `operator` constructor, can only be `sequence`, `choice`, or `all`.

Example 2. Consider the schema in Fig. 2. Starting from the given context nodes, the application of the node specification filters the nodes as follows. □

Context Nodes	NodeSpec	Identified Nodes
sequence ⁴ , address ⁵ , name ⁶	<code>node()</code>	sequence ⁴ , address ⁵ , name ⁶
picture ²³ , audio ²⁴ , video ²⁵	<code>element(picture)</code>	picture ²³
sequence ⁸ , sequence ²¹ , choice ²²	<code>operator(sequence)</code>	sequence ⁸ , sequence ²¹

Predicates can be employed in node specification to further filter the sequence of nodes by specifying conditions delimited by square brackets. Each node of the sequence is the context for the evaluation of conditions. Conditions can be the boolean combination of basic conditions whose syntax is in the bottom of Fig. 3. All the kinds of conditions require compatibility among the operands. When compatibility does not occur, the *false* value is returned.

Abbr.for	Long Form	Short Form
Axis	HL::child	/
	HL::descendant-or-self	/
	LL::child	!
	LL::descendant-or-self	!!
Node type	attribute()	@*
	attribute(<i>QName</i>)	@ <i>QName</i>
	operator()	*
	operator(<i>OperatorName</i>)	<i>OperatorName</i>
	type()	##
	type(<i>QName</i>)	# <i>QName</i>
	element()	*
	element(<i>QName</i>)	<i>QName</i>
Step	self::node()	.
	HL::parent::node()	..
	LL::child::type()	type()
Predicate	[position() <i>=IndexValue</i>]	[<i>IndexValue</i>]
	[exists(<i>SinglePath</i>)]	[<i>SinglePath</i>]

Table 1. Abbreviated Syntax

- *ExistCond*. For each node of the sequence, the existence of a path is checked. The condition is satisfied by the node whenever the path evaluation returns a non-empty set.
- *PosCond*. The nodes of the sequence whose position meets the comparison criteria are returned. The position is compared with a number or with **last()** representing the last position in the sequence.
- *TypeCond*. This condition can be applied to nodes representing elements, types, or attributes. The condition is satisfied by a node in the sequence if: (i) its type meets the comparison criterion with a simple type (either built-in or user-defined); (ii) its type is simple, complex, or anonymous.
- *DTypeCond*, *ResCond*. These conditions are satisfied by nodes representing simple types. A node satisfies the first condition if its type meets the comparison criterion with a built-in native type (or the type returned by the evaluation of a path).
- *PropCond*. This condition is satisfied by nodes with properties. A node in the sequence satisfy the condition if the property meets a comparison criterion.

Example 3. Consider the schema in Fig. 2. Starting from the given context nodes, the application of predicates filters the nodes as follows. □

Context Nodes	Predicate	Ident. Nodes
sequence ¹³ , sequence ²¹	exists(LL::child::operator())	sequence ²¹
from ¹⁴ , to ¹⁵ , date ¹⁶ , subject ¹⁷	position() <i>>=</i> last() <i>-1</i>	date ¹⁶ subject ¹⁷
mail ² , attachment ¹⁹	type() <i>is</i> anonym	attachment ¹⁹
address ⁵ , name ⁶	property(minOccurs)=0	name ⁶

3.2 Abbreviated Syntax

Following the same idea of XPath, XSPaht commonly used expressions can be expressed in the abbreviated syntax. Table 1 shows all available shorter forms along with their equivalent long form.

#		XSPath expr.	eval.
1	EXT ABBR	/HL::child::element(mail)/HL::child::element(attachment) /mail/attachment	{19}
2	EXT ABBR	/HL::child::type(envelopeType)/HL::child::attribute(header) /#envelopeType/@header	{18}
3	EXT ABBR	/HL::descendant-or-self::element() [/HL::child::type()='subjectType'] //*[type()='subjectType']	{14, 15}
4	EXT ABBR	/HL::descendant-or-self::type() [restriction(pattern)] //##[restriction(pattern)]	{27}
5	EXT ABBR	HL::child::element(attachment)/LL::child::operator(sequence) /LL::child::operator(choice) ./attachment!sequence!choice	{22}

Fig. 4. Examples of XSPath expressions

Child and descendant-or-self axes can be abbreviated both at high and low level. At high level, the abbreviation is identical to the abbreviation in XPath (i.e. //). At low level, the / separator between two steps is removed when it is followed by ! or !!. This simplifies the specification of paths at low level. The relative expression `LL::child::operator(sequence)/LL::child::operator(choice)`, for instance, can be shortened as `!sequence!choice`.

Also the following entire steps can be abbreviated: the step which identifies the current nodes, the parent nodes of the context node, and the context node type definition. Conditions on the position of a node in a sequence and on the existence of a path can be abbreviated as well. Since a node of type `element` is always associated with a single node of type `type` the following kind of path: `lev::child::element(elemName)/lev::child::type()/...` where `lev` is one of the possible levels, and `elemName` is the name of an element, can be shortened as follows: `lev::child::element(elemName)/...`

3.3 Examples of Navigational Expressions

Referring to the schema in Fig. 2, some examples of navigational expressions are presented. For each query, Fig. 4 reports the corresponding extended and abbreviated XSPath expressions and the nodes identified in the graph. Query (1) shows the simpler form of expression: starting from the root node /, it allows to navigate through elements. The type of node `attachment` can be omitted because each node has always a single type. The expression allows the navigation through a link edge. Query (2) shows a path starting from a node representing a type (identified through the # symbol) and ending up with a node representing an attribute (identified through the @ symbol). Query (3) shows a path starting from an arbitrary descendant node (using // in the abbreviated syntax) arriving at any node representing an element (* means any name). A structural condition is imposed for selecting nodes whose type is `subjectType`. Query (4) shows an expression used to find types. Specifically, the filtering condition checks that the identified node has a restriction property whose type is `pattern`. The previous expressions work on the high level representation of a schema. Query (5), by contrast, is an example of expression working at both levels. Indeed, a high level step is employed to identify element `attachment` starting from the context node `mail`. Then, through low level steps the `choice` operator is reached.

4 Conclusions

In this paper we have proposed a navigational XML Schema query language. The language derives from XPath and is characterized by a seamless switching between abstraction levels in navigating the schemas, where the low level exposes the operators in the definition of complex types structures while the high level masks them. As far as we know this is the first attempt to develop a path language for XML Schema. However, our work takes advantage of concepts developed in the area of XPath 2.0, XQuery, and both relational [6] and object-oriented [3] schema-based query languages. XPath 2.0 expressions can contain conditions on the type of elements and attributes, on the repeatability of an element, and so on. XSPPath, however, differs from XPath 2.0 because it is specifically tailored to work on schemas rather than on documents. Moreover, XSPPath has been designed to work on a graph schema model more complex than the Infoset (tree) model of XML documents.

XSPPath semantics can be found in [2]. The semantics relies on the XPath semantics [11]. XSPPath expressions allow the identification of nodes in the schema graph of type **element**, **operator**, **root**, **type**, and **attribute**. Since XSPPath expressions could be exploited to identify nodes in valid documents, a correspondence between each type of nodes in the schema and the nodes in a document must be specified. XPath expressions on the documents are obtained for the identification of elements and attributes relying on a function ($\mathcal{DP}$, see [2]) that is applied on each node returned from the evaluation of an XSPPath expression and generates a set of XPath expressions. These XPath expressions are then applied on the valid documents to return the document portions.

An interpreter for XSPPath expressions has been developed, using the parser generator JavaCC (<https://javacc.dev.java.net/>). Each expression is translated into a union of XPath expressions. A single expression gives rise to a union of XPath expressions because a given node in the graph representation of a schema can be reached from the root through different paths. The obtained XPath expressions can be executed through XQilla (<http://xqilla.sourceforge.net/>) on schemas and documents contained in a commercial XML-enabled DBMS.

References

1. Amer-Yahia, S., et al.: Approximate Matching in XML. In: ICDE (2003)
2. Cavalieri, F., Guerrini, G., Mesiti, M.: Navigational Path Expressions on XML Schemas. Technical report (2008), <http://www.disi.unige.it/person/GuerriniG>
3. Chaudhri, V.K., Karp, P.D.: Querying Schema Information. In: KRDB. CEUR Workshop Proceedings, vol. 8, pp. 4.1–4.6(1997)
4. Guerrini, G., Mesiti, M., Sorrenti, M.: Schema Evolution: Incremental Validation and Efficient Document Adaptation. In: Xsym, pp. 92–106 (2007)
5. Hacigumus, H., et al.: Providing Database as a Service. In: ICDE (2002)
6. Lakshmanan, L., Sadri, F., Subramanian, S.: SchemaSQL – An Extension to SQL for Multidatabase Interoperability. ACM Transaction on Database Systems 26(4), 476–519 (2001)

7. Spellman, P.T., et al.: Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biology* 3 (2002)
8. Theobald, A., Weikum, G.: Adding Relevance to XML. In: Suciu, D., Vossen, G. (eds.) *WebDB 2000. LNCS*, vol. 1997, pp. 105–124. Springer, Heidelberg (2001)
9. W3C. XML Schema, Second Edition (2004)
10. W3C. XML Path Language (XPath) 2.0 (2007)
11. Wadler, P.: Two Semantics for XPath (1999)

Transforming Tree Patterns with DTDs for Query Containment Test

Junhu Wang¹, Jeffrey Xu Yu², Chengfei Liu³, and Rui Zhou³

¹ Griffith University, Gold Coast, Australia
J.Wang@griffith.edu.au

² Chinese University of Hong Kong, Hong Kong, China
yu@se.cuhk.edu.hk

³ Swinburne University of Technology, Melbourne, Australia
{cliu, rzhou}@ict.swin.edu.au

Abstract. We study the problem of testing XPATH query containment under DTDs, where the XPATH expressions are given as tree patterns involving $/, //, []$ and $*$, and the DTD are given as acyclic schema graphs. We focus on efficient algorithms to transform a tree pattern P involving $*$ into a new one P' which does not have $*$, using DTD G , so that testing containment of P in any other pattern Q under G is reduced to testing whether P' is contained in Q without DTD, provided Q does not have $*$.

1 Introduction

Query containment, which is to decide whether the answer set of one query is always a subset of another, is fundamental for many applications including query optimization and query rewriting using views. The problem has been well studied for relational databases. With the increasing importance of XML data, the problem of XML query containment has attracted many researchers (see, e.g., [5,4,6]). In particular, algorithms for testing containment of tree patterns were studied, in the absence of DTDs, in [3]. Recall: a tree pattern represents an XPATH expression that may involve the child axis ($/$), descendant axis ($//$), branching conditions ($[]$), and wildcard ($*$). When a DTD is present, [7] shows that if the DTD is *duplicate-free* and the tree patterns involve only $/$ and $[]$, then testing whether tree pattern, P , is contained in another pattern, Q , under the DTD can be reduced to testing whether P is contained in Q under two types of constraints implied by the DTD. This result was recently extended in [2] to tree patterns involving $/, //$ and $[]$, under non-recursive and non-disjunctive DTDs. It is shown that in this case, testing whether P is contained in Q can be done by chasing P to P' using five types of constraints (hereafter referred to as the *LWZ constraints*) implied by the DTD, and then test whether P' is contained in Q without the DTD.

In this paper, we study efficient algorithms for testing containment of tree patterns involving all of $/, //, []$ and $*$, under DTDs that can be represented as acyclic schema graphs. It is worth noting, however, for such tree patterns, testing whether one is contained in another is co-NP hard even when there is

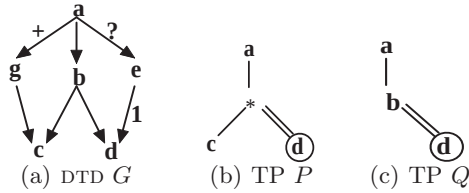


Fig. 1. Example DTD and tree patterns

no DTD [3]. Thus it is unlikely that an efficient algorithm exists in the presence of DTDs. In this work we concentrate on a common restricted case, that is, with P involving $*$ but Q not. It turns out that allowing the extra $*$ in P makes the problem considerably more complicated, and a straightforward extension to the constraint-based method of [7] and [2] will not work. Thus we use a direct transformation approach: to transform P to an equivalent DAG-pattern P' under the DTD using a set of algorithms (rather than constraints and chase rules) first, and then do further processing using the *LWZ* constraints. We make the following contributions: (1) Given DTD G as an acyclic schema graph, we transform TP P involving $*$ into P' without $*$, such that for any tree pattern Q that does not have $*$, P is contained in Q under G iff P' is contained in Q without G . (2) We define DAG-patterns, and show DAG-patterns are a necessary means in the above process of transformation.

The rest of the paper is organized as follows. The preliminaries are given in Section 2. Section 3 presents the TP transformation algorithms. Section 4 concludes the paper.

2 Preliminaries

DTDs, XML Trees and Tree Patterns. We assume a set Σ of XML tags, and adopt the approach of [2] to model a non-disjunctive DTD as a connected directed graph G such that (1) each node is labeled with a distinct tag, (2) each edge is labeled with one of 1 , $?$, $+$, and $*$, which indicate “exactly one”, “one or zero”, “one or many”, and “zero or many”, respectively. Here, *the default edge label is $*$* , and (3) there is a unique node, called the root, which has an incoming degree of zero. The set of tags occurring in G is denoted Σ_G . Because a node in a DTD G has a unique label, we also refer to a node by its label. In this paper we will implicitly assume all DTDs are acyclic. Such a graph represents a non-recursive and non-disjunctive DTD. We will use DTD and schema graph interchangeably. A DTD example is shown in Figure 1 (a).

An XML *tree* is a tree with every node labeled with a tag in Σ . A *tree pattern* (TP) is a tree with a unique *distinguished node*, and with every node labeled with a symbol in $\Sigma \cup \{*\}$ (here $*$ is the wildcard which represents any tag), every edge labeled with either $/$ or $//$. The path from the root to the distinguished node is called the *distinguished path*. Figure 1 (b) and (c) show two TPs P and Q , where single and double lines are used to represent $/$ -edges and $//$ -edges

respectively, and a circle is used to indicate the distinguished node. A TP corresponds to an XPATH expression. The TP in Figure 1 (b) corresponds to the expression $a/*[c]//d$. Let P be a TP. We will use DN_P , and DP_P to denote the distinguished node and the distinguished path of P respectively. Note: the TPs in our discussion correspond to the fragment $\mathcal{P}\{//, \cdot, *\}$ defined in [3]. A subset of $\mathcal{P}\{//, \cdot, *\}$, denoted $\mathcal{P}\{//, \cdot, \emptyset\}$, contains all TPs that do not have $*$ -nodes.

Below, for any tree or DTD T , we will use $N(T)$, $E(T)$ and $rt(T)$ to denote the node set, the edge set, and the root of T respectively. We will also use $\text{label}(v)$ to denote the label of node v , and call a node labeled a an a -node. An XML tree t is said to conform to DTD G if (1) for every node $v \in N(t)$, $\text{label}(v) \in \Sigma(G)$, (2) $\text{label}(rt(t)) = \text{label}(rt(G))$, (3) for every edge, $(u, v) \in E(t)$, there is a corresponding edge $(\text{label}(u), \text{label}(v))$ in G , and (4) for every node $v \in N(t)$, and every tag $\tau \in \Sigma(G)$, the number of children of v labeled with τ is constrained by the label of the edge $(\text{label}(v), \tau)$ that appears in G . We denote the set of all XML trees conforming to G by T_G .

A *matching* of a TP P in an XML tree t is a mapping δ from $N(P)$ to $N(t)$ which is (1) *label-preserving*, i.e., $\forall v \in N(P)$, $\text{label}(v) = \text{label}(\delta(v))$ or $\text{label}(v) = *$, (2) *root-preserving*, i.e., $\delta(rt(P)) = rt(t)$, and (3) *structure-preserving*, i.e., for every $/$ -edge (x, y) in P , $\delta(y)$ is a child of $\delta(x)$; for every $//-$ edge (x, y) , there is a path from $\delta(x)$ to $\delta(y)$. Each matching δ produces a node $\delta(\text{DN}_P)$, which is known as an *answer* to the TP. We use $P(t)$ to denote the *answer set* of TP P over an XML t , i.e., $P(t) = \{\delta(\text{DN}_P) \mid \delta \text{ is a matching of } P \text{ in } t\}$.

A TP P is said to be *satisfiable* under G , if there exists an XML tree $t \in T_G$ such that $P(t) \neq \emptyset$. In this paper, when we discuss TPs under a DTD G , we will implicitly assume the TPs involved are satisfiable under G .

Tree Pattern Containment and Containment Mapping. A TP P is said to be *contained* in another TP Q , denoted $P \subseteq Q$, if for every XML tree t , $P(t) \subseteq Q(t)$. In the presence of DTD G , P is said to be *contained in Q under G* , denoted $P \subseteq_G Q$, if for every XML tree $t \in T_G$, $P(t) \subseteq Q(t)$. P and Q are said to be *equivalent* (resp. *equivalent under G*) if $P \subseteq Q$ and $Q \subseteq P$ (resp. $P \subseteq_G Q$ and $Q \subseteq_G P$).

***-nodes and *-paths.** We call a node labeled $*$ a **-node*. In addition, we use **-path* to refer to a path, in a TP, that starts from a non- $*$ node, followed by a consecutive sequence of $*$ -nodes, and ends with a non- $*$ node or a leaf $*$ -node in P . In particular, a $*$ -path that ends with a leaf $*$ -node in P will be called an *open *-path*, and a $*$ -path that ends with a non- $*$ node will be called a *closed *-path*. Given any tree T , we use the term a -parent (resp. a -child, a -descendant) to refer to the parent (resp. child, descendant) node which is labeled a .

3 Tree Pattern Transformation under DTDs

Given a TP P and a DTD G , we will transform P into another pattern, P' , such that $P' =_G P$, and for every TP $Q \in \mathcal{P}\{//, \cdot, \emptyset\}$, $P \subseteq_G Q$ iff $P' \subseteq Q$. This process is divided into three steps: (1) $*$ -node relabeling, (2) $*$ -path expansion,

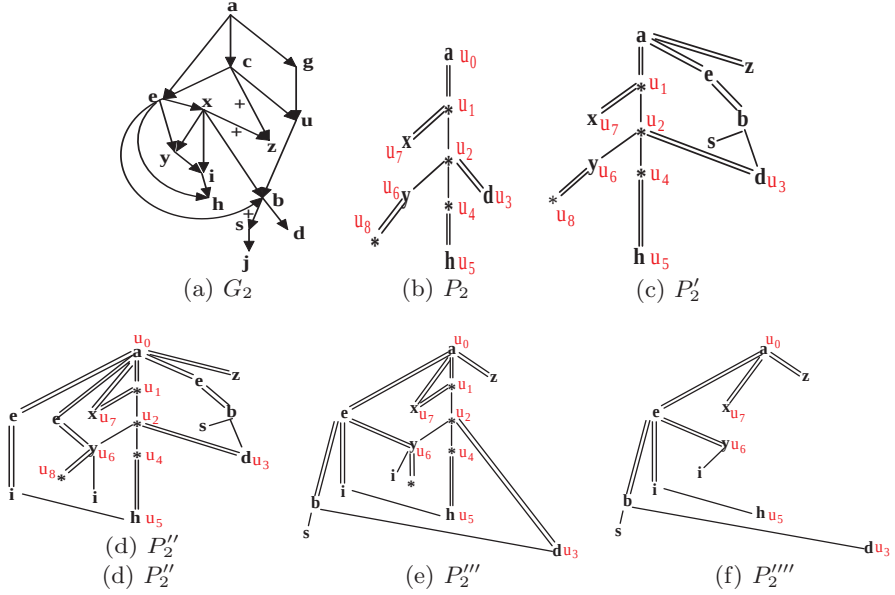


Fig. 2. DTD G_2 , TP P_2 , and the transformed patterns

and (3) processing using *LWZ* constraints. We focus on steps (1) and (2) here. We assume $label(rt(P)) \neq *$ (Otherwise, relabel $rt(P)$ with $label(rt(G))$).

3.1 Relabeling *-Nodes

Consider the TP P and DTD G in Figure 1. With G , any node that has both c -child and d -descendant must be a b -node. Therefore, the $*$ -node in P can be *relabelled* with b , resulting a TP equivalent to P under G .

Generally, $*$ -node relabeling can be done using the algorithm **FindLabel** in [1] which can find, for each node v , the set $L(v)$ of all possible labels under G . We can relabel v with c if $L(v) = \{c\}$. The algorithm runs in $O(|N(P)| \times |N(G)|^2)$.

3.2 Expanding *-Paths

Before presenting our algorithms for expanding $*$ -paths, we need to define some terms and notation. We will use a sequence of dot-separated labels to denote a path in G , and say a path in a DTD is a *mandatory path* if all of its edges are labeled 1 or +. Let G be a DTD, and $p = a_1.a_2 \dots .a_n$ be a path in G . By an “instance” of p , we mean a chain of nodes $v_1, v_2, \dots, v_n$ connected by $/$ -edges such that $label(v_i) = a_i$ ($i \in [1, n]$). We will also use $/$ -path to refer to a path that consists of only $/$ -edges.

Definition 1. Let $p_v = u_0 /_1 u_1 /_2 \dots /_k u_k$ be a $*$ -path in TP P (where $/_i$ is either $/$ or $//$), and $p_G = l_0.l_1 \dots .l_m$ ($m \geq k$) be a path in G . We say p_G is an image of p_v in G if there is a mapping ρ from $N(p_v)$ to $N(p_G)$ such that

Algorithm 1. ExpandTP(P, G)

```

1: for all closed *-path  $p$  in  $P$  do
2:   ExpandClosedStarPath( $p, G$ )
3: for all open *-path  $p$  in  $P$  do
4:   ExpandOpenStarPath( $p, G$ )
5: if *-paths  $p_1, \dots, p_m$  share some *-nodes and  $u$  is the last common *-node on  $p_1, \dots, p_m$  then
6:   if  $\exists$  a  $\tau$ -node on  $newpath(p_1), \dots, newpath(p_m)$ , and either Condition (A) or Condition
   (B) is true then
7:     merge the nodes labeled  $\tau$  on the paths  $newpath(p_1), \dots, newpath(p_m)$ 

```

1. $\forall i \in [0, k], \rho(u_i) \in L(u_i);$
2. $\forall i \in [0, k-1], \rho(u_i)$ is before (resp. immediately before) $\rho(u_{i+1})$ if $\neq_i$ is // (resp. /).

The mapping ρ above is called a match of p_V in p_G . The set of all images of p_V in G is denoted $image(p_V, G)$.

For example, for the TP and DTD in Figure 2, $a.c.e.b.d$, $a.c.e.x.b.d$, $a.e.x.b.d$ are images of the *-path $u_0//u_1/u_2//u_3$, while $a.c.u.b.d$, $a.g.u.b.d$ are not. Note that if p_V is not a /-path, there can be more than one match of p_V in an image p_G . For example, there are two matches of $u_0//u_1/u_2//u_3$ in $a.c.e.x.b.d$, which map u_1, u_2 to c, e and e, x respectively. It can be easily proved that, a path $p_G \in G$ is an image of a *-path $p_V = u_0 \neq_1 u_1 \neq_2 \dots \neq_k u_k \in V$ iff it starts from $label(u_0)$, ends at a node in $L(u_k)$, and passes through a node in each of $L(u_1), \dots, L(u_{k-1})$, and satisfies the length requirement of p_V .

Definition 2. Given a node $x \in G$, we use M_x to denote the set of all nodes in G reachable by some mandatory path from x , including x itself. Let p be a path in G and x, y be two nodes in p such that x appears before y . We use $p[x, y)$ to denote the set of nodes on p from x to the node immediately before y .

We are now ready to explain the *-path expansion process. Given G and P , the expansion of P using G is done by the procedure **ExpandTP**(P, G), as shown in Algorithm 1. The procedure first calls the two sub-procedures, **ExpandClosedStarPath** and **ExpandOpenStarPath**, to expand each individual *-path. **ExpandClosedStarPath**, as shown in Algorithm 2, deals with a closed *-path $p_V = u_0 \neq_1 u_1 \neq_2 \dots \neq_k u_k$ in V . It expands p_V by adding an additional path between u_0 and u_k , and possibly adding some children nodes under the nodes that are on the new path. For example, after expanding the *-path $u_0//u_1//u_2//u_3$, the pattern P_2 in Figure 2 (b) will become P'_2 shown in Figure 2 (c). We now explain the algorithm by looking at how it transforms P_2 into P'_2 . First, $image(p_V, G)$ is found, and the common nodes $a_1, \dots, a_n$ that lie on all $p \in image(p_V, G)$ are found (line 2-3). In our example, we find $image(p_V, G_2)$ contains $a.c.e.b.d$, $a.c.e.x.b.d$, $a.e.x.b.d$, and the common nodes are $a_1 = e$ and $a_2 = b$. Since the DTD is acyclic, these common nodes appear in the same order in all paths in $image(p_V, G)$, and they are arranged in such an order: $a_1, \dots, a_n$ (line 7-8). We then add a new path from u_0 to u_k that passes through a sequence of nodes, $v_1, \dots, v_n$, labeled

Algorithm 2. ExpandClosedStarPath (p_V, G)

```

1: Suppose  $p_V = u_0 /_{f_1} u_1 /_{f_2} \dots /_{f_k} u_k$ .
2: Find the image set  $image(p_V, G)$ ;
3: Find the set  $C$  of common nodes, excluding  $label(u_0)$  and  $label(u_k)$ , on all paths in
    $image(p_V, G)$ ;
4: if  $C = \emptyset$  then
5:    $n \leftarrow 0$ ;
6: else
7:   Suppose  $C = \{a_1, \dots, a_n\}$ ;
8:   Take any path from  $image(p_V, G)$  and check the order of occurrence of  $a_1, \dots, a_n$  on the
     path; Assume the order is  $a_1, \dots, a_n$ .
9: Add an additional path  $u_0 // v_1 // \dots // v_{n-1} // v_n // u_k$  in  $V$ , where  $label(v_i) = a_i$  (for all  $i \in [1, n]$ );
10:  $a_0 \leftarrow label(u_0)$ ;  $a_{n+1} \leftarrow label(u_k)$ ;  $v_0 \leftarrow u_0$ ;  $v_{n+1} \leftarrow u_k$ ;
11: for all  $p \in image(p_V, G)$  do
12:   take each node  $z \in p - \{u_k\}$  and find  $M_z$ , the mandatory descendant set of  $z$ ;
13:    $M(n+1, p) \leftarrow \{\}$ ;
14: for ( $i = n$  to  $0$ ,  $i - -$ ) do
15:   if on every path in  $image(p_V, G)$ ,  $a_i$  is immediately before  $a_{i+1}$  then
16:     change  $(v_i, v_{i+1})$  to  $/$ -edge;
17:   for all  $p \in image(p_V, G)$  do
18:      $M(i, p) \leftarrow M(i+1, p) \cup \bigcup_{z \in p[a_i, a_{i+1}]} M_z$ ;
19:    $M_i \leftarrow \bigcap_{p \in image(p_V, G)} M(i, p) - \{a_i\}$ ;
20:   for each label  $\tau \in M_i$  do
21:     if  $\exists \tau' \in M_i$  such that there is a mandatory path from  $\tau'$  to  $\tau$  in  $G$  then
22:        $M_i \leftarrow M_i - \{\tau\}$ ;
23:   for each  $\tau \in M_i - M_{i+1}$  do
24:     add a  $\tau$ -child  $v$  under  $v_i$  and label the edge  $(v_i, v)$  with  $//$ , if  $v_i$  does not have a  $\tau$ -
     descendant already;
25:   if the only path from  $a_i$  to  $\tau$  in  $G$  is the edge  $(a_i, \tau)$  then
26:     relabel  $(v_i, v)$  with  $/$ ;

```

vspace-0.4cm

$a_1, \dots, a_n$ respectively, and the edges on this path are temporarily labeled $//$ (line 9). Each edge on the path will be relabeled with $/$ if their labels appear in parent-child relationship in all paths of $image(p_V, G)$ (line 14-16). In our example, we add a path from u_0 to u_3 that passes through an e -node, followed by a b -node, and the edge between the b -node and u_3 is relabeled with $/$. Line 11-12 find the mandatory descendant set M_z for each node z which lies in a path in $image(p_V, G)$. Based on this, line 17-19 find the set M_i of all nodes $x \in G$ such that, for all $p \in image(p_V, G)$, there exists a node y in $p[a_i, u_k]$ such that there is a mandatory path from y to x . Note that $M_{i+1} \subseteq M_i$. In our example, we find $M_0 = \{z, s, j\}$, $M_1 = \{s, j\}$ and $M_2 = \{s, j\}$. Line 20-22 refine M_i by removing redundant nodes, i.e., nodes that can be reached by mandatory paths from other nodes (this is for efficiency reasons). In our example, node j is removed from M_0 , M_1 and M_2 , resulting $M_0 = \{z, s\}$, $M_1 = \{s\}$ and $M_2 = \{s\}$. Since the pattern P is assumed to be satisfiable under G , the node v_i may be added a τ -descendant for every $\tau \in M_i$. Line 23-24 add such descendant nodes (if such descendants do not exist already), but if a τ -descendant is added under v_{i+1} , there is no need to add it under v_i (this explains the condition $\tau \in M_i - M_{i+1}$ in line 23). In our example, we add an s -node under the b -node, and add a z -node under v_0 ($\equiv u_0$). Line 25-26 then relabel the edge (v_i, v) with $/$ if possible. In the example, the edge from the b -node to the s -node is relabeled.

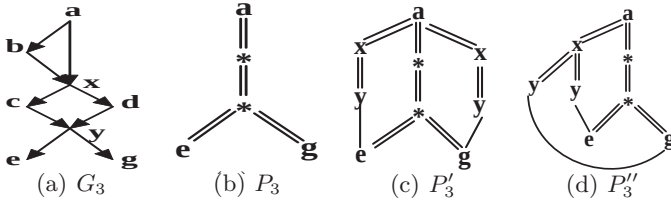


Fig. 3. DTD G_3 , TP P_3 , and chased patterns

ExpandOpenStarPath is a minor variation of **ExpandClosedStarPath**, which is used to expand open $*$ -paths. For example, it expands the $*$ -path $u_6//u_8$ by adding an i -child under u_6 , as shown in Figure 2 (d). The main difference from **ExpandClosedStarPath** lies in line 3, where $label(u_k)$ is not excluded, and in line 9, where the new path is $u_0//v_1//\dots//v_{n-1}//v_n$. In addition, the $n+1$ and n in line 13 and 14 are changed to n and $n-1$ respectively. We omit the details here due to page limit.

Let us denote the new path added due to $*$ -path p , using the sub-procedures, $newpath(p)$. After expanding all individual $*$ -paths, the main procedure **ExpandTP**(V, G) considers the case where some $*$ -paths, e.g., $p_1, \dots, p_m$, share $*$ -nodes. If u is the last common $*$ -node on all these $*$ -paths (i.e., u appears on all $p_1, \dots, p_m$, and no $*$ -descendant of u appears in all $p_1, \dots, p_m$), then it checks whether u satisfies one of the following conditions:

Condition (A): every match of p_1 , in every image of p_1 , maps u to a position at or after τ .

Condition (B): $\forall p_G \in image(p_1, G)$ and $\forall$ match e of p_1 in p_G , if τ is after $e(u)$, then for any $j \in [1, m]$, and any image q_G of p_j such that q_G contains $e(u)$ and u can be matched to $e(u)$, the segment on p_G from $e(u)$ to τ appears on q_G , and every edge on the segment is labeled $?$ or 1 .

If Condition (A) is true (in this case τ must appear in all images of $p_2, \dots, p_m$), all the τ -nodes on $newpath(p_1), \dots, newpath(p_m)$ are merged (and so are the path segments from the common start node to the τ -node in the new paths). For example, in Figure 2, the node u_2 is the last common $*$ -node on the $*$ -paths $u_0//u_1/u_2//u_3$, $u_0//u_1/u_2//u_4//u_5$, and $u_0//u_1/u_2//u_6$. u_2 is mapped to a position at or after e by all matches of $u_0//u_1/u_2//u_6$ in all images of these paths. Therefore, we can merge the e -nodes on the new paths added for these $*$ -nodes (the pattern after expanding all individual $*$ -paths is P_2'' in Figure 2), resulting P_2''' as shown in Figure 2 (e).

Note that the condition (A) and (B) above are important. If neither condition is true, we should not merge the τ -nodes, even if every $newpath(p_1), \dots, newpath(p_m)$ has a τ -node. For example, for the DTD G_3 and TP P_3 in Figure 3, the new paths added for both $*$ -paths pass through an x -node and a y -node (see Figure 3 (c)), and we can only merge the x -nodes on the new paths, not the y -nodes (see Figure 3 (d)).

XSelMark: A Micro-benchmark for Selectivity Estimation Approaches of XML Queries

Sherif Sakr

National ICT Australia (NICTA)
Sydney, Australia
`sherif.sakr@nicta.com.au`

Abstract. Estimating the sizes of query results and intermediate results is a crucial part of any effective query optimization process. Due to several reasons, the selectivity estimation problem in the XML domain is more complicated than that in the relational domain. Several research efforts have proposed selectivity estimation approaches in the XML domain. Lacking of a *suitable* benchmark was one of the main reasons which prevented a *real* assessment and comparison between the approaches to be conducted. In this paper we propose a selectivity estimation benchmark for XML queries, XSelMark. It consists of a set of 25 queries organized into seven groups and covers the main aspects of selectivity estimation of XML queries. These queries have been designed with respect to an XML document instance of a popular benchmark for XML data management, XMark. In addition, we suggest some criteria of assessing the capability and quality of XML queries selectivity estimation approaches. Finally, we use the proposed benchmark to assess the capabilities of the-state-of-the-art of the selectivity estimation approaches.

1 Introduction

Modern implementations of query processors are heavily relying for their efficient performance on sophisticated optimizer components to achieve a proper selection of many optimization decisions such as: access paths, join orders and materialization strategies. Estimating the sizes of query results and intermediate results is a crucial part of any effective query optimization process. In fact, the selectivity estimation problem in the XML domain is more complicated than that in the relational domain. There are several reasons behind this such as: 1) the absence of strict schema notion in the XML data. 2) the dualism between structural and value-based querying. 3) the high expressiveness of the XML query languages [5]. 4) the non-uniform distribution of tags and data. 5) the correlation and dependencies between the occurrences of the elements. In the recent past, several research efforts have proposed different selectivity estimation approaches in the XML domain [6,15,16]. However, these approaches are never comprehensively assessed, evaluated and compared. One of the main reasons for this situation is that there is a lack of a *suitable* benchmark that facilitates the ability to conduct such real assessments and comparisons.

Although the XML research community has proposed several benchmarks [3,7,13,14,17,20] which are very useful for their intended targets and perspectives, none of these benchmarks fits in the context of being able to assess and evaluate the different selectivity estimation approaches of XML queries. The author of this paper has been faced with this problem during his work in [16,18]. In [13], Michiels et al. have motivated the crucial need of different micro-benchmarks in order to get a good understanding of the different aspects in implementing efficient query processors in the XML domain. Therefore, the goal of this paper is to contribute and develop an XML Micro-benchmark, *XSelMark*, which is mainly focussed on exercising the selectivity estimation aspects of XML queries. The proposed benchmark aims of to be a guide for researchers and implementors in benchmarking and improving their research efforts in this domain. *XSelMark* consists of 25 queries organized into seven groups where each group is intended to address the challenges posed by the different aspects of XML query result size estimation.

The remainder of this paper is organized as follows. Section 2 briefly gives an overview on the related benchmarks in the XML domain. Section 3 describes the main aspects of the selectivity estimation problem in the XML domain. Section 4 presents the set of queries of the *XSelMark* benchmark. A brief overview and an assessment of the supported features of the-state-of-the-art in the selectivity estimation approaches of XML queries is presented in Section 5 before we conclude Section 6.

2 Related Work

In general, XML benchmarks can be classified into two main categories: 1) Application (*Macro*) benchmarks [3,14,17,20] which are used to evaluate the overall performance of an XML management system. Hence, this kind of benchmarks are not very useful for conducting a detailed assessment of specific aspects of an implementation that need improvement. 2) Micro-benchmarks [7,13] which are designed to assess the performance of specific features of a system. In this section we give a brief overview about the state-of-the-art of XML benchmarks.

XMach-1 [3] is a scalable multi-user benchmark. It is based on a web application and considers text documents and catalog data. It only defines a small number of XML queries that cover multiple functions and update operations for which system performance is determined. The main goal of XMach-1 is to test how many queries per second the query engine can execute. XBench [20] is designed to cover a large number of XML database applications. These applications are characterized by whether they are data-centric or text-centric and whether they consist of a single document or multiple documents. XMark [17] is a single-user benchmark. The database model is based on an internet auction site and consists of one big regularly structured XML document with text and non-text data. The TPOX benchmark [14] is based on a financial application scenario. It is mainly focussed on exercising all aspects of XML database management systems such as: storage, indexing, logging, transaction processing

and concurrency control. The work load of TPOX consists of insert, update and delete operations as well as query operations.

XPathMark [7] is a *Micro* XPath 1.0 benchmark for XMark. It presents a set of XPath queries which covers the major aspects of the XPath language including different axes, node tests, Boolean operators, references, and functions. The targets of XPathMark is to assess the functional completeness, correctness, efficiency and data scalability of XPath implementations. MemBeR [13] is another *Micro-Benchmark* which has a main focus to benchmark the XQuery engines with respect to the efficiency of their implementation to four important XQuery constructs: XPath navigation, XPath predicates, XQuery FLWORS and XQuery Node Construction.

3 Main Aspects of Selectivity Estimation in the XML Domain

When looking for an efficient, capable and accurate selectivity estimation approach for XML queries, there are several issues that need to be addressed. From the experience of our work in [16,18], the major issues of this problem include:

- *It should support structural and data value queries.* In principal, all XML query languages can involve structural conditions in addition to the value-based conditions. Therefore, any complete selectivity estimation system for the XML queries requires maintaining statistical summary information about both of the structure and the data values of the the underlying XML documents. A recommended way of doing this is to separate the structural summaries of the XML document from the data summaries and then group the related data values according to their path and data types into homogenous sets [11].
- *It must be practical.* The performance characteristics of the selectivity estimation process is a crucial aspect for any approach. The selectivity estimation process of any query or sub-query must be much faster than the real evaluation process and the required summary structure(s) for achieving this estimation process must be efficient in terms of memory space consumption.
- *It should be strongly capable.* The standard query language for XML namely XPath and XQuery are very rich languages. It provides a wide set of functions and features such as: structure and content-based search, path expressions, element construction, join, sort, duplicate elimination and aggregation operations. Thus, a *good* selectivity estimation approach should be able to provide accurate estimates for a wide range of these features.
- *It should be composable.* The XML query languages, specially XQuery, are compositional in nature as sub-expressions are combined with each other to form the final query. Hence, a *good* selectivity estimation approach should be able to estimate the selectivity of the final expressions as well as each sub-expressions. This feature is crucial for any cost-based query optimizer to enable a proper selection of cheap execution plans.

- *It must be accurate.* On the one hand, providing an accurate estimation for the query optimizer can effectively accelerate evaluation process of any query. However, on the other hand, providing the query optimizer with incorrect selectivity information will lead the query optimizer to incorrect decisions and consequently to inefficient execution plans.
- *It should be independent.* The selectivity estimation process should be independent of the actual evaluation process and should be applicable with different query engines which are applying different evaluation mechanisms.

4 XSelMark Benchmark Queries

XMark [17] is a well-known benchmark for XML data management. The XMark database is modelling an internet auction web site. XMark comes with an XML generator that produces XML documents according to a numeric scaling factor proportional to the document size. We base the queries of our proposed benchmark on the structure of the XMark document "*auction.xml*" which is described in detail in [17]. The set of queries of our proposed benchmark, XSelMark, represents a mix of XML queries which covers a wide set of the major selectivity estimation aspects in the domain of XML queries. They are designed in a way to allow a *realistic* assessment for the advantages and shortcomings of the proposed XML selectivity estimation approaches and to identify their respective impact. The set of queries are expressed using two standard XML query languages XPath and XQuery. Due to lack of space, we do not present the source code of some queries. The source code of all queries can be downloaded from the benchmark Web site at [1]. The queries are grouped under subsection headings which indicate the feature to be tested.

4.1 Group 1: Path Expressions

Q1) Path expression with non-recursive axes: Find the names of all persons. `/site/people/person/name/text()`

Non-recursive XPath axes are *child*, *parent*, *attribute*, *following-sibling* and *preceding-sibling*.

Q2) Path expression with recursive axes: Find all description nodes descendant of all item nodes. `/site//item//description`

Recursive XPath axes are *descendant*, *descendant-or-self*, *ancestor* and *ancestor-or-self*.

Q3) Path expression with wild cards: Return the item subtrees of all regions. `/site/regions/*/item/*`

Q4) Path expression with ordered-based axes: Return the description nodes which are following the tags with the name *closed_auction*.

`/site//closed_auction/following::description`
where ordered-based axes are *following*, *following-sibling*, *preceding* and *preceding-sibling*. Supporting such type of queries requires capturing specific statistical information about the order of the elements in the XML documents.

Q5) Branching XPath Expressions: Return the names of all persons who have age information in their profiles. `/site//person[profile/age]/name`

4.2 Group 2: Twig Expressions

Q6) Simple twig expression: Return the names and descriptions of all items.

```
for $b in //item return ($b/name,$b/description)
```

Q7) Twig expression with element construction: Return the restructured results of the names and descriptions of all items.

```
for $b in //item
return
<Result>
  <name>{$b/name}</name>
  <description>{$b/description}</description>
</Result>
```

4.3 Group 3: Predicates

The estimation of predicate selectivity is a well-known problem in database theory and practice. Most common solutions of this problem rely on histograms for capturing the distribution of data values, and on the use of the uniform distribution when nothing is known about the data involved in the predicate. In the context of XML, predicate selectivity estimation poses new challenges such as: 1) The predicates can be structural-based as well as value based. 2) Positional predicates represent a special form of predicates over the order information of the elements in the XML document. 3) XML elements are usually distributed in a non-uniform way, hence assuming a simple uniform distribution of the elements structure may lead to many potential estimation errors especially when the operated sequence of nodes are constructed by merging nodes from different groups of data elements.

Q8) Positional Predicates: Return the third bidder of each open auction.

Q9) Equality Predicates: Return the closed auctions with price equal to 40.

Q10) Range Predicates: Return the closed auctions with price less than 40.

Q11) Conjunctive/Disjunctive Predicates: Return the closed auctions with price greater than 40 and less than 100.

Q12) Predicates with merged nodes from different paths: Return the african and asian items with id value greater than 'item100'.

```
for $b in (/site//africa/item, /site//asia/item)
where data($b/@id) > 'item100'
return $b
```

An accurate estimation of such query should consider the different distribution for the data values nodes resulting from each different path expression as well as the percentage of each path in constructing the nodes of the operated sequence.

Q13) Predicates with merged nodes from different paths and hybrid nature: Return the price nodes and quantity nodes with value greater than 100.

```
for $b in (/site//price,/site//quantity)
where data($b) > 1 and data($b) > 100
return $b
```

This query is more challenging than the previous one because the resulting nodes of the operated sequence are representing completely different data items (*price, quantity*) which may have totally different distributions for their data values.

Q14) String Predicates: Return all persons with id value greater than "person200".

4.4 Group 4: Value-Based Joins (Theta Joins)

This group of queries assess the ability and the accuracy of the selectivity estimation approaches on effective and accurate dealing with value-based join operations between the data values of XML nodes.

Q15) Value-based join instances where the values of each operand are constructed by path expression: Return all pairs of increase value and price value where the increase value is greater than the price value.

Q16) Value-based join instances where the values of one operand are constructed by path expression and the values of the other operand are constructed by path expression manipulated with arithmetic expression: Return all pairs of increase value and price value where the increase value is greater than the price value multiplied by 2.

```
for $x in /site//increase, $y in /site//price
```

```
where data($x) > data($y) * 2
```

```
return <pair>{$x,$y}</pair>
```

Q17) Equi-Joins of data values: Return all pairs of increase value and price value where the increase value is equal to the price value.

4.5 Group 5: Arithmetic and Comparison Operations over Data Value Statistics

This group of queries assess the ability of the selectivity estimation approaches on their ability of not only being able to capture summary information about the data values of the XML elements but also on their ability of applying arithmetic and comparison operations over these summary information in a consistent and accurate way which does not hurt the quality of the selectivity estimation results.

Q18) Arithmetic over Data Value Statistics 1: Return all pairs of increase value and price value where the sum of the increase value and the price value is greater than 100.

```
for $x in /site//increase, $y in /site//price
```

```
where data($x) + data($y) > 100
```

```
return <pair>{$x,$y}</pair>
```

Q19) Arithmetic over Data Value Statistics 2: Return all pairs of increase value and price value where the sum of the increase value and the price value is equal to 100.

Q20) Arithmetic and Comparison operations over Data Value Statistics 3: Return all triples of increase value, price value and income where the sum of the increase value and the income value is greater than the sum of the price value and the income value.

4.6 Group 6: Nested Expressions

XQuery, as with many other XML query languages such as SQL/XML [4], is a free nesting language, where nested queries can be used for many targets such as reshaping elements or computing aggregate values. Since the result of nested queries may be the input for navigational or filtering operations in the outer query, predicting the size of nested query results will require building *on-the-fly* statistics about these intermediate results.

Q21) Let - Aggregates: Return the names of persons and the number of items that they bought.

```
for $p in /site/people/person
let $a :=
  for $t in /site//closed_auction
  where $t/buyer/@person = $p/@id
  return $t
return <item>
  <person>{$p/name/text()}</person>
  <count>{count($a)}</count>
</item>
```

Q22) Predicates with values constructed by aggregate function: Return the open auctions with sum of bidder increases that are greater than 1000.

```
for $b in /site/open_auctions/open_auction where
sum(data($b/bidder/increase)) > 1000 return
<increase>{$b}</increase>
```

4.7 Group 7: Data Dependent Estimations

This group of queries requires capturing additional specific forms of summary information about the data values of the underlying XML documents.

Q23) Full Text Search: Return the names of all items whose description contains the word "gold".

Q24) Distinct Operator: Return the distinct price values.

Q25) Existential Document Order: Return the open auctions where a certain person issued a bid before another person.

```
for $b in /site/open_auctions/open_auction
where
  some $pr1 in $b/bidder/personref[@person = "person20"],
  $pr2 in $b/bidder/personref[@person = "person51"]
  satisfies $pr1 << $pr2
return <history>{$b}</history>
```

5 XML Selectivity Estimation: An Assessment of the State-of-the-Art

The work of Aboulmaga et al. [2] is considered to be the first to deal with the selectivity estimation of simple path expressions. They presented two different

techniques: *path tree* and *Markov table* which capture any distinct path of length up to m and its selectivity. XPathLearner system [12] has employed the same summarization and estimation techniques presented in [2] with two main modifications: 1) it gathers and refines the required statistical information in an on-line manner from query feedbacks. 2) it supports the handling of predicates by storing statistical information for each distinct tag-value pair in the source XML document. In [19] Wang et al. have proposed a special histogram structure for the selectivity estimation of XPath queries in a dynamic context named as *Bloom Histogram*. A bloom histogram H is constructed by sorting the frequency values of the distinct paths in XML data and then grouping the paths with similar frequency values into buckets. In [10], Li et al. have described a histogram-based framework for estimating the selectivity of XPath expressions with a main focus on the order-based axes (following, preceding, following-sibling, preceding-sibling).

In [6] Fisher et al. have proposed the *SLT* XML tree synopsis. The main idea of this summary synopsis is to remove the repeated patterns in the XML tree and to replace the multiple occurrences of equal subtrees by pointers to a single occurrence of the subtree. In [15] Polyzotis et al. have proposed the XCluster synopses as a clustering-based framework that can capture the key correlations between and across structure and values of different types. XCluster employs the well-known histogram techniques for summarizing the data values. This approach can support twig queries with predicates. However, the authors did not mention how XCluster can be extended to deal with more complicated query situations such as value-based join operations and nested expressions.

The work of [16,18] has described the design and implementation of a relational algebraic based framework for estimating the selectivity of XQuery expressions. In this approach, XML queries are translated into relational algebraic plans [9]. Summary information about the structure and the data values of the underlying XML documents are kept separately. Then by exploiting the relational algebraic infrastructure, the special properties of the generated algebraic plans, the summary information and a set of inference rules, the relational estimation approach is able to provide accurate selectivity estimations in the context of XML and XQuery domains.

5.1 Features Assessment

We used the set of XSelMark benchmark queries for an initial assessment of the supported features by the state-of-the-art. Table 1 lists the set of queries supported by each approach. The assessment has shown some interesting preliminary results: 1) Most of the selectivity estimation approaches [8,10,12,19] are limited on their abilities to support only small subsets of the XML query languages. They are only able to deal with structural XPath queries. 2) The two synopses of [10,6] are the only two synopses which are able to support the selectivity estimation of order-sensitive XPath axes. 3) The approaches of [15,16] cover a wider range of the XML query features. The synopsis of [15] is the only one which is able to deal with the estimation of full text search queries while

Table 1. An assessment of the state-of-the-art of the selectivity estimation approaches

	XPath- Learner [12]	Path-Order Histogram [10]	Bloom Histogram [19]	SLT Gramar [6]	XCluster [15]	Relational Alg. Est. [16,18]
Q1	X	X	X	X	X	X
Q2	X	X	X	X	X	X
Q3	X	X	X	X	X	X
Q4	-	X	-	X	-	X
Q5	X	X	-	X	X	-
Q6	-	-	-	-	X	X
Q7	-	-	-	-	X	X
Q8	-	-	-	X	-	X
Q9	X	-	-	-	X	X
Q10	X	-	-	-	X	X
Q11	X	-	-	-	X	X
Q12	-	-	-	-	-	X
Q13	-	-	-	-	-	X
Q14	-	-	-	-	X	X
Q15	-	-	-	-	-	X
Q16	-	-	-	-	-	X
Q17	-	-	-	-	-	X
Q18	-	-	-	-	-	X
Q19	-	-	-	-	-	X
Q20	-	-	-	-	-	X
Q21	-	-	-	-	-	X
Q22	-	-	-	-	-	-
Q23	-	-	-	-	X	-
Q24	-	-	-	-	-	-
Q25	-	-	-	-	-	-

[16] is able to uniquely deal with many of the features of XQuery languages such as join operation and different type of predicates.

6 Conclusion

Several research efforts have been invested on designing Macro-Benchmarks to assess the overall performance of XML data management systems. There is currently a big demand for several Micro-Benchmarks which assess specific aspects in the XPath, XQuery and XML management system domains. Several research efforts have proposed different selectivity estimation approaches in the XML domain. Due to the lack of a *suitable* benchmark, it was difficult to assess, evaluate and compare these approaches and in order to get a clear view about the state-of-the-art. We proposed XSelMark as a Micro-Benchmark to assess the the selectivity estimation approaches of XML queries. An initial assessment for the features and capabilities of the current approaches has shown that most of them are limited to supporting the estimation of the structural XPath queries. Hence, several avenues for further research and development are still widely open in this domain to provide accurate, capable and complete frameworks aligned with the rich querying capabilities of the standard XML query languages. We believe that XSelMark is useful for both researchers and developers. It identifies the major aspects of selectivity estimation of XML queries, helps researchers to discover the strengths and weaknesses of the current approaches and provides the researchers and developers with a clearer view of developing more enhanced mechanisms of selectivity estimation of XML queries. As a future work, we are planning to use XSelMark to perform more detailed assessment of the selectivity estimation approaches of XML queries in terms of their accuracy, performance and memory requirements.

References

1. XSelMark: A Micro-Benchmark of Selectivity Estimation of XML Queries, <http://XSelMark.sourceforge.net/>
2. Aboulmaga, A., Alameldeen, A., Naughton, J.: Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. In: VLDB (2001)
3. Böhme, T., Rahm, E.: XMach-1: A Benchmark for XML Data Management. In: BTW (2001)
4. Eisenberg, A., Melton, J.: Advancements in SQL/XML. SIGMOD Record 33(3), 79–86 (2004)
5. Fernández, M., Malhotra, A., Marsh, J., Nagy, M., Walsh, N.: XQuery 1.0 and XPath 2.0 Data Model (XDM). World Wide Web Consortium Proposed Recommendation (November 2006), <http://www.w3.org/TR/xpath-datamodel>
6. Fisher, D., Maneth, S.: Structural Selectivity Estimation for XML Documents. In: ICDE (2007)
7. Franceschet, M.: XPathMark: An XPath Benchmark for the XMark Generated Data. Database and XML Technologies (2005)
8. Freire, J., Haritsa, J., Ramanath, M., Roy, P., Siméon, J.: StatiX: making XML count. In: SIGMOD (2002)
9. Grust, T., Sakr, S., Teubner, J.: XQuery on SQL Hosts.. In: VLDB (2004)
10. Li, H., Lee, M., Hsu, W., Cong, G.: An Estimation System for XPath Expressions. In: ICDE (2006)
11. Liefke, H., Suciu, D.: XMill: An efficient compressor for XML data. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) SIGMOD (2000)
12. Lim, L., Wang, M., Vitter, J., Parr, R.: XPathLearner: An On-line Self-Tuning Markov Histogram for XML Path Selectivity Estimation. In: VLDB (2002)
13. Michiels, P., Manolescu, I., Miachon, C.: Toward microbenchmarking XQuery. Information System 33(2) (2008)
14. Nicola, M., Kogan, I., Schiefer, B.: An XML transaction processing benchmark. In: SIGMOD (2007)
15. Polyzotis, N., Garofalakis, M.: XCluster Synopses for Structured XML Content. In: ICDE (2006)
16. Sakr, S.: Cardinality-Aware and Purely Relational Implementation of an XQuery Processor. PhD thesis, University of Konstanz (2007)
17. Schmidt, A., Waas, F., Kersten, M., Carey, M., Manolescu, I., Busse, R.: XMark: A Benchmark for XML Data Management. In: VLDB (2002)
18. Teubner, J., Grust, T., Sakr, S., Maneth, S.: Dependable Cardinality Forecasts for XQuery. In: VLDB (2008)
19. Wang, W., Jiang, H., Lu, H., Xu Yu, J.: Bloom Histogram: Path Selectivity Estimation for XML Data with Updates. In: VLDB (2004)
20. Yao, B., Özsu, T., Keenleyside, J.: XBench - A Family of Benchmarks for XML DBMSs. In: VLDB Workshop (2003)

A Method for Semi-automatic Standard Integration in Systems Biology

Dagmar Köhn¹ and Lena Strömbäck²

¹ Universität Rostock, Institute for Computer Science, Chair for
Database and Information Systems, Germany
`dk103@informatik.uni-rostock.de`

² Linköpings Universitet, Department of Computer and Information Science, Sweden
`lestr@ida.liu.se`

Abstract. The development of standards for biological pathways has led to a huge amount of model data stored in a variety of different formats represented in XML (e.g. SBML) or OWL (e.g. BioPAX). There is an urgent need for the conversion of data between the formats, but the fact that the transformation is hard to realize hampers the integration of data in the area. This article proposes a general, semi-automatic solution by transforming XML Schema based data into an OWL format. Biologists will be supported in querying data of any format and comparing different data files or schemas to each other using OWL as a common format for matching. Additionally, a backwards transformation to XML Schema is provided. The paper presents a first architectural approach and its prototype implementation. The evaluation showed that the approach is promising.

1 Motivation

Advanced experimental methods within biology rapidly generate new knowledge about how genes, proteins, and other substances interact. A major goal within the area is a complete description of the protein interaction network underlying cell physiology [9]. Moreover, the understanding of genetic networks and protein pathways is recognized as being a crucial part of future genomics research [4]. Reusable software modules, new ontologies [12], and improved technologies for database and knowledge management [5] are the key players for solutions to these challenges in the future. To fulfill the biological vision, emphasis has been put on representation formats that allow for exchange and integration of knowledge. An investigation during spring 2006 [18] showed that there were more than 80 defined formats for the representation of pathway models or related information. Of those, 14 are in common use. In most cases, the standards are defined using the eXtensible Markup Language (XML, [20]), but some of them also use the Web Ontology Language (OWL, [23]).

The different communities in Systems Biology have been struggling with the fact that there exist diverse standards that do overlap partly, but at the same

time do have their own specific strengths. The importance of standard integration becomes more and more obvious as the number of interdisciplinary research projects increases constantly [17,19]: The projects become more comprehensive and different formats have to be used for the modulation and analysis of parts of the system. To provide an interface between the standards, a transformation of one standard into the others has to be made available. Some communities already provide converters, for example on www.sbml.org for SBML models. However, the main drawback of this solution is the hard encoded transformation: As soon as a new version of a standard is available, the converter has to be adapted. Another problem is that a great number of converters are needed and have to be maintained for all the combinations of pathway standards to be supported. To overcome those problems, the approach taken in this work provides a more general solution. The described architecture could, if fully investigated, be a tool that provides both semi-automatic conversion between formats and re-use of old conversions when new versions of a representation format appear – which is needed but not available with current technology.

This paper describes an architecture for semi-automatic integration of standards. First, it shortly describes XML Schema and OWL model. Afterwards, we present the implemented architecture which builds on existing technology, such as schema matchers and converters from XML to OWL. The comparison of XML and OWL has already been discussed in several fields of research before [2,7,8,16] and the solution introduced here re-uses the mapping recommendation of [7] with an improved implementation that is adjusted to the needs of biological data transformation. In a similar way, OWL matching tools (e.g. SAMBO [13], Protégé [14], COMA++ [1]) are re-used. We describe the results of our evaluations which showed that our approach is feasible.

2 Standards and Standard Definition Formalisms

XML Schema [22] defines the formats' structure and supports the development of an XML model which is *syntactically* restricted to the occurrence of certain elements. When an XML instance fulfills the requirements of the XML Schema it is valid and complies with the according standard.

As an example, we consider a small extract of an SBML file taken from the Reactome Database [10] (listing 1.1¹): The XML Schema defines elements of simple (e.g. `speciesReference`, ll 3-4), compound, or complex types (e.g. `reaction`, ll 1-13, consisting of `listOfReactants` (ll 2-5), `listOfProducts` (ll 6-9) and `listOfModifiers` (ll 10-12)). In addition, the definition of attributes (e.g. `name` and `id` for `reaction`) is possible. The nesting of elements sets up the document structure starting from a single root element.

¹ EGFR stands for “Epidermal Growth Factor Receptor”. The Reactome IDs correspond to the following molecules: REACT_2812_1 $\hat{=}$ ATP; REACT_9820_1 $\hat{=}$ EGF:EGFR dimer; REACT_2741_1 $\hat{=}$ ADP; REACT_9673_1 $\hat{=}$ EGF:Phospho-EGFR dimer.

```

1 <reaction name="EGFR_autophosphorylation" id="REACT_9388_1">
2   <listOfReactants>
3     <speciesReference species="REACT_2812_1" />
4     <speciesReference species="REACT_9820_1" />
5   </listOfReactants>
6   <listOfProducts>
7     <speciesReference species="REACT_2741_1" />
8     <speciesReference species="REACT_9673_1" />
9   </listOfProducts>
10  <listOfModifiers>
11    <modifierSpeciesReference species="REACT_9820_1" />
12  </listOfModifiers>
13 </reaction>

```

Listing 1.1. Code snippet from the SBML XML Schema for the EGFR model

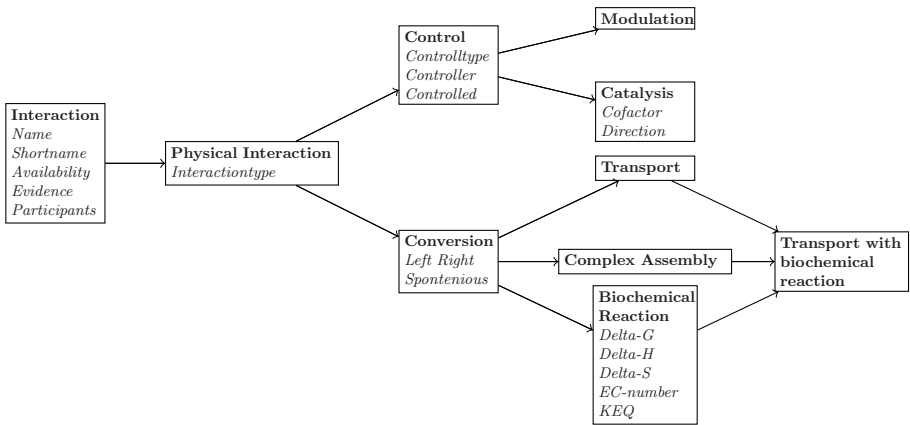


Fig. 1. Extract from the BioPAX model

OWL specifies an OWL model of *semantic* concepts and verifies OWL documents against it. This is exemplified in Figure 1 which shows part of the BioPAX specification:

An OWL model defines `owl:Classes` independently, which are put into IS_A relations using `owl:SubclassOf` constructs (e.g. `Conversion` is subclass of `PhysicalInteraction`). The attributes (e.g. `Left` for conversion) are defined by using `owl:DatatypeProperty` and `owl:ObjectProperty`.

As OWL concentrates on the semantics behind terms it is very specific in the definition of relations between classes, i.e. `subClassOf`, `intersectionOf`, and `unionOf`. On the contrary, the only way of defining semantics in XML Schema is the hierarchical composition of elements and type definitions which lead to an *implicit* definition of semantics, i.e. the semantic of an element cannot be detected automatically by a machine, but only from the elements' context. For example, the XML element `name` in the SBML example (listing 1.1, line 1) is the name of a reaction. The same syntax is also used for the name of other entities, such as species names. Researchers argue that a general transformation between

XML Schema and OWL model is not feasible, as XML Schema does “not contain any semantic information, whereas OWL is derived from RDF, which is meant to express semantic relations between elements” [7]. However, we agree that XML Schema contains little implicit semantics “conveyed on the basis of a shared understanding derived from human consensus” [11], which makes a translation possible.

3 Integration of XML-Based and OWL-Based Standards

In this work, the focus is on a *generic*, but *semi-automatic* solution for the matching of different pathway standards supporting re-usability and comparability of biochemical models defined in XML- and OWL-based formats. The general architecture can be divided into the following steps:

1. Provide a schema definition (e. g. the SBML Schema), if the initial point was an instance file (e. g. an SBML model)
2. Transform the XML Schema into an OWL model representation
3. Repeat 1 and 2 for all standards that are to compare
4. Match the (created) OWL models on OWL level
5. Use the matching correspondences to either form a joint format containing all the information of both starting schemas, or to assign data of the source document to the target document

To illustrate the use of the architecture, we could imagine the integration of PSI MI information with an existing BioPAX model. This task can be realized by transforming the according XML Schema version of the PSI MI standard into an OWL model representation (step 2) and performing a matching of the PSI MI OWL model and the BioPAX OWL model (step 4). For corresponding parts, the existing data of both models can then be integrated and merged into a comprehensive model (step 5). A second use case is the transformation of a given data set in a standard format into another standard format, e. g. adding an SBML data source to a BioPAX data repository. In that case, the (SBML) XML Schema of the instance file is transformed into an OWL model presentation (step 2) and then is matched with the (BioPAX) OWL model (step 4). A valid OWL instance file in BioPAX format is created and the appropriate data provided by the SBML instance file is integrated (step 5).

The steps mentioned above are realized in the architecture as follows: During the *transformation step*, an existing XML Schema is transformed into an equivalent OWL model considering the XML Schema concepts listed in table 1. The transformation keeps naming and structure of the original models to provide best mapping results during the matching process. In particular, the hierarchy of the XML Schema remains unchanged. Concepts such as cardinalities or data models (differences between **choice**, **sequence** and **all**) are of minor interest though. No additional names or ids should be generated in order to avoid a distortion of matching results later on. The result of the transformation is a valid OWL model that can be read by existing OWL tools.

Table 1. XML Schema – OWL model transformation rules

XML Schema	OWL model
xsd:complexType	owl:Class
xsd:element	owl:DatatypeProperty owl:ObjectProperty
xsd:attribute	owl:DatatypeProperty
xsd:import	reference in name attribute
xsd:enumeration	rdfs:comment
xsd:pattern	rdfs:comment
xsd:annotation	rdfs:comment

To get back from the created OWL model to the XML based representation, the whole transformation XML Schema $\rightarrow$ OWL model is recorded (*recall transformation*) using the XML Path Language (XPath [21]). By storing XPath expressions during the transformation step, the XML Schema can be re-created from the generated OWL model as each part of the created OWL model and its equivalent in the according XML Schema can be unambiguously identified.

In a *matching step*, the created and/or the original models are matched. A matching “takes two schemas S_1 and S_2 as input and returns a mapping between those two schemas as output” [15]. The resulting mapping then defines relations between the two schemas and therefore allows for a comparison of common parts in both schemas. There exist a number of matching algorithms [6] which use different techniques, for example syntactical matching or structure matching. Since perfect matching involves semantic interpretation, a semi-automatic approach has to be used here.

The three modules mentioned constitute the essential functionality of the approach to allow for a first evaluation. They have been realized as a prototype implementation. The recall transformation has been implemented, but not yet integrated into the system.

4 Evaluations

The evaluation of the architecture addresses the following questions: How much information is lost during the transformation step? How good is the transformation algorithm compared to other XML to OWL converters? How satisfying is the result of the transformation? Basis for all evaluations was a set up list of 17 correspondences between the three most commonly used standards (SBML, PSI MI, BioPAX).

Most important is the estimation *how much information the transformation step loses*. As COMA++ supports a matching on XML Schema and on OWL model level, it has been used to compare the matching results for different standards on both XML and OWL level. Column 1 and 2 in table 2 show the summary of results for the exemplified matching of SBML $\rightarrow$ PSI MI on XML Schema and OWL model level: COMA++ finds six of the 17 suggestions on

Table 2. Correspondences on XML Schema and OWL level found by COMA++ using the recall transformation and the XML2OWL tool [3] to match SBML and PSI MI

criterion	Schema level	OWL level	
	original model	recall transformation	XML2OWL
total suggestions	6	8	13
found proposed correspondences	1	2	2
additional correct suggestions	2	3	4
Recall(%)	5.88	11.76	11.76
Precision(%)	16.67	25.00	15.38
RelRecall(%)	15.87	25.00	28.57
RelPrecision(%)	50.00	62.50	46.15
Fmeasure _{rel}	0.24	0.36	0.35
overall _{rel}	0.31	0.50	0.57

XML Schema level, and eight of the 17 suggestions on OWL model level. All matching suggestions found on XML Schema level are also detected on OWL model level. In contrast to the Recall, the Relative Recall (RelRecall) takes into account additionally found correspondences which were not originally in the list of expected correspondences. Here, two additional correspondences were found on XML Schema level and three additional on OWL model level. The fact that the recall in all cases is much lower than the relative recall shows that the list of proposed correspondences set up to evaluate the architecture misses some “obvious” suggestions and that some of the proposed correspondences might have been too specific to be found by matching algorithms. Looking at the precision measures, the values are close for both attempts. A precision of around 50% is at any rate acceptable for the matching. Finally, it should be mentioned that SBML and BioPAX do have more overlaps than SBML and PSI MI. Unfortunately, the comparison of matching results on both levels could not be done using BioPAX, as it is an OWL-based standard that does not provide an XML Schema.

Regarding the *quality of the transformation step* see table 2, column 2 and 3 for a comparison of the implemented transformation with the existing XML2OWL tool [3] – a promising approach that also supports the conversion of XML Schema to OWL model [2,3]. Looking at the result model one gets from the XML2OWL tool, a lot of additional ids and names are created, so that the matcher finds a greater number of matching correspondences (13), which – on the other hand – are often not correct (only six of the 13 correspondences being reasonable). As a consequence, the relative precision is much lower for that approach.

The correspondences between the (created) OWL models are found during the *matching step*. The three matching tools COMA++, Protège and SAMBO have been evaluated (compare results in table 3) and COMA++ and SAMBO have been identified as most suitable. For example, both found a reasonable number of matching suggestions for the SBML → PSI MI matching. SAMBO gave 12 correspondences using the prototype implementation, and COMA++ returned five correspondences. Some matching tools returned better results for the matching of certain pairs of standards than for others, e.g. the matching of

Table 3. Correspondences for Different Matchings using SAMBO (S), PROMPT (P), and COMA++ (C): x/x results from prototype/XML2OWL approach, * matching not executed, $\sum_{all}$ total number of matching results, $\sum_{reas}$ correct correspondences

	SBML $\rightarrow$ PSI MI			SBML $\rightarrow$ BioPAX			PSI MI $\rightarrow$ BioPAX		
	S	P	C	S	P	C	S	P	C
$\sum_{all}$	12/8	3/*	8/13	0/0	1/*	7/95	2/0	6/0	0/0
$\sum_{reas}$	12/8	3/*	5/6	0/0	1/*	2/5	3/0	6/0	0/0

SBML $\rightarrow$ PSI MI worked best using SAMBO (five of the six proposed suggestions found correctly), whereas the SBML $\rightarrow$ BioPAX matching was performed best by COMA++ (two of the 5 proposed suggestions found correctly). As mentioned before, the high number of matchings in COMA++ using the XML2OWL transformation approach (see table 3, $\sum_{all}$) results from artificially generated IDs, which cannot be considered as valid correspondences. Please note that the evaluations only consider automatically gained results which have to be enhanced by semi-automatic alignment by the users in order to improve the results.

5 Conclusions and Future Directions

In this paper, an architecture for the *version-independent integration of biological data* stored in various formats has been proposed. The evaluations in the last section of the paper have shown that the approach is promising. Results can be further improved in the future by enhancing both the matching algorithms and the transformation of XML Schema into OWL model. Also, user interaction needs to be implemented. The conclusion drawn from the evaluation section is that the architecture can support the user during his work on different standards, especially during the integration process of data, and during the search for data available in different standards. Future tasks include the complete implementation of the architecture and a more comprehensive evaluation.

Acknowledgements. We acknowledge the financial support of the Center for Industrial Information Technology, the German Academic Exchange Service (DAAD), and the German Research Foundation (DFG). The first author is a member of the DFG research training school “dIEM oSiRiS” (GRK 1387). The second author is member of the EU Network of Excellence REWERSE (Sixth Framework Programme project 506779). We also thank Patrick Lambrix and He Tan for comments on the work and support with the SAMBO system.

References

1. Aum Mueller, D., Do, H.-H., et al.: Schema and ontology matching with coma. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, June 2005, pp. 906–908. SIGMOD, ACM Press, New York (2005)

2. Bohring, H., Auer, S.: Mapping xml to owl ontologies. In: Proceedings of the 13. Leipziger Informatik-Tage, vol. 72, pp. 147–156 (2005)
3. Brischniz: The xml2owl demonstration platform (last checked: 2008-06-10), <http://xml2owl.sourceforge.net/>
4. Collins, F.S., Green, E.D., et al.: A vision for the future of genomics research. *Nature* 422, 835–847 (2003)
5. Davidson, S., Overton, C.G., et al.: Challenges in integrating biological data sources. *Journal of Computational Biology* 2(4), 557–572 (1995)
6. Doan, A., Halevy, A.: Semantic integration research in the database community: A brief survey. *AI Magazine* 26 (March 2005)
7. Ferdinand, M., Zirpins, C., et al.: Lifting xml schema to owl. In: Web Engineering - 4th International Conference, ICWE 2004, pp. 354–358. Springer, Heidelberg (2004)
8. Gibbons, F.: The psi mi to biopax converter approach, from an email-conversation (last checked 2008-06-10) (May 2006), <http://llama.med.harvard.edu/~fgibbons>
9. Hermjakob, H., Montecchi-Palazzi, L., et al.: The hupo psi's molecular interaction format - a community standard for the representation of protein interaction data. *Nature Biotechnology* 22(2), 177–183 (2004)
10. Joshi-Tope, G., Gillespie, M., et al.: Reactome: a knowledgebase of biological pathways. *Nucleic Acids Research* 33, D428–D432 (2005)
11. Shengping, L., Jing, M., et al.: Xsdl: Making xml semantics explicit. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372, pp. 64–83. Springer, Heidelberg (2005)
12. Lambrix, P.: Ontologies in Bioinformatics and Systems Biology. In: Artificial Intelligence Methods and Tools for Systems Biology, October 2005. *Computational Biology*, vol. 5, pp. 129–147. Springer, Heidelberg (2005)
13. Lambrix, P., He, T.: Sambo -a system for aligning and merging biomedical ontologies. *Journal of Web Semantics, Special issue on Semantic Web for the Life Sciences* 4, 196–206 (2006)
14. Fridman Noy, N.: PROMPT (last checked 2008-06-10) (2000), <http://protege.stanford.edu/plugins/prompt/prompt.html>
15. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 334–350 (2001)
16. Reif, G.: The weesa project (last checked 2008-06-10) (2005), <http://www.infosys.tuwien.ac.at/weesa/>
17. Rübenacker, O., Moraru, I., et al.: Kinetic modeling using biopax ontology. In: IEEE International Conference on Bioinformatics and Biomedicine, pp. 339–348 (2007)
18. Strömbäck, L., Hall, D., et al.: A review of standards for data exchange within systems biology. *Proteomics* 7(6), 857–867 (2007)
19. Uhrmacher, A.M., Rolfs, A., et al.: Dfg research training group 1387/1: diem osiris - integrative development of modelling and simulation methods for regenerative systems. *it - Information Technology* 49(6), 388–395 (2007)
20. W3C. Extensible Markup Language (1999), <http://www.w3.org/XML>
21. W3C. XML Path Language (November 1999), <http://www.w3.org/TR/XPath>
22. W3C. Xml schema (May 2001), <http://www.w3.org/XML/Schema>
23. W3C. Web Ontology Language (2004), <http://www.w3.org/2004/OWL/>

FDBMS Application for a Survey on Educational Performance

Livia Borjas¹, Alejandro Fernández¹, Jorge Ortiz¹, and Leonid Tineo²

¹ IUT Federico Rivero Palacios, Departamento de Informática,
Carretera Panamericana, Caracas, Venezuela
{lborjasIUT, afernandezIUT, jortizIUT}@gmail.com

² Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela
leonid@usb.ve

Abstract. Important efforts have been done in providing fuzzy querying capabilities. Nevertheless, at present time too few real life systems are taking advantage of these works. We present here an application for reviewing results of a periodical student opinion survey on educational performance of professors. This application has been developed using SQLfi Fuzzy DBMS.

Keywords: SQLf, Fuzzy Querying, Database Application.

1 Introduction

Emerging technologies that support decision-making processes are the tip of the iceberg under review. This is the case of queries involving user preferences whose definition and use lead to discriminated answers. Fuzzy Sets theory applied to data has originated the so-called Fuzzy Databases, which allow [3]: — Imprecise data in a modeled reality, such as: “*Leo’s age is young and his performance is high*”. Here *young* and *high* are imprecise values for attributes age and performance. — Fuzzy queries over precise data, such as: “Find the *most popular* professors according to their students opinion”. Here *most popular* express a user preference in the requirement. Some works are intended to provide fuzzy database systems: — OMRON [12], a fuzzy information retrieval library that contains a SQL variant with fuzzy logic on traditional databases; — FQUERY [10] is an effort that adds fuzzy query functionality over a small database management system; — FSQL [5] is an extension of SQL for fuzzy relational databases; — SQLf [2][7][8], is a fuzzy logic based extensions of SQL over relational databases that includes more querying structures than other fuzzy database languages. Based on it, SQLfi, a Fuzzy Database Management System has been implemented with Oracle 9i Java [4].

Some work has been made in fuzzy querying built for specific applications[3]. Nevertheless too few applications have been made with support on FDBMS, we can enumerate [1][6][9][12]. We present here a complete, real and practical example of fuzzy queries on relational databases using SQLfi. The study case is Professor Educational Performance Survey at Universidad Simón Bolívar, Caracas Venezuela. They

have a periodical survey of student opinion on the educational performance of the professors. All this survey obtained data is stored in a relational database has not been sufficiently taken advantage of. This database could be used to the support of some decision making using fuzzy querying capabilities.

Rest of this paper is organized as follows: — Section 2 gives an Exposition of the Problem solved by the application. — Section 3 briefly shows the Fuzzy Querying Background. — Section 4 describes the application in terms of Fuzzy Queries Design. — Section 5 presents User Interfaces provided by the application for the specification of fuzzy queries. — Section 6 points out the Concluding Remarks of this work.

2 Exposition of the Problem

Universidad Simón Bolívar (USB) has a relational database with data collected from the survey of student opinion on teaching performance. It consists of an item list of professor aptitudes, attitudes and facts that student evaluates according own appreciation. USB has a matrix organization: Departments are conformed by professors, while students are assigned to Coordinations of Studies. Students may be enrolled to courses associated to their corresponding studies program that are offered by different departments. Professors are evaluated for contract renovation, promotion and/or programs of stimulus to the outstanding work. The academic department chief and the different served study programs coordinators must give a substantial opinion about the professor performance. In the survey process, students give appreciation of their enrolled courses professors. Each professor is notified the result about his own performance.

At present time, this survey database has not been sufficiently taken advantage of thus the survey is in risk of lost its significance. Moreover, some decisions are made without the suitable support. Each academic department chief is notified of the survey result about each one of the professors in the department. Nevertheless the coordinator of studies program has no information from the survey result. Stimulus programs committees have no access to the survey result but with a copy that must be given by the professor.

We present a first approach to the solution of this problem. We have developed an end-user application for formulation of fuzzy queries in this study case. With this tool academic department chiefs, program studies coordinator and stimulus programs committees would be able to review the survey of student opinion in a rather semantic way, using natural language terms with a fuzzy based intrinsic interpretation. These terms use makes the application closer to what it is to be reflected in a professor performance review, thus accurately and individually orienting the education, training and development needs of the professor headcount.

3 Fuzzy Querying Background

Fuzzy database are currently in the spotlight due to the significant advantages they offer regarding query results: flexibility that cannot be achieved through classic database management systems (CDBMS). This flexibility feature adds elements that contribute to

decision-making processes. In a classic query, only rows perfectly meeting condition are retrieved. While in a fuzzy query frontier rows are included in the answer that is rows that are close to meet conditions, within an accepted range established by the search criteria. Through these fuzzy database management systems (FDBMS), we are able to define our own search criteria based on linguistic terms that are familiar for final user in its competence domain. Meaning for these terms is user dependent and is based on fuzzy set theory. Therefore, the answer of a fuzzy query becomes a fuzzy relation or fuzzy answer set. Answers are discriminated by their satisfaction degree to querying condition. This degree is a real number in $(0,1]$ set, where 1 represents total satisfaction, and 0 represents total exclusion, values in the middle are less rigid they are partially included or satisfy the query with a lower reliability level. In order to avoid super-populated answers and undesired low quality answers, the fuzzy query may be calibrated with a threshold for satisfaction degrees.

Example 1: Suppose we want to find young people from PERSON relation in Table 1. The term young would not be accepted by a CDMBS, we would have to say, for example, that: “young persons are those between 15 and 25 years of age” obtaining CLASSIC query result in Table 1. On the other hand, a FDBMS might handle the term young by creating a predicate that establishes a flexible range to identify a young person as in Fig. 1, we restrict the query with the threshold 0.5, obtaining FUZZY query result in Table 1.

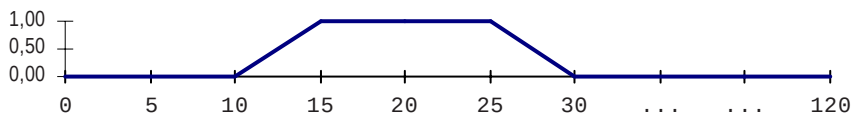


Fig. 1. Membership function of the fuzzy predicate young for the Example 1, which semantics is as follows: “anyone starts to become a young since turns 10 till reaches 15, then continues to be young between 15 and 25 years of age, and definitively ceases to be young after turning 30.”

We can see in previous example that different results are obtained when using a CDBMS or a FDBMS, and note that, with the FDBMS, we can have more useful results to count on, especially in decision-making support, due to gradual answer.

Some fuzzy query languages have been proposed. SQLf is defined as a structured fuzzy query language for relational databases originated as an extension of standard SQL through the use of fuzzy sets. A SQLf query has the following syntax:

```
select <attributes> from <relations> where <fuzzy condition>
      with calibration [n|λ|n,λ]
```

The result is a fuzzy set of rows formed by attributes in the *select* clause from the Cartesian product of the relations in the *from* clause. The membership degree of each row is given by the satisfaction degree to the fuzzy condition. In the *where* clause, some logical expressions can be used with user-defined fuzzy terms (atomic predicates, modifiers, connectors, comparators and quantifiers) and predefined fuzzy operators [2]. The *with calibration* clause specify a tolerance which may express a maximum number “n” of best rows (quantitative) or may specify a satisfaction degree

Table 1. Instance of PERSON relation, results of CLASSIC query and a FUZZY query

PERSON			CLASSIC			FUZZY		
Name	Age		Name	Age		Name	Age	μ
Carmen Domínguez	26		Pedro Ramírez	15		Pedro Ramírez	15	1.0
Daniel Rodríguez	12		Nadia Núñez	20		Nadia Núñez	20	1.0
Isabel Castro	14		José Sánchez	25		José Sánchez	25	1.0
José Chacon	13					Isabel Castro	14	0.8
José Sánchez	25					Carmen Domínguez	26	0.8
Juan Amado	27					José Chacon	13	0.6
León Judá	33					Juan Amado	27	0.6
Nadia Núñez	20							
Pedro Ramírez	15							
Ramón Carrizo	30							

“ λ ” where returned row must have degree greater or equal to “ λ ” (qualitative). Fuzzy terms are specified by means of a *create fuzzy* statement.

create fuzzy <term kind > <term name> as <term specification>

4 Fuzzy Queries Design

Database schema for USB survey of student opinion is quite complex. For shake of simplicity, we present a reduced version of this schema. We denote underlined the primary keys and italics the foreign keys. Relational schema is: DEPARTMENT(did, dname), PROFESSOR (pid, pname, *did*), COORDINATION(oid, oname), COURSE (cid, cname, *did*), PROGRAM(*oid*, *cid*), PERIOD (rid, rbegin, rend), SURVEY (vid, *cid*, *rid*, *snum*, *pid*), ITEM (vid, *inum*, *irep*, ivalue). For the expression of fuzzy queries we need to define linguistic terms. In this work, terms are Spanish that is Venezuela’s official language. We have established:

— Predicates: *excelente*, *bueno*, *regular* and *deficiente* (in English: excellent, good, regular and deficient). In order to simplify the user specification of these predicates, we establish an entailment between them, using only three parameters X_e , X_b and X_r that would be replaced in the actual definition. See Fig. 2.

— Comparators: *MuchoMejorQue*, *MejorQue*, *PeorQue* and *MuchoPeorQue* (in English: Far better, Better, Worse and Much Worse). For simplicity of final user

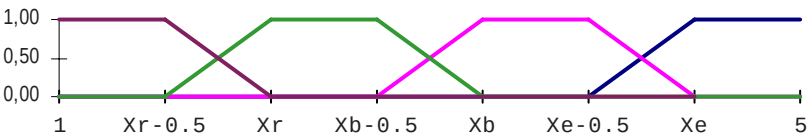


Fig. 2. Fuzzy predicates deficiente, regular, bueno and excelente

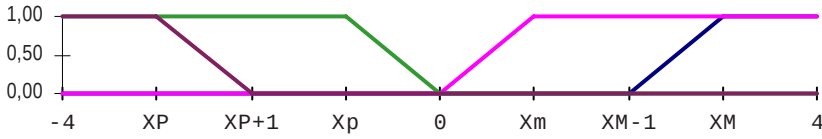


Fig. 3. Fuzzy comparators MuchoPeorQue, PeorQue, MejorQue and MuchoMejorQue

specification, we establish only one parameter for each comparator XM, Xm, Xp or XP. In this case the comparison degree is specified to be the satisfaction degree of the distance between elements x and y measured as the difference in a fuzzy set that defines the comparator. See Fig. 3.

— **Quantifiers:** *LaMayoría*, *LaMitad* and *LaMinoría* (in English: Most, Half, and Few). We establish an entailment between these quantifiers, using only two parameters XM and Xm that would be replaced in the actual quantifiers' definition. See Fig. 4.

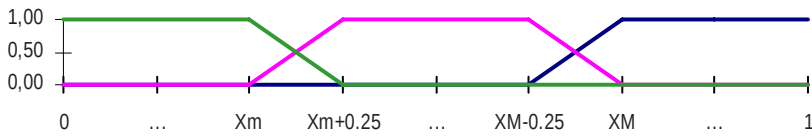


Fig. 4. Fuzzy quantifiers LaMinoría, LaMitad and LaMayoría

In the application queries especificaition is made via a final user interface. Therefore, user needs to know neither SQLf nor fuzzy sets theory. Some fuzzy queries that user may buid using our application, an their expression in SQLf, are these:

Example 2: Given a period \$X1, a course \$X2, an item number \$X3, and a threshold \$X4, professors that have excellent result for this item. It would be expressed in SQLf as:

```
select V.vid,V.cid,V.rid,V.snum,V.pid from SURVEY V, ITEM I where
  V.rid=$X1 and V.cid=$X2 and V.vid=I.vid and I.inum=$X3 and
  I.ival=excelente with calibration $X4;
```

Example 3: Given a period \$X1, items number \$X2, and a threshold \$X3, find professors that have worst result for this item than the previous period result. It would be expressed as:

```
select V1.vid,V1.cid,V1.rid,V1.snum,V1.pid from SURVEY V1, ITEM I1,
SURVEY V2, ITEM I2 where V1.rid = $X1 and V1.vid=I1.vid and I1.inum=$X2
and V2.rid = $X1-1 and V2.vid=I2.vid and I2.inum=$X2 and V2.pid=V1.pid
and I1.ival PeorQue I2.ival with calibration $X3;
```

Example 4: Given a period \$X1, items number \$X2, and a threshold \$X3, find professors that have result for this item much better than the result in most of previous periods. In SQLf:

```
select V1.vid,V1.cid,V1.rid,V1.snum,V1.pid from SURVEY V1, ITEM I1 where
  V1.rid=$X1 and V1.vid=I1.vid and I1.inum=$X2 and I1.ival MuchoMejorQue
```

```
LaMayoría(select I2.ival from SURVEY V2, ITEM I2 where V2.rid<$X1 and
V2.vid=I2.vid and I2.inum=$X2 and V2.pid=V1.pid) with calibration $X3;
```

Example 5: Given a period \$X1 and a threshold \$X2, find professors that have deficient result only for few items. It should be expressed in SQLf as:

```
select V.vid,V.cid,V.rid,V.snum,V.pid from SURVEY V, ITEM I where
V.rid=$X1 and V.vid=I.vid group by V.vid,V.cid,V.rid,V.snum,V.pid having
LaMinoría are I.ival=deficiente with calibration $X2;
```

5 User Interfaces

The system’s most remarkable feature is the variety of queries and results, which can be used by career coordinators and department chiefs in evaluating professors’ performance. This application allows — Queries by Academic Period: it allows observing professor performance during a specific period or in a single course throughout different periods. — Queries by Course: Only one course might need to be consulted and this is made possible with this interface. — Queries by Item: review of emitted opinion on a certain item might be needed no matter periods or courses. All these interfaces are similar and contain in an intuitive way the different clauses of the SQLf statement (see Fig. 5).

Fig. 5. Screen for specifying a query by course for the review of survey results. First two bands establish classic conditions for the where clause, third band is used for specifying a simple fuzzy condition with a predicate, fourth band is intended for specifying query nesting or partitioning using a fuzzy quantifier, and last band allows the specification of calibration.

In addition, it features a module where users can redefine their preferences. Interface design of this application is rather simple but powerful to capture the meaning of fuzzy terms and the structure of fuzzy queries. There is a screen for fuzzy predicates, other for fuzzy comparators and other for fuzzy quantifiers (Fig. 6). Interfaces are in Spanish, Venezuela’s official language.

Modificación de criterios de Evaluación

Criterios de evaluación anteriores		
El criterio excelente esta establecido entre los valores	4.31	5.0
El criterio bueno esta establecido entre los valores	3.2	3.8099999999999996
El criterio regular esta establecido entre los valores	2.03	2.7
El criterio deficiente esta establecido entre los valores	1.0	1.5299999999999998

Criterios (Desplace hasta donde Usted Considere Aceptable Cada Criterio)

Criterio	Grado	Valor	
		Desde	Hasta
EXCELENTE		3.75	5
BUENO		2.75	3.25
REGULAR		1.75	2.25
DEFICIENTE		1	1.25

Opciones

Seleccione Cada Criterio y Presione el Boton Aceptar

Fig. 6. Screen for modifying fuzzy predicates specification. Sliding bars allow the selection of corresponding values for parameters according to previous section. Actual parameter values are shown as a reference. There are also predefined values that user may load with a button.

6 Concluding Remarks

We presented here a real life application of SQLfi Fuzzy Database Management System (FDBMS). It concerns to the review of the student opinion survey about teaching performance at Universidad Simón Bolívar, Venezuela. It would be very useful in decision-making for contract renovation, promotion and stimulus program. It provides intuitive interfaces, thus final user is not forced to deal with fuzzy sets theory and SQLf details. We have used a little but representative subset of SQLf. It would be interesting to explore all the power of SQLf in development of other real life applications. Relevance of this work is that at present time FDBMS is an open research field without commercial products of wide diffusion and therefore there are too few real applications implemented using FDBMS.

Acknowledgments. This work is supported in part by the Governmental Venezuelan Foundation for Science, Innovation and Technology FONACIT Grant G-2005000278. *“Don't be deceived, my dear brothers. Every good and perfect gift is from above, coming down from the Father of the heavenly lights, who does not change like shifting shadows. He chose to give us birth through the word of truth, that we might be a kind of firstfruits of all he created.”* James 1:16-18 (New International Version).

References

- 1 Aranda, C., Galindo, J.: Gestión de una Agencia de Viajes usando Bases de Datos Difusas y FSQ, <http://www.turismo.uma.es/turitec/turitec99/pdf/bd1.pdf>
- 2 Bosc, P., Pivert, O.: SQLf: A Relational Database Language for Fuzzy Querying. IEEE Transactions on Fuzzy Systems 3(1) (1995)

- 3 Cox, E.: Relational Database Queries using Fuzzy Logic. *Artificial Intelligent Expert*, 23–29 (1995)
- 4 Eduardo, J., Goncalves, M., Tineo, L.: A Fuzzy Querying System based on SQLf2 and SQLf3. In: *The XXX Latin-American Conference on Informatics* (2004)
- 5 Galindo, J.: New Characteristics in FSQ, a Fuzzy SQL for Fuzzy Databases. *WSEAS Transactions on Information Science and Applications* 2 2, 161–169 (2005)
- 6 Goncalves, M., León, G., Martínez, D., Tineo, L.: Una Herramienta Web para la Evaluación de Desempeño Docente, sobre un Sistema de Consultas Difusas. In: *Actas de InfoUy CLEI 2002, Conferencia Latinoamericana de Informática, Uruguay* (2002)
- 7 Goncalves, M., Tineo, L.: SQLf Flexible Querying Language Extension by means of the norm SQL2. *Proc. of The 10th IEEE Inter. Conf. on Fuzzy Systems* 1 (2001)
- 8 Goncalves, M., Tineo, L.: SQLf3: An extension of SQLf with SQL3 features. In: *Proc. Of The 10th IEEE International Conference on Fuzzy Systems, December 2001, vol. 3* (2001)
- 9 Goncalves, M., Tineo, L.: A Web Tool for Web Document and Data Source Selection with SQLfi. In: *Proc of the 9th International Conference on Enterprise Information Systems, ICEIS* (2007)
- 10 Kacprzyk, J., Zadrozny, S.: Fuzzy Queries in Microsoft AccessTM v.2. In: *Proc. of Fuzzy IEEE 1995 Workshop on Fuzzy Database Systems and Information Retrieval* (1995)
- 11 Ma, Z.M., Yan, L.: Generalization of Strategies for Fuzzy Query Translation in Classical Relational Databases. *Information and Software Technology* 49(2), 172–180 (2007)
- 12 Nakajima, H., Sogoh, T., Arao, M.: Fuzzy Database Language and Library-Fuzzy Extension to SQL. In: *Proc. of Second IEEE International Conference on Fuzzy Systems*, pp. 477–482 (1983)
- 13 Valdés, Y.: Propuesta del Modelo Conceptual EER Difuso en el Control de la Calidad del Papel (CMPC-MAULE): Implementación en FSQ, Proyecto Especial de Grado, Universidad Católica del Maule, Talca, Chile (2003)

Hierarchy Encoding with Multiple Genes^{*}

Martin van Bommel and Ping Wang

Department of Mathematics, Statistics, and Computer Science
St. Francis Xavier University
Antigonish, Nova Scotia, B2G 2W5, Canada
{mvanbomm,pwang}@stfx.ca

Abstract. Efficient implementation of type inclusion testing is important for data and knowledge base systems employing large hierarchies. The bit vector encoding of a partially ordered set representing a type hierarchy permits constant-time type inclusion testing. Current such methods employ a simple encoding, associating a single gene for each join-irreducible element. We present an algorithm using multiple genes for those elements with many siblings. The new algorithm provides a significant improvement on the encoding size for hierarchies with low multiple inheritance factors.

1 Introduction

Inheritance hierarchies can appear in numerous applications, including object-oriented programming languages, databases, knowledge bases, and conceptual graphs. The organization of objects that are instances of classes into such a partial order of nodes requires the efficient testing of type inclusion (subsumption) to determine if a relationship exists between a pair of objects. To facilitate this end, and to reduce the size of the storage structure, several encoding methods have been proposed which model a hierarchy as a lattice, which is represented using a bit-vector encoding [2,3,4,8,11].

Two variations of such encoding methods are static and dynamic encodings. Given a pre-defined hierarchy, the goal of static encoding is to efficiently produce the best possible encoding, with the assumption that the hierarchy will remain unchanged. The encoding of an object-oriented compiler's class hierarchy is one possible application [8]. For the case of an evolving hierarchy, dynamic encoding employs an efficient incremental algorithm to encode new nodes as they are added. Persistent applications such as data and knowledge bases, where new types or classes are added during execution, require such an encoding.

The usual approach to encoding is to associate a distinguishing bit, called a gene, to select nodes, and have nodes inherit the genes of all of their ancestors. Heuristics are employed since the encoding problem is NP-hard [4]. A major limitation of these approaches is that they employ a *simple encoding* scheme; that is, they use only one gene per node, despite the fact that the optimal encoding

^{*} Supported by the Natural Sciences and Engineering Research Council of Canada.

usually requires more. “Most authors have shied away from the complexity of multiple gene encoding.” [4]

This paper presents an encoding algorithm which employs a heuristic in which specific patterns of nodes are assigned multiple genes, thus reducing the overall encoding size. The goal is to achieve a more compact encoding without sacrificing the efficiency of the encoding method or the flexibility of dynamic encoding.

2 Background

An inheritance hierarchy can be represented as a partially ordered set, *poset* $(P, \leq)$, with the binary relation $\leq$ reflexive, antisymmetric, and transitive. The relation $a \leq b$ implies that either a and b are the same class, a is an immediate child of b , or a is an immediate child of some class c , and $c \leq b$. Two elements a and b of a poset P are said to be *comparable* if either $a \leq b$ or $b \leq a$.

Consider a poset $(P, \leq)$ and a subset A of P . An element $b \in P$ is called an *upper bound* of A if $a \leq b$ for all $a \in A$. Also, b is called a *least upper bound* (LUB) of A if it is also the case that $b \leq a$ whenever a is an upper bound of A . Conversely, element $b \in P$ is called a *lower bound* of A if $b \leq a$ for all $a \in A$, and a *greatest lower bound* (GLB) of A if it is also the case that $a \leq b$ whenever a is a lower bound of A .

A *lattice* is a poset in which every pair of elements has a GLB and LUB. The LUB of the set of two elements $\{a, b\}$ is denoted $a \vee b$ and is called the *join* of a and b . Similarly, the GLB of $\{a, b\}$ is denoted $a \wedge b$ and is called the *meet* of a and b . A *lower semilattice* is a poset in which every pair of elements has a GLB. A more detailed discussion on posets and lattices can be found in standard texts on discrete mathematics, such as [7].

Consider the poset $(P, \leq)$. Let $Anc(x) = \{y \in P \mid y < x\}$ and $Desc(x) = \{y \in P \mid y > x\}$. An element $j \in X$ is said to be *join-irreducible* if there exists $x \in X$ such that $x \notin Desc(j)$ and $Anc(j) \subset Anc(x) \cup \{x\}$. Similarly, we can define *meet-irreducible*. Let $J(P)$ and $M(P)$ denote the sets of all join-irreducible elements and all meet-irreducible elements respectively. Markowsky [9] shows that the only optimal encoding preserving join (meet) operations for a lattice are those obtained by associating a different number or bit to each join-irreducible (meet-irreducible) element.

Let $(P, \leq)$ be a poset, and $\chi : J(P) \rightarrow S = \{1, \dots, k\}$. Habib, Nourine, and Raynaud [6] provide the following definition. A *simple encoding* is the mapping $\varphi(x) : X \rightarrow 2^S$ with $\varphi(x) = \bigcup_{j \in Anc(x)} \chi(j)$ such that φ is an embedding from P onto 2^S ; that is, $x \leq_P y$ iff $\varphi(x) \subset \varphi(y)$. The problem is to determine the most compact such encoding. Unfortunately, Caseau, Habib, Nourine and Raynaud [4] prove that the simple encoding is polynomially equivalent to graph coloring, and in turn, it is an NP-hard problem. In fact, the general encoding problem (also known as the 2-dimension problem) is to find the *smallest* k such that there exists a mapping $\varphi(x) : X \rightarrow 2^{\{1, \dots, k\}}$ such that $\varphi(x) \rightarrow 2^S$ with $\varphi(x) = \bigcup_{j \in Anc(x)} \chi(j)$ is an embedding from P onto 2^S ; that is, $x \leq_P y$ iff $\varphi(x) \subset \varphi(y)$. Clearly, this is also an NP-hard problem.

3 Previous Work

A number of papers have examined the encoding problem [1,3,4,5,8,10,11], but this section focuses on those contributing directly to the results in this work.

Caseau [3] proposes a top-down encoding that reuses bit positions during the incremental encoding of a lattice. The procedure is presented in an incremental fashion; that is, it can handle the addition of new nodes as children of leaves, “which is the case with most object-oriented class hierarchies.” [3] The algorithm does require a preliminary lattice completion step. The variation of this encoding proposed by van Bommel and Wang [11] does not require the preprocessing step. Caseau [3] also proposes that, for nodes with a large number of children, splitting the children into smaller groups by adding additional nodes to the hierarchy can reduce the size of the encoding.

Krall, Vitek, and Horspool [8] present what they call a “near optimal” simple encoding method which is claimed to be faster, simpler, and producing smaller codes. Their algorithm is based on the coloring of a “conflict graph” built from the join-irreducible elements. It also requires a preprocessing step that “splits” and “balances” the hierarchy, introducing new nodes in an effort to reduce the encoding size. The algorithm is based on a static hierarchy, and is not easily adapted to a dynamic application. Caseau et al. [4] present an improved variation of the “near optimal” encoding by computing a more accurate conflict graph. They also discuss a way to compute the coloring of a dynamic conflict graph.

4 Multiple Genes

In order to maintain the concept of simple encoding where a single gene is assigned to each join-irreducible element, both Caseau [3] and Krall et al. [8] introduce new nodes to the middle of the hierarchy to which such genes can be assigned. An equivalent encoding without the additional nodes requires the assignment of multiple genes per node. This section examines two situations where such an assignment is preferable.

4.1 An Alternative to Gene Redefinition

Caseau’s [3] top-down incremental encoding scheme ensures that at most one gene is assigned to a node by redefining the gene for a node if it results in conflicts. Consider the insertion of node g in Fig. 1. Since g inherits the genes of its ancestors, it includes genes one through four, leading to an apparent relationship with node e . Caseau’s gene redefinition reassigns c gene five, thus removing the conflict, and have node g inherit genes one and three through five.

The modification of genes already defined for nodes can lead to difficulties in implementation in persistent incremental applications where many nodes include this gene. An alternative approach is to permit the introduction of more than

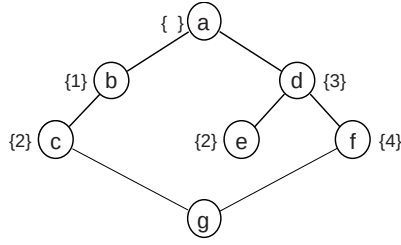


Fig. 1. Resolving conflicts

one gene per node. In Fig. 1, node e can have gene five added to its current assignment, thereby eliminating the conflict.

What follows is a variation of the algorithm of van Bommel and Wang [11]. The encoding γ being considered is a mapping from a hierarchy representing a poset $(P, \leq)$ to a lattice $(L, \supseteq)$, where L contains binary words. Two codes c_1 and c_2 are related by $c_1 \supseteq c_2$ if and only if for every 1-bit in the code of c_2 , code c_1 contains a 1-bit in the same position. The operations $c_1 \cap c_2$ and $c_1 \cup c_2$ are equivalent to bit-by-bit binary *and* and *or*. Nodes added to the hierarchy are encoded by calling function `Encode`. The function assumes the existence of function `Parents`, which returns the set of immediate parents of a node.

```

Encode( $x : node$ ) ::
  let  $\{x_1, \dots, x_n\} = \text{Parents}(x)$  in
    if  $n = 1$  then
       $\gamma(x) \leftarrow \text{Add}(x_n, \text{FreeBit}(x_n))$ 
    else
       $\gamma(x) \leftarrow \bigcup_{i=1}^n \gamma(x_i)$ 
      ResolveConflicts( $x$ ).
  
```

```

ResolveConflicts( $x : node$ ) ::
  for each  $y \in \text{IncSet}(x)$  do
    if  $\gamma(x) = \gamma(y)$  then
      Propagate( $x, \text{FreeBit}(x)$ )
      Propagate( $y, \text{FreeBit}(y)$ )
    else if  $\gamma(x) \subset \gamma(y)$  then
      Propagate( $x, \text{FreeBit}(x)$ )
    else if  $\gamma(x) \supset \gamma(y)$  then
      Propagate( $y, \text{FreeBit}(y)$ ).
  
```

```

FreeBit( $x : node$ ) : int ::
  forbid  $\leftarrow \gamma(x)$ 
  for each  $y \in \text{Nodes}$  do
    if  $\gamma(y) \supset \gamma(x)$  then
      forbid  $\leftarrow \text{forbid} \cup \gamma(y)$ 
    if not  $\gamma(x) \supseteq \gamma(y)$  then
      forbid  $\leftarrow \text{forbid} \cup \text{BitDiff}(\gamma(x), \gamma(y))$ 
  return FirstZero(forbid).
  
```

`Add` (not shown) simply returns the code of node x_n with an extra 1-bit (gene) in the position provided by the call. The first available bit position that distinguishes a code from others without defining conflicts is found via `FreeBit`. This function compares the code with all currently assigned codes of incomparable nodes to determine those bit positions that are unavailable, and picks the first available one. `BitDiff` (not shown) returns a code containing only this bit

for a pair of codes, or returns the empty code if none exists. FirstZero (also not shown) returns the position of the lowest order zero of a code.

The variation which may result in multiple genes per node is in the detection of conflicts and the changes that are applied to solve them. A conflict occurs if the new code assigned to a node creates the same code as some other node or the appearance of a relationship with an incomparable node. ResolveConflicts compares the new code with that of all nodes in the incomparable set. The incomparable set of a node, determined via IncSet (not shown), is made up of classes that are neither subclasses nor superclasses of the class. If the new code matches that of another node, both the new node and the other are assigned another gene. If the new node's code makes it appear as an ancestor of an incomparable node, the incomparable node is assigned another gene, which is added to its descendants' codes. However, if the new node's code makes it appear to be a descendent, the new node is assigned another gene.

The effect of Propagate (not shown) is to add to the codes of the node, and its descendants, the bit position returned by FreeBit. If any new conflicts are created during this process, they are resolved in the same manner with a recursive call to ResloveConflicts.

The proof of correctness of this algorithm and its efficiency are similar to that of the vBW algorithm [11]. The code size produced is similar to that of both the vBW algorithm [11] and Caseau's [3] without balancing.

4.2 An Alternative to Splitting

Both Caseau [3] and Krall et al. [8] reduce the size of the simple encoding by splitting children lists. For those nodes with more than some limit of a number of children, two new intermediate parent nodes are added to the hierarchy, with each assigned some of the children. As illustrated in the left encoding in Fig. 2, the effect of adding these nodes is the assignment of multiple genes per node, which is essentially multiple gene encoding.

An alternative to the introduction of intermediate nodes is the assignment of multiple bits for each of the children. The right encoding in Fig. 2 illustrates a more efficient encoding of the hierarchy that cannot be achieved via splitting. Krall et al. [8] claim that "using more bits to identify a type complicates the algorithm and makes it difficult to find a near optimal solution." While this may be true for a static encoding, in a dynamic environment the use of multiple bits can accommodate additions to the hierarchy more efficiently than splitting.

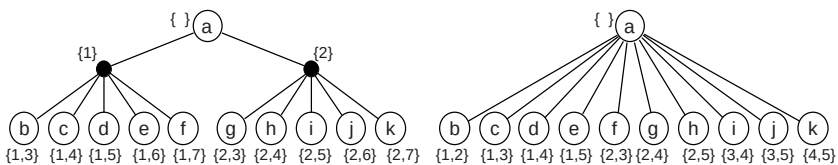


Fig. 2. Splitting vs multiple gene encoding of a tree

Caseau et al. [4] hint at a heuristic for assigning multiple genes, but details are not provided. Also presented is a bounds on the size of such an encoding in the case that the hierarchy is a tree, and a claim that the number of bits required to encode the children of a node with n children is in the range $[\log n, 2 + \log n]$ using multiple bits per child.

A dynamic encoding method which incorporates multiple genes for the encoding of the immediate children of nodes with many children assigns genes in an incremental manner. When a node has only a few children, the children are assigned a single gene. Once more children are added, the genes already assigned are combined to create multiple gene assignment for each of the nodes. The number of distinct genes d used for c children is the minimal such d where there exists some g such that

$$\binom{d}{\lceil d/2 \rceil} \geq c \quad \text{and} \quad \binom{d}{g} \geq c \quad .$$

The number of genes assigned per child g is the minimal such g .

4.3 An Example

Consider the hierarchy illustrated in Fig. 3. Applying the multiple bit per child strategy on the first eleven nodes leads to the encoding illustrated on the top right. Adding the final two nodes to this encoding leads to node m inheriting all five genes, thus leading to conflicts with nodes b through e . The resolution of these conflicts results in gene six being added to nodes b through e , as illustrated on the bottom left. Contrasting this to the best possible use of simple splitting for the same hierarchy illustrated on the bottom right illustrates the inefficiencies inherent in such an approach.

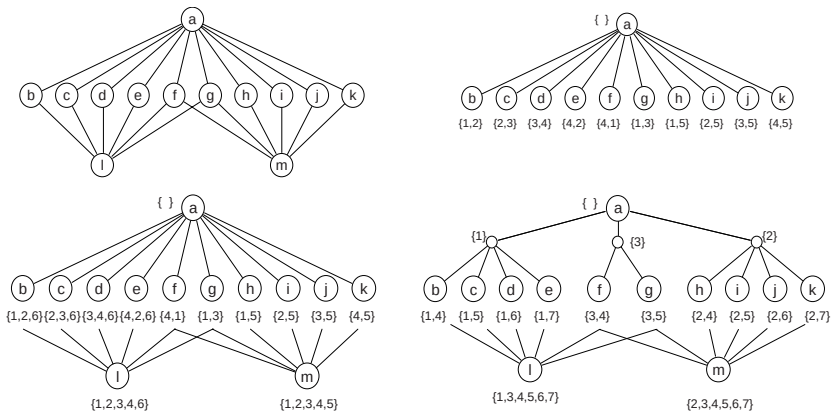


Fig. 3. Encoding another hierarchy

5 Balancing

A significant factor in the efficiency of the encoding of Krall et al. [8] is from the balancing that is used during the splitting phase. Balancing attempts to minimize the maximum path length of the hierarchy. The path length is a count of the number of children of the nodes along the longest path from the root to the leaf that require a unique gene. As stated, Caseau et al. [4] note that developing an incremental version of this balancing and splitting would be difficult.

Incorporating the idea of balancing into the multiple bit top-down encoding method described above requires a complete analysis of the hierarchy. Unfortunately there is no guarantee that further additions to the hierarchy will not counteract any measures taken in an attempt to balance. In a discussion of a possible incremental version of their algorithm, Krall et al. [8] note that incremental balancing could result in a large increase in the space and execution time required if the balancing changes, and thus state that “balancing could be replaced by a simpler splitting process which ignores the depth of the tree.”

5.1 An Alternative to Balancing

As shown, splitting children itself does not necessarily improve the encoding size when used with multiple gene encoding. Rather, splitting children with few descendants from those with many descendants can improve the encoding size. An analysis on sample hierarchies revealed that the best heuristic to use for such splitting involves determining a rough estimate of the encoding size for children. Each leaf node requires a gene. For each parent node, the number of genes required is calculated as the number of children plus the maximum of the number of genes required by its children. For nodes with multiple children, if there exists at least one child with an encoding size of ten or greater and children with encoding size of less than ten, a new node is introduced as a parent of those nodes with encoding size less than ten.

5.2 An Example

Consider the hierarchy illustrated using a regular encoding technique on the top left of Fig. 4, which requires fifteen distinct genes. Applying the multiple genes per child strategy on nodes with more than three siblings leads to the encoding illustrated on the top right, which requires twelve distinct genes. Using the encoding of Krall et al. [8] with balancing, as illustrated on the bottom left, also requires twelve distinct genes. Contrasting this to the use of the alternative approach to splitting used with multiple genes per child illustrated on the bottom right, which requires only ten distinct genes, illustrates the effectiveness of the approach.

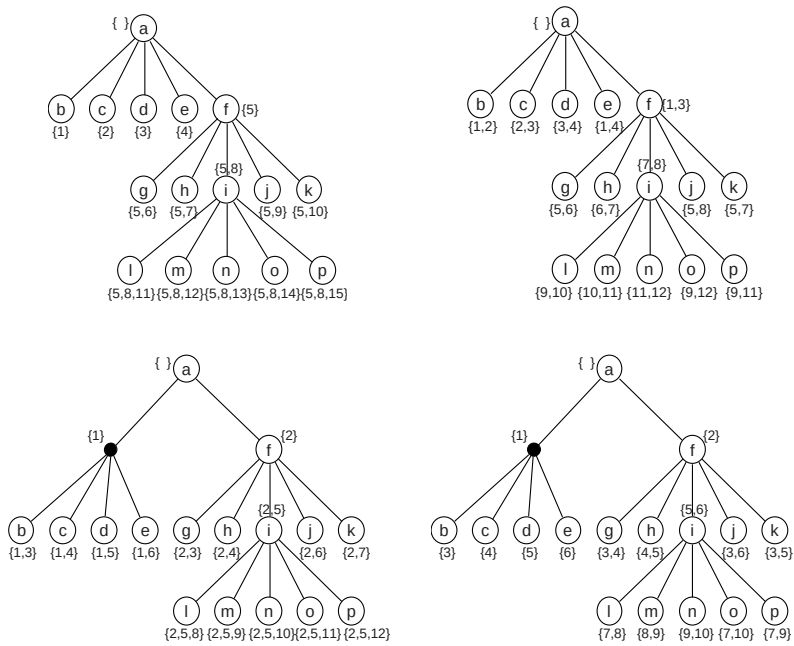


Fig. 4. Multiple gene encoding with splitting

6 Comparison and Conclusions

To compare the performance of the use of both of the multiple gene strategies with other approaches, test data containing class libraries was obtained from the collection of Andreas Krall [8]. Table 1 outlines the characteristics of the data and illustrates the comparison with using the encoding of Caseau [3] without splitting, Caseau [3] with splitting, the multiple gene encoding method described in this paper, Krall et al. [8] using balancing, and the multiple gene

Table 1. Hierarchy characteristics and encoding results

library	classes	depth	parents		childs max	Bit Counts				
			max	avg		Caseau	Split	Multi	Krall	Multi2
Visualwks2	1956	15	1	1	181	420	124	58	50	41
Digitalwk3	1357	14	1	1	142	325	116	50	36	35
NextStep	311	8	1	1	142	177	92	27	23	21
ET++	371	9	1	1	87	181	61	42	30	24
Unidraw	614	10	2	1.01	147	227	96	44	30	29
Self	1802	18	9	1.05	232	297	180	72	53	55
Ed	434	11	7	1.66	77	128	90	76	54	79
LOV	436	10	10	1.71	76	130	92	77	57	70
Laure	295	12	3	1.07	7	34	34	30	23	28
Java	225	7	3	1.04	73	97	50	23	19	19

encoding method with simple splitting described in this paper. The balancing method of Krall is shown to produce a compact encoding, but is only effective for static encoding. For several of the hierarchies, the use of multiple genes is almost as effective as balancing, and the multiple gene encoding method with simple splitting can be better, especially for hierarchies with little or no multiple inheritance. Only the Krall encoding is not performed in a top-down incremental fashion, and thus could not be used for dynamic hierarchies.

Reducing the overall size of the encoding of type hierarchies improves the efficiency of type inclusion testing. With static encoding, a thorough analysis of the hierarchy can be used, but with dynamic hierarchies, such analysis is difficult. Simple encoding can produce reasonable results, if employed in conjunction with careful balancing and splitting approaches. The use of multiple genes in encoding certain elements of large type hierarchies improves the overall encoding size for those hierarchies with little multiple inheritance, and simple splitting helps to further reduce it. This method is competitive to that involving simple encoding with balancing, but can be used in applications involving dynamic hierarchies, or combined with balancing in a static encoding scheme.

References

1. Agrawal, R., Borgida, A., Jagadish, J.V.: Efficient management of transitive relationships in large data and knowledge bases. In: *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, June 1989, pp. 253–262 (1989)
2. Ait-Kaci, H., Boyer, R., Lincoln, P., Nasr, R.: Efficient implementation of lattice operations. *ACM Transactions on Programming Languages and Systems* 11(1), 115–146 (1989)
3. Caseau, Y.: Efficient handling of multiple inheritance hierarchies. In: *Proceedings of the International Conference on Object-Oriented Systems, Languages, and Applications*, October 1993, pp. 271–287 (1993)
4. Caseau, Y., Habib, M., Nourine, L., Raynaud, O.: Encoding of multiple inheritance hierarchies and partial orders. *Computational Intelligence* 15(1), 50–62 (1999)
5. Ganguly, D., Mohan, C., Ranka, S.: A space-and-time efficient coding algorithm for lattice computations. *IEEE Transactions on Knowledge and Data Engineering* 6(5), 819–829 (1994)
6. Habib, M., Nourine, L., Raynaud, O.: A new lattice-based heuristic for taxonomy encoding. In: *International KRUSE Symposium on Knowledge Retrieval, Use and Storage for Efficiency*, pp. 60–71 (1997)
7. Kolman, B., Busby, R., Ross, S.: *Discrete Mathematical Structures*, 3rd edn. Prentice Hall, New Jersey (1996)
8. Krall, A., Vitek, J., Horspool, R.N.: Near optimal hierarchical encoding of types. In: *European Conference on Object-Oriented Programming*, pp. 128–145 (1997)
9. Markowsky, G.: The representation of posets and lattices by sets. *Algebra Universalis* 11, 173–192 (1980)
10. van Bommel, M.F., Beck, T.J.: Incremental encoding of multiple inheritance hierarchies supporting lattice operations. *Linköping Electronic Articles in Computer and Information Science* 5(1) (December 2000)
11. van Bommel, M.F., Wang, P.: Encoding multiple inheritance hierarchies for lattice operations. *Data and Knowledge Engineering* 50(2), 175–194 (2004)

Knowledge Mining for the Business Analyst

Themis Palpanas¹ and Jakka Sairamesh²

¹ University of Trento

² IBM T.J. Watson Research Center

Abstract. There is an extensive literature on data mining techniques, including several applications of these techniques in the e-commerce setting. However, all previous approaches require that expert users interpret the data mining results, making them cumbersome to use by business analysts. In this work, we describe a framework that shows how data mining technology can be effectively applied in an e-commerce environment, delivering significant benefits to the business analyst. Using a real-world case study, we demonstrate the added benefit of the proposed method. We also validate the claim that the produced results represent actionable knowledge that can help the business analyst improve the business performance, by significantly reducing the time needed for data analysis, which results in substantial financial savings.

1 Introduction

Data mining has been extensively used in the past for analyzing huge collections of data, and is currently being applied to a variety of domains [9]. More recently, various data mining techniques have also been proposed and used in the more specific context of e-commerce [19,10]. The downside of the previous discussion is that, despite all the success stories related to data mining, the fact remains that all these approaches require the presence of expert users, who have the ability to interpret the data mining results. We argue that an important problem regarding the use of data mining tools by business analysts is the gap that exists between the information that is conveyed by the data mining results, and the information that is necessary to the business analyst in order to make business decisions.

In this work, we describe a framework that aims at bridging the gap mentioned above. We demonstrate how data mining technology can be effectively applied in an e-commerce environment, in a way that delivers immediate benefits to the business analyst. The framework we propose takes the results of the data mining process as input, and converts these results into actionable knowledge, by enriching them with information that can be readily interpreted by the business analyst. By applying this methodology to the vehicle manufacturing industry, we show that the business analyst can significantly reduce the time needed for data analysis, which results in substantial financial savings. For example, shortening the vehicle warranty resolution cycle by 10 days can save an Original Equipment

Manufacturer (OEM) around \$300m and reduce the total number of warranty claims by 5%.

In summary, we propose a method that allows the analyst to:

- quickly discover frequent, time-ordered, event-patterns,
- automate the process of event-pattern discovery using a feedback loop,
- enrich these event-patterns with demographics related to the processes that generated them,
- identify the commonalities among these event-patterns,
- trace the events back to the process that generated them, and
- predict future events, based on the history of event-patterns.

Previous work has proposed algorithms for solving the problem in the first step of the above process, and we leverage those algorithms. Yet, there is no work that covers the entire process that we are proposing in this study.

1.1 Related Work

There exists a large body of work in knowledge extraction from huge datasets [8], and many recent studies try to improve on the performance and functionality of the various data mining algorithms. However, these algorithms are targeted to expert users, and are very cumbersome to be used by business analysts. The same is also true for commercial data mining solution packages [3,4,2]. The CRISP-DM project [1] has proposed an end-to-end process for data mining. In this paper, we present a specific framework that addresses some of the problems arising in one steps of the above process, namely, the step of organising and presenting to the user the new knowledge gained by applying some data mining techniques.

Several applications of data mining in the e-business environment [19,10] prove the usefulness of this kind of techniques for improving the business operations. Nevertheless, the proposed solutions are specific to the particular tasks for which they were developed. Previous work has also studied the problem of applying data mining techniques in the context of data warehouses [14,16,17,13], which are used by business analysts during the decision-making process.

2 Knowledge Mining Framework

In this section, we describe in more detail the framework that we propose for knowledge mining. Figure 1 depicts a high level view of our approach.

Pre-Processing. When the data first enter the system, there are a series of pre-processing steps that aim at cleaning the data and bringing them in a format suitable for our system. The purpose of the pre-processing steps is to make sure that all the data conform to the same standard and are semantically comparable. Examples of actions taken during this phase include converting all measures to metric system, all codes to the same standard, and ensuring the semantic equivalence of data.

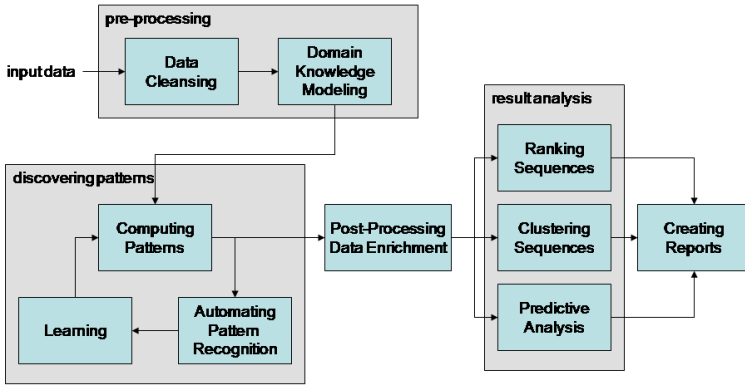


Fig. 1. Process overview

Discovering Patterns. In the next series of steps, we identify patterns of interest in the data. The computed patterns should also conform to some user-defined constraints. These constraints determine the form of the patterns that are going to be computed. For example, the user may set the minimum and maximum lengths for each of the reported patterns, in the case where we are interested in mining for frequent sequences. The user may also define the maximum elapsed time between the start and the end of a sequence, as well as between two consecutive items in the sequence. The proposed framework, allows the users to try out the various parameter alternatives (a principle similar to exploratory mining [12]).

Data Enrichment. The frequent sequences that have been produced during the previous step only hold some minimal, vital information in order to identify the items that participate in each one of the frequent sequences. The goal of data enrichment is to take as input the computed frequent sequences, and correlate them with all the relevant bits of information that are stored in the system. Data originating from different parts of the business are gathered and integrated with the data mining results. The data may refer to various phases of the lifecycle of each specific item, and they enrich the discovered sequences with contextual information pertaining to the processes that generated them.

Result Analysis. The enriched sequences that were produced during the previous phase can then be analyzed in a variety of ways. The diagram of Figure 1 depicts three different analysis modules that we are proposing.

- **Ranking Sequences.** This module uses various knowledge models and utility functions in order to rank the event-patterns according to different criteria. The results of this methodology capture a macro view of the business performance issues based on a small but important fraction of the available information (for details see [6]).

- **Clustering Sequences.** The purpose of this module is to use the contextual information associated with each event-pattern in order to identify clusters of similar event-patterns. When a clustering algorithm (for example, [7]) is run against the enriched event-patterns, it produces groupings of those patterns that are semantically meaningful within the business context, and help the analyst to gain insight on the root causes for each behavior.
- **Predictive Analysis.** This module aims at using the history of event-patterns to predict future events. The identified patterns represent an approximation of the historical behavior of the items under examination. Given these data, we can make projections for the future behavior [15].

Note that apart from the above three analysis modules that we have implemented in our system, other modules can be incorporated in the proposed framework as well.

Report Creation. In the final phase of the framework we propose, we produce a series of reports that summarize the results of the previous data analysis phases. In our framework we have developed a graphical user interface, through which the analyst has access and can customize several different report types.

3 Case Study

In this section, we describe a case study with one of the two vehicle manufacturing companies we collaborated with. The manufacturer has data relevant to the characteristics of each vehicle. The data refer to warranty claims made for vehicles of particular models during the period of the year 2005. The first dataset we examined includes almost 2,000,000 records of warranty claims, referring to almost 1,000 different failure reasons. These claims were referring to approximately 250,000 unique vehicles, corresponding to more than 100 different vehicle models.

3.1 Proposed Process

In this section, we elaborate on the process that we propose for extracting new knowledge from the available data, for the vehicle manufacturing company. In the following presentation, we focus on a single stream of knowledge extraction and management, namely, that of *frequent sequences*.

Pre-Processing. The input to our system are the warranty claims of the vehicles. Each warranty claim comes with the following pieces of information.

- The vehicle identification number (VIN).
- The date that the vehicle visited the mechanic.
- The mileage of the vehicle at the time of the visit to the mechanic.
- The ids for all the failed vehicle parts, as identified by the mechanic.
- The root cause for each one of the failures, as identified by the mechanic.
- The cost of the visit to the mechanic, broken down by part and labor cost.

All these claims are gathered from different sources, normalized, and stored in a database called *warranty claims data store*. The organization of these data in a database helps in the subsequent steps of data analysis.

Frequent Sequence Mining. We define failure set $f = (f_1, f_2, \dots, f_m)$ to be a nonempty set of failures (i.e., a set describing all the known events in which a particular vehicle failed), and failure sequence $s = \langle s_1, s_2, \dots, s_n \rangle$ to be an ordered list of failure sets. A failure-sequence which contains all the failures of a vehicle (identified by the VIN) ordered by the claim date, is called a vehicle-failure sequence. We say that a vehicle supports a failure-sequence if this failure sequence is contained in the vehicle-failure sequence of this particular vehicle. The problem of mining failure patterns [18,11,5] is to find the failure sequences that are supported by many vehicles.

Post-Processing Data Enrichment. The output of our frequent failure patterns analysis is a long list of failure sequences, which we augment with statistics relating to several of the attributes contained in the warranty claims database. More specifically, with each failure pattern we associate the following information (related to the vehicles that failed): Number of vehicles supporting the failure pattern; l most common vehicle models; l most common engine types; l most common manufacturing plants; l most common makers; l most common build-years. In addition to the above information, which refers to the entire pattern, we also associate with each particular failure of each failure pattern the following information: l most common cause-codes for the failure; Minimum, maximum, average, and standard deviation of the mileage at which the failure occurred; Minimum, maximum, average, and standard deviation of the replacement part cost for the failure; Minimum, maximum, average, and standard deviation of the labor part cost for the failure.

Result Analysis - Report Creation. The wealth of this information makes the use of a database imperative, in order to organize all the results and help in their analysis. Even though the database can be used to produce a listing with all the failure patterns along with the additional statistics outlined above, the real power of this approach stems from the fact that the analyst can query the database, and get results that are relevant to the particular aspects of the failure patterns she is focusing on.

The failure pattern database can be used to answer any query that correlates any combination of the attributes that the database captures (listed in the previous paragraphs). A small sample of the questions that this database can answer is presented in the following sections.

3.2 Evaluation Results Using Aggregated Behavior Reports

We first present sample results on failure sequence statistics related to the aggregated behavior of vehicles. This is a way to direct the analyst to examine some problems that are considered more important than others. Our framework can be useful in answering many other queries as well.

Ranking by the difference of part and labor cost for a specific type of failures

In this case, we are looking for failure sequences that involve engine failures, for which the labor cost is more than the part cost. This query is interesting, because it shows that for some repairs under warranty the part cost is very small, while the labor cost is much higher (around \$1,200). When redesigning an engine, it may be beneficial to take these cases into consideration so as to make sure that that the labor cost for repairing this kind of failures is reduced (e.g., access to a particular part is made easier).

Ranking by frequency of occurrence for a specific engine model

This query reports the most frequent failure sequences, for which the most common engine model is “A”. These sequences are interesting, because they inform the analyst what are the most frequent recurring problems related to a specific engine model. Our data show that more than 2,300 vehicles that are mounted with the specific engine model exhibit the same problems. These results can help identify potential problems in the design or manufacturing of this engine model.

3.3 Evaluation Results Using Focused Reports

In the examples that follow, we isolate some frequent failure sequences of interest, and analyze the detailed characteristics of the vehicles that exhibit these failure sequences.

Once again, it is important to note that the following are just two examples we used during our study. The approach we propose allows the user to focus on any aspect that is important to the team of data analysts.

Failures in brakes and electrical components

This example focuses on vehicles that visited the mechanic two different times within the same year, for problems related to the brakes and the electrical components. Table 1 lists the most common causes for each failure in the sequence. As we can see, in the majority of the cases the failure cause is the same. This indicates that there may be a problem with the design or the manufacturing of the failed parts.

Table 1. Example 1: Cause code break-down. (All the code ids and descriptions have been replaced by dummy values for reasons of anonymity.)

failure X cause code	%	failure Y cause code	%
inoperative	72	leaking	64
shorted	13	rubs	9
leaking	7	broken	4

Table 2. Example 2: Cause code break-down. (All the code ids and descriptions have been replaced by dummy values for reasons of anonymity.)

failure X cause code	%	failure Y cause code	%	failure Z cause code	%
leaking	100	leaking	47	leaking	21
		loose	5	broken	11
				loose	11

Failures in driving axle, wheels, and brakes.

In this case, we examine vehicles that visited the mechanic three different times during the same year, for problems related to the driving rear axle, the wheels, and the brakes. Note that all these problems relate to the same sub-system of the vehicles, and have occurred one after the other. When we look at the causes of these failures (see Table 2), it is obvious that the main problem is leaking parts. Furthermore, it turns out that all the vehicles that had those failures were manufactured in year 2004, and the vast majority of them, almost 90%, in the same factory (see Table 3). These data provide a clear indication to the analyst as to where to direct the efforts necessary for resolving the problems in the vehicles.

Table 3. Example 2: Demographics break-down. (All the code ids and descriptions have been replaced by dummy values for reasons of anonymity.)

bld_dte	%	model	%	plant	%	engine	%
2004	100	M1	74	P1	89	E1	79
		M2	21	P2	5	E2	16
		M3	5	P3	5	E3	5

3.4 Discussion

By following the proposed process, the analyst (or in this particular case, the engineer responsible for the design and manufacturing of engine type “E”) can quickly focus on the most important problems that are relevant to her work. Actually, the same analyst can view, prioritize, and evaluate the corresponding information according to different criteria, such as cost, which relates to the financial aspect of the business, or frequency of failures, which relates to customer satisfaction and the marketing aspect.

These benefits of the presented method were also validated by different analysts from the two vehicle manufacturing companies that provided us with their data. By using our framework, they were able to not only cut down the time spent on data analysis and interpretation to a small fraction of the time they used to spend (from more than 45 days down to a few days for specific types of

analysis), but they were also able to perform more focused analysis and deliver reports with a high impact factor.

4 Conclusions

In this work, we described a framework that enriches the results of the data mining process with information necessary for the business analyst. This information pertains to different aspects of the data mining results, and can help the analyst manipulate and interpret these results in a more principled and systematic way.

As our case study with a real-world problem demonstrates, the proposed framework has a great value in the e-commerce context. It converts the data mining results into actionable knowledge, that the business analyst can use to improve the business operations. In our case study, this meant changing the design and manufacturing processes in order to avoid expensive warranty claims for specific failures.

References

- [1] Cross Industry Standard Process for Data Mining, <http://www.crisp-dm.org/>
- [2] DB2 Intelligent Miner, <http://www-306.ibm.com/software/data/iminer/>
- [3] Microsoft SQL Server Business Intelligence,
<http://www.microsoft.com/sql/solutions/bi/default.aspx>
- [4] Oracle Data Mining, <http://www.oracle.com/technology/products/bi/odm/>
- [5] Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential Pattern Mining Using a Bitmap Representation. In: International Conference on Knowledge Discovery and Data Mining (2002)
- [6] Chen, M., Sairamesh, J.: Ranking-Based Business Information Processing. In: E-Commerce Technology (2005)
- [7] Domeniconi, C., Papadopoulos, D., Gunopulos, D., Ma, S.: Subspace clustering of high dimensional data. In: SDM (2004)
- [8] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2006)
- [9] Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge (2000)
- [10] Li, Y.-H., Sun, L.-Y.: Study and Applications of Data Mining to the Structure Risk Analysis of Customs Declaration Cargo. In: ICEBE, pp. 761–764 (2005)
- [11] Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Technical Report C-1997-15, Department of Computer Science, University of Helsinki (1997)
- [12] Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory Mining and Pruning Optimizations of Constrained Associations Rules. In: ACM SIGMOD International Conference, Seattle, WA, USA (June 1998)
- [13] Palpanas, T.: Knowledge Discovery in Data Warehouses. ACM SIGMOD Record 29(3), 88–100 (2000)
- [14] Palpanas, T., Koudas, N., Mendelzon, A.O.: Using datacube aggregates for approximate querying and deviation detection. IEEE Trans. Knowl. Data Eng. 17(11), 1465–1477 (2005)

- [15] Pednault, E.: Transform Regression and the Kolmogorov Superposition Theorem. Technical Report RC-23227, IBM Research (2004)
- [16] Sarawagi, S.: User-Adaptive Exploration of Multidimensional Data. In: VLDB International Conference, Cairo, Egypt, September 2000, pp. 307–316 (2000)
- [17] Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-driven Exploration of OLAP Data Cubes. In: International Conference on Extending Database Technology, Valencia, Spain, March 1998, pp. 168–182 (1998)
- [18] Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
- [19] Yang, X., Weiying, W., Hairong, M., Qingwei, S.: Design and Implementation of Commerce Data Mining System Based on Rough Set Theory. In: ICEBE, pp. 258–265 (2005)

Controlling the Behaviour of Database Servers with 2PAC and DiffServ

Luís Fernando Orleans¹, Geraldo Zimbrão¹, and Pedro Furtado²

¹ COPPE/UFRJ - Computer Science Department - Graduate School and Research in Engineering – Federal University of Rio de Janeiro

² CISUC – Department of Informatics Engineering – University of Coimbra - Portugal
{lforleans,zimbao}@cos.ufrj.br, pnf@dei.uc.pt

Abstract. In order to avoid stress conditions in information systems, the use of a simple admission control (SAC) mechanism is widely adopted by systems' administrators. Most of the SAC approaches limit the number of concurrent work, redirecting to a waiting FCFS queue all transactions that exceed that number. The introduction of such a policy can be very useful when the most important metric for the system is the total throughput. But such a simple AC approach may not be sufficient when transactions have deadlines to meet, since in stressed scenarios a transaction may spend a lot of time only waiting for execution. This paper presents 2 enhancements that help keeping the number of transactions executed within the deadline near to the throughput. The enhancements are DiffServ, in which short transactions have priority, and a 2-Phase Admission Control (2PAC) mechanism, which tries to avoid the previous mentioned problem by limiting the queue size dynamically using informations provided by a feedback control. It also introduces the QoS-Broker – a tool which implements both SAC and 2PAC – and uses it to compare their performances when submitted to the TPC-C benchmark. Our results show that both total throughput and throughput within deadline increase when the 2 enhancements are used, although it becomes clear that 2PAC has a much bigger impact on performance than DiffServ.

Keywords: QoS, DiffServ, Admission Control, 2PAC.

1 Introduction

Congested request processing systems lead to degraded performances and quality of service. So far, a Simple Admission Control (SAC) was sufficient to avoid stress conditions on database servers. All incoming transactions that would make the server exceed the allowed multi-programming level (MPL) [17] would be directly forwarded to a FCFS queue and wait there for future execution. Although this is a fair approach, where all transactions are served in their arrival order, it is hard to be applied if transactions have deadlines to meet. Intuitively, as the transactions arrival rate increases, the number of transactions that will be redirect to the waiting queue will also increase, hence the time each transaction will spend waiting for execution. The inclusion of

time-constraints in database transactions makes the traditional admission control implementations unable to reach the QoS conditions that might be established between the service provider and the service contractor. This way, another admission control layer is necessary: one that should try to avoid the uncontrolled growing of the queue, creating what we called a 2-Phase admission control (2PAC). In this paper we propose the 2PAC policy, which learn workloads patterns and adapts the queue size to provide much better guarantees than SAC. Our proposal also handles efficiently transaction heterogeneity: we also propose a DiffServ enhancement to the basic 2PAC proposal, which prioritize short transactions by preventing them to pass through any of the admission control layers when it makes sense, boosting the performance. Our experimental work compares SAC, 2PAC and 2PAC plus DiffServ, using the TPC-C benchmark. We do not cover a best-effort configuration, since it had been already extensively studied. Two scenarios were created: a high load scenario and a medium load scenario. Both will be explained in deeper details later. The 2PAC has better throughput than its competitors in all cases, having a small fraction of accepted transactions breaking their deadlines. The main reason is the second admission control phase, which tries to optimize the queue size according to the transactions previous durations. This way, all transactions that presents high response times that would probably miss their deadlines are rejected by the middleware.

The remaining of the paper is organized as follows: section 2 lists the related work. Section 3 gives the background in which this work stands for. Section 4 presents the 2 Phase Admission Control algorithm and section 5 explains how a DiffServ mechanism can be used within a database context. Section 6 gives the experiment setup, while section 7 lists the results obtained from those experiments. Finally, section 8 concludes the paper, also proposing some future works.

2 Related Work

In [6] the authors propose session-based Admission Control (SBAC), noting that longer sessions may result in purchases and therefore should not be discriminated in overloaded conditions. They propose self-tunable admission control based on hybrid or predictive strategies. Reference [5] uses a rather complex analytical model to perform admission control. There are also approaches proposing some kind of service differentiation: [3] proposes architecture for Web servers with differentiated services. [16] proposes an approach for Web Servers to adapt automatically to changing workload characteristics and [9] proposes a strategy that improves the service to requests using statistical characterization of those requests and services.

In [15], it is proposed a dynamic load-balancing algorithm, called ORBITA, that tries to guarantee deadlines by applying some kind of DiffServ, where small tasks have priorities and execute on a dedicated server. The big tasks have to pass through the admission control mechanism and can be rejected, if the maximum MPL (calculated in runtime) had been reached.

Comparing to our own work, none of the previously mentioned intended to study how to manage the growth of the waiting queue, which has its size dynamically

computed according to the workload characteristics, in order to accept only the tasks that will be able to execute within the deadline.

3 Background

This section provides the necessary background for the complete understanding of this work.

3.1 Simple Admission Control (SAC)

The SAC architecture is presented in figure 1: all incoming transactions that would exceed the MPL wait for execution in a FCFS queue. Every time a transaction is committed, it leaves the system and another one is picked up from the queue.

Reference [17] presents the architecture illustrated in figure 1, and proves that the near-to-maximum throughput can be achieved with a relatively low MPL. The work proposed by [15] shows an interesting mechanism that tries to avoid deadlines from being broken. Although that study is about load-balancing, a theme that is out of the scope of this paper, it guided our work as it dynamically compute the size of the request and calculate the appropriate MPL.

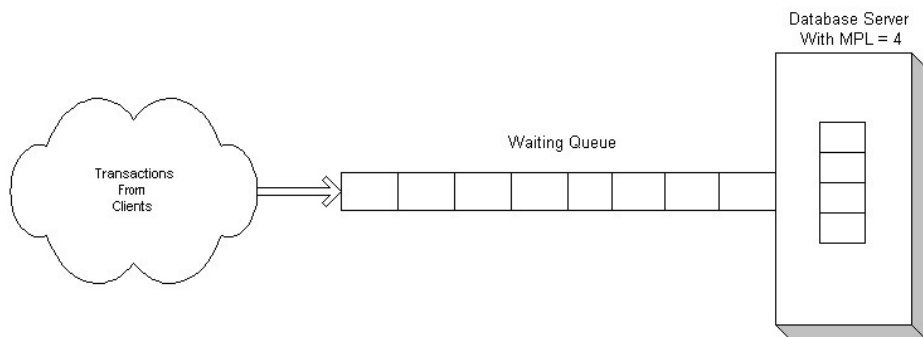


Fig. 1. Standard admission control components. A waiting queue and a fixed number of concurrent executing transactions, also known as multiprogramming level (MPL).

3.2 TPC-C Benchmark

The TPC-C benchmark (<http://www.tpc.org/tpcc/>) is a specification provided by the Transaction Processing Performance Council that simulates a real-world database application using a closed-model, where the number of clients is fixed and each transaction submission is followed by a “think time”, which is responsible for simulating the time a user spends analyzing the last request before sending a new transaction. When the think time assumes lower values, the load on the system increases, causing a stress scenario.

3.3 QoS-Broker

The QoS-Broker middleware [10] is a tool that intercepts the conversation between an application and a database server. It uses the Proxy Design Pattern to provide a transparent admission control layer, without requiring deep source code modifications.

As all the SQL statements go through the Proxy, it is possible to calculate the mean time each transaction type takes to execute. It is also possible to keep a list with the used transactions as well as to classify the transactions in read-only and update transactions (which comprise insert, update and delete SQL statements). For each transaction, the QoS Broker is able to calculate the mean response time, thus providing an accurate way to estimate the queue waiting time and try to avoid the deadline misses.

The QoS-Broker can be configured to use both SAC and 2PAC, as well as Diff-Serv. If SAC is used, then the transactions that would cause system overload are redirected to FCFS queue. On the other hand, if the 2PAC flag is set, those transactions are enqueued only if the QoS-Broker calculates that they would be able to execute within the deadline – otherwise the transactions are not accepted. Finally, the Diff-Serv flag gives priority to short transactions.

4 Two-Phase Admission Control (2PAC)

This type of admission control provides not one, but two layers of admission control. It controls the number of concurrent transactions executing (by checking the MPL constraint) and it also restricts the number of transactions on the queue. The 2PAC mechanism needs to know an estimation about transactions durations and, this way, it can calculate the time a transaction will have to wait on the queue and determine if it will end its execution before the deadline. To determine the acceptance of a transaction on the queue, the following constraint has to be satisfied:

$$T_{\text{exec}} + T_{\text{queue}} < T_{\text{deadline}} \tag{1}$$

Where T_{exec} is the time a transaction will need to execute, T_{queue} is the time a transaction will have to wait for execution and T_{deadline} is the deadline time.

The basic 2PAC algorithm is listed below:

Algorithm 1: 2PAC

1. When a transaction tx is submitted by a client, calculate the estimated execution time for tx .
 2. If the sum of estimated execution time of tx with the estimated time tx will spend on the queue is less than the deadline then tx is enqueued. Otherwise, it is rejected.
-

In order to effectively calculate the estimated time a transaction spend executing, the QoS Broker middleware was improved and was added a feature that saves a pounded mean of past transactions for each type. The transaction's mean duration is calculated as:

$$E(N)=0.4*T_x(N-1)+0.3*T_x(N-2)+\dots +0.1*T_x(N-4) \quad (2)$$

Where $E(N)$ is the estimated duration of the N_{th} arrived transaction, and T_x clauses are the real durations of the last transactions of the same type. Our formula uses the last 4 real measures to calculate the estimation, trying to obtain a mean time which reflects the recent behaviour of the database server. This way, 2PAC tries to avoid executing transactions that will possibly break their deadlines by rejecting them.

It's worth to notice that the rejection of a transaction does not necessarily imply on a real rejection: the transaction may be redirected to a lower priority waiting queue, and only execute when the other one is empty, for example.

The last statement about the implementation of the 2PAC method on the QoS Broker tool concerns about the recognition of the transaction's type in runtime. In order to reduce complexity of the implementation, the client is obligated to inform the QoS Broker which transaction he/she is about to execute.

5 DiffServ

DiffSev is an acronym for Differentiation of Services. It is a fundamental building block for QoS networks in the sense that it gives some kind of priority to more critical packets, e.g., video or audio streaming packets. This concept can be applied also in information systems, where some transactions may be prioritized. According to the related work [15], in a distributed or parallel environment, it is feasible to give priority to short transactions without overwhelming the throughput of the big transactions.

In this work, the short transactions can pass through the admission control mechanisms if the DiffServ flag is properly set. Intuitively, this can be a reasonable choice in a 1-server configuration only when the number of big transactions is much higher than the number of small ones – otherwise it can degrade the performance by letting lots of small tasks execute at the same time.

6 Experiment Setup

All experiments were executed using a Pentium 4 3.2GHz, with 2GB RAM DDR2 and a 200 GB SATA HD which was responsible for creating the threads that simulate the clients. The server was a Pentium II MMX 350MHz, with 256MB RAM and a 60GB IDE HD. Both computers were running a Debian Linux, with Kernel version 2.6 and were connected by a full-duplex 100Mbps Ethernet link. A PostgreSQL 8.1 database server was running on the server machine and the database size was 1.11GB.

The client machine used a Sun Microsystems' Java Virtual Machine, version 1.5. The database was created with 10 warehouses, which allows a maximum of 100 terminals (emulated clients) by the TPC-C specification.

6.1 Workload Composition

In order to effectively test the performance of the QoS-Broker, we used 2 transaction mixes and 2 load scenarios, leading up to 4 different workload compositions. The details of each are given below. The default transaction mix is the TPC-C default mix, while the heavy-tailed alternative depicts a typical scenario where short transactions represent 95% of the system load, similar to the observations contained in [13].

Table 1. Transaction mixes

Transaction Mix	Transactions Occurrences
Heavy-Tailed (used for comparison purposes)	• Delivery: 5% • Stock-Level: 95% • Other transactions: 0%
Default	• New Order: 45% • Payment: 43 % • Other transactions: 4%

Table 2. Think times

Load Type	Think Time
Medium-Load	Exponential distribution, with mean 8 seconds and a maximum of 80 seconds
High-Load	Exponential distribution, with mean 4 seconds and a maximum of 40 seconds.

6.2 Other Considerations

Each experience was executed for a period 20 minutes, and during the first 5 minutes no data was collected. Before each round, the database was dropped and then recreated, which guarantees that all experiments encountered the same database state. Finally, all experiments were made using 100 emulated clients.

The TPC-C specification establishes 5 seconds as the maximum response time for each transaction – except for the Stock Level transaction, which has to be completed within 20 seconds. Our experiments showed that the Stock Level is the fastest transaction of the transaction mix, taking 70.35ms to execute in average, considering

standalone execution (transaction executing on an empty system). We considered a 5 seconds deadline for all transactions, including Stock Level.

7 Results

For the first round of experiences, a heavy-tailed scenario was used. The results are displayed in figures 2 and 3. Since the number of short transactions is much greater than the number of long transactions, no DiffServ was necessary. The first thing to be noticed from the graphics is that the throughput within deadline (TWD – number of transactions that ended within the deadline per minute) for the SAC case is much smaller than the total throughput (TT – total number of transactions that ended per minute) for the medium load case. For the high load scenario, the TWD is even worse: no transaction ended within the deadline at all! So it turns out the necessity of a new approach, one which tries to avoid deadline misses. The graphic in figure 2 shows the effectiveness of the 2PAC approach, since the TWD is always close to TT.

In figure 3, we can see that the maximum throughput is reached with only 2 MPL by the system with 2PAC. This occurs because the workload is comprised mostly by short transactions, which execute very fast. As the MPL increases, the TT keeps almost unaltered, while TWD decreases. Two conclusions can be taken from the last statement: 1) 2PAC is more robust than SAC, since SAC has its performance deeply degraded by higher MPL values; and 2) the workload variability is responsible for the degradation of TWD of 2PAC in figure 3 for higher MPL values, since more long transactions get to execute.

In the second round of experiments, we used the default transaction mix, established by the TPC-C specification. In this transaction mix, the number of short transactions is much smaller than the number of big transactions, so we used the DiffServ engine as another enhancement to the system. Figures 4 and 5 show the graphics for medium load and high load, respectively. Again, it turns out that 2PAC mechanism is

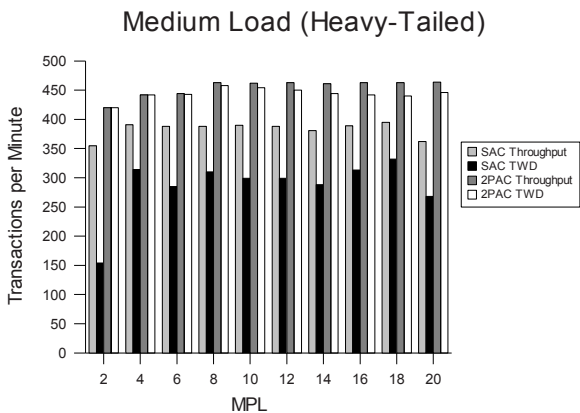


Fig. 2. Performance comparison in a heavy-tailed, medium-loaded scenario. SAC has a fewer number of transactions within deadline (TWD) than 2PAC.

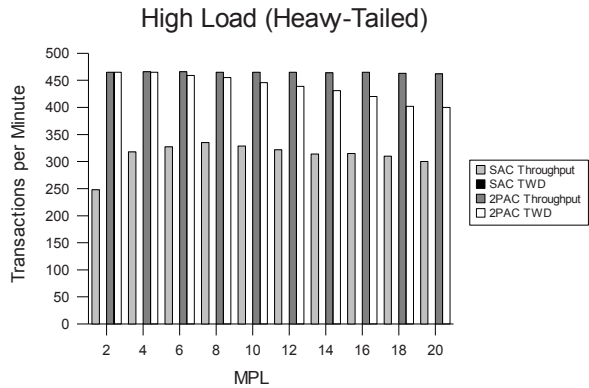


Fig. 3. Performance comparison in a heavy-tailed, high-loaded scenario. SAC has no transactions ended within deadline (TWD) whereas 2PAC scales very well.

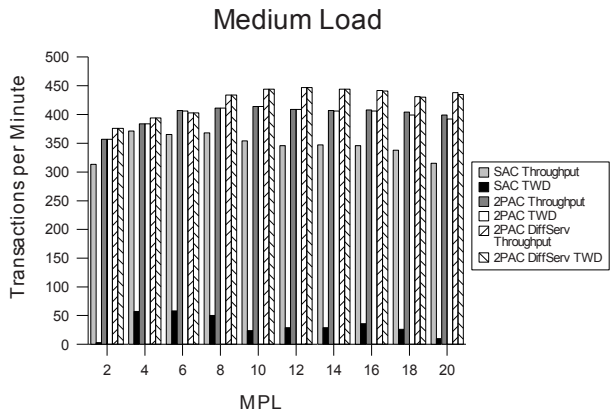


Fig. 4. Performance comparison in a medium-loaded scenario, using the standard transaction mix. SAC performs poorly whereas its competitors are able to handle a reasonable number of transactions without breaking their deadlines.

much more robust than SAC, which can be attested by the comparison of TWD of both methods: 2PAC's TWD is almost the same as TT for all MPL values, in SAC these values of TWD are very low (0 for the high load scenario). The use of DiffServ also increased the performance, but its contribution is smaller than the isolated use of 2PAC.

Figure 4 shows that SAC has a very small TWD, due to the massive presence of long transactions, achieving its highest value (58 transactions per minute) with MPL 6. On the other hand, the 2PAC approach has as maximum TWD 414 transactions per minute, more than 7 times higher than SAC! When the DiffServ flag is set, then the TWD goes to 447 transactions per minute.

The graphic 5 shows that no matter how the load on the system increases, the TT of all techniques remains almost unaltered. But, in SAC case, the queue grows uncontrolled

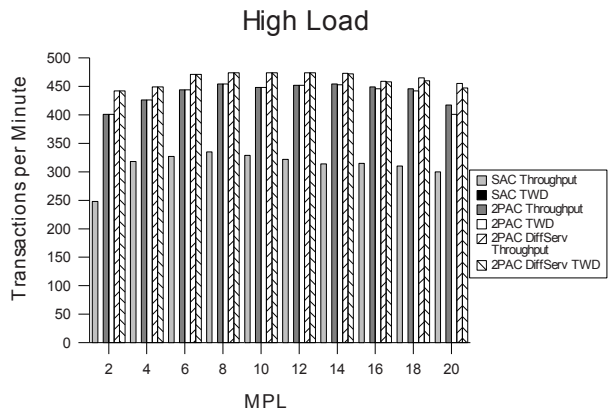


Fig. 5. Performance comparison in a high-loaded scenario, using the standard transaction mix. Again, SAC has no transaction ending before the deadline was reached. The 2PAC approach has a better performance and the DiffServ enhancement represents a little performance gain.

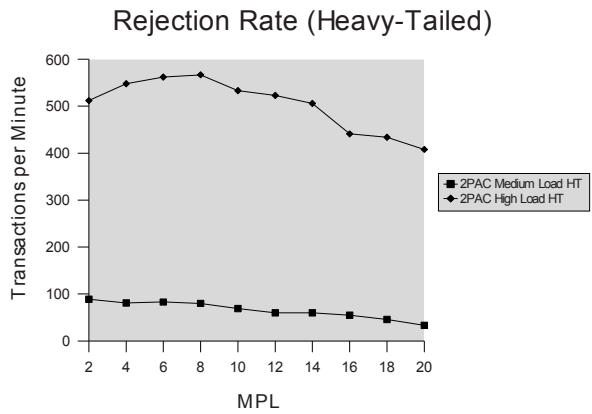


Fig. 6. Transactions rejection rate when 2PAC is used in a Heavy-Tailed scenario. As the MPL increases, the number of rejected transactions decreases.

and the time a transaction spends waiting for execution makes it miss the deadline. In fact, TWD is zero for all MPL values in SAC. Again, the 2PAC has a better performance, keeping TWD close to TT, reaching its maximum value at 14 MPL with 453 transactions per minute. With DiffServ enhancement, the maximum TWD is reached with MPL values between 8 and 12, with 474 transactions per minute.

The graphics 6 and 7 compliment the last two providing informations about what makes the system perform so well with 2PAC: the rejection of transactions that would not be able to fully execute within the deadline. The rejection rate when DiffServ is

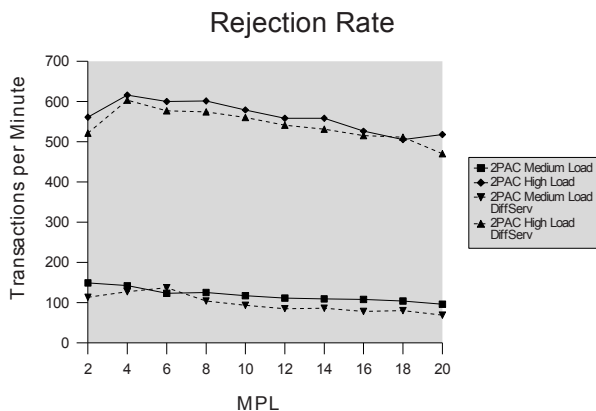


Fig. 7. Transactions rejection rate when 2PAC is used in a TPC-C's default transaction mix scenario. Note that the use of the DiffServ enhancement is directly responsible for the fewer number of transactions being rejected.

used is smaller since there are no small transactions being rejected: they all go directly to execution. This reinforces the idea that there is no silver bullet to solve this deadline guarantees problem. If the system load is too high, then some of the transactions should be discarded since the system will not be able to execute them within the deadline.

8 Conclusions

Stressed database servers may use an admission control mechanism to achieve better throughput. This becomes a problem when transactions have deadlines to meet as traditional admission control (SAC) models (with a FCFS waiting queue) may not be applied, since the queue time is a potential point for QoS failures. This paper presented the 2-Phase Admission Control (2PAC) model, which estimates the execution time of a transaction according to a mathematical formula that takes into account the last 4 execution times of the same transaction. Once the execution time is estimated, it is possible to calculate how long the transaction will spend on the waiting queue and, furthermore, if the transaction will be able to be completed before the deadline. If the middleware calculates that the transaction would miss the deadline, it is rejected by the system.

Then, we compared the performances of both models (SAC and 2PAC), under medium and high loads and for both heavy-tailed and TPC-C's default workload. A last enhancement, DiffServ, was included in the experiments for the default workload. DiffServ gives priority for short transactions, letting them pass through the admission control mechanism.

The results showed that, in order to reach a good rate of transactions ended within deadline, it is necessary to limit the number of transactions on the waiting queue. This way, all transactions that are supposed to miss their deadlines are rejected. Despite the

high number of transactions being rejected, the improvement on system's performance is almost 8 times higher when both 2PAC and DiffServ enhancements are used.

As future works, we intend to investigate how a multi-server environment is affected by admission control policies and try to establish a connection between them, leading to a complete highly scalable, distributed database system.

References

1. Amza, C., Cox, A.L., Zwaenepoel, W.: A Comparative Evaluation of TransparentScaling Techniques for Dynamic Content Servers. In: International Conference On Data Engineering (2005)
2. Barker, K., Chernikov, A., Chrisochoides, N., Pingali, K.: A Load BalancingFramework for Adaptive and Asynchronous Applications. *IEEE Transactions on Parallel and Distributed Systems* 15(2) (2004)
3. Bhatti, N., Friedrich, R.: Web server support for tiered services. *IEEE Network* 13(5), 64–71 (1999)
4. Cardellini, V., Casalicchio, C.M., Yu, P.S.: The State of the Art in Locally Distributed Web-Server Systems. *ACM Computing Surveys* 34, 263–311 (2002)
5. Chen, X., Mohapatra, P., Chen, H.: An admission control scheme for predictable server response time for Web accesses. In: World Wide Web Conference, Hong Kong (2002)
6. Cherkasova, Phaal: Session-based admission control: A mechanism for peak load management of commercial Web sites. *IEEE Req. on Computers* 51(6) (2002)
7. Crovella, M., Bestavros, A.: Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 835–836 (1999)
8. Dyachuk, D., Deters, R.: Optimizing Performance of Web Service Providers. In: International Conference on Advanced Information Networking and Applications, Niagara Falls, Ontario, Canada, pp. 46–53 (2007)
9. Elnikety, S., Nahum, E., Tracey, J., Zwaenepoel, W.: A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. In: World Wide Web Conference, New York City, NY, USA (2004)
10. Furtado, P., Santos, C.: Extensible Contract Broker for Performance Differentiation. In: International Workshop on Software Engineering for Adaptive and Self-Managing Systems, Minneapolis, USA (2007)
11. Harchol-Balter, M.: Task assignment with unknown duration. *Journal of the ACM*, 49 (2002)
12. Harchol-Balter, M., Crovella, M., Murta, C.: On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59(2), 204–228 (1999)
13. Harchol-Balter, M., Downey, A.: Exploiting process lifetime distributions for dynamic load-balancing. *ACM Transactions on Computer Systems* (1997)
14. Knightly, E., Shroff, N.: Admission Control for Statistical QoS: Theory and Practice. *IEEE Network* 13(2), 20–29 (1999)
15. Orleans, L.F., Furtado, P.N.: Fair load-balance on parallel systems for QoS. In: International Conference on Parallel Programming, Xi-An, China (2007)
16. Pradhan, P., Tewari, R., Sahu, S., Chandra, A., Shenoy, P.: An observation-based approach towards self managing Web servers. In: International Workshop on Quality of Service, Miami Beach, FL (2002)

17. Schroeder, B., Harchol-Balter, M.: Achieving class-based QoS for transactional workloads. In: International Conference on Data Engineering, p. 153 (2006)
18. Serra, A., Gaïti, D., Barroso, G., Boudy, J.: Assuring QoS Differentiation and Load-balancing on Web Servers Clusters. In: IEEE Conference on Control Applications, pp. 885–890 (2005)
19. TPC-C Benchmark Homepage, <http://www.tpc.org/tpcc/>

Compressing Very Large Database Workloads for Continuous Online Index Selection*

Piotr Kołaczkowski

Warsaw University of Technology, Institute of Computer Science
P.Kolaczkowski@ii.pw.edu.pl

Abstract. The paper presents a novel method for compressing large database workloads for purpose of autonomic, continuous index selection. The compressed workload contains a small subset of representative queries from the original workload. A single pass clustering algorithm with a simple and elegant selectivity based query distance metric guarantees low memory and time complexity. Experiments on two real-world database workloads show the method achieves high compression ratio without decreasing the quality of the index selection problem solutions.

Keywords: database workload compression, automatic index selection.

1 Introduction

Efficient solution of the *index selection problem* (ISP) usually requires the calculation of a set of queries approximating the database workload. There are many algorithms for solving the ISP [1,2,3,4,5,6,7,8]. The more queries are given at their input, the more resources and time they need to find good results. As it is inconvenient to create the input data by hand, database administrators usually use query logs to get the approximation of the future workload. Unfortunately the query logs can be very large. Millions of queries logged by a transactional system during a day are not uncommon. Without compression, solving the ISP for all the logged queries may take unreasonably large amounts of time. For the purpose of practical usability of the automated physical database tuning tools, it is crucial to reduce the number of the analyzed queries. The simplest solution would be to pick only a small random sample of the workload. However, this method might skip some important queries by accident and in effect significantly degrade the results given by the database tuning tool. A good method for workload compression should not have such side effects. It should achieve high compression ratio while keeping the quality of the results at the acceptable level.

For the purpose of the experimental analysis we define the *quality ratio* as the ratio between the original workload costs after automatic selection of indexes with and without using the workload compression. The *quality loss* is the difference between these two costs. The ideal workload compression algorithm would

* The work has been granted by Polish Ministry of Education (grant No 3T11C 002 29).

have the quality ratio of 1 and quality loss of 0. The most known algorithms cause some quality loss and their quality ratio is less than 1.

The *online ISP* is a more difficult variant of ISP, where indexes can be created or dropped at any time and the workload is not known in advance, but should be treated as a stream of continuously incoming queries. Solving this problem is crucial for building self-managing, autonomic database systems [9,10]. Efficient solving of the online ISP requires different workload compression algorithms that can compress the workload *incrementally*, “on the fly”.

We present an algorithm that removes most of the queries from the workload and leaves only those essential to solve the online ISP accurately and efficiently. To achieve this, the algorithm employs single-pass clustering with a selectivity based metric. Because each query is analyzed only once, the method can be used to improve the performance of continuous online database tuning systems [11,12,13]. To the best of our knowledge, our method is the first one that requires only one pass to efficiently compress large SQL workloads while keeping high quality of results of the ISP.

2 Previous Work

The problem of SQL workload compression was first stated by S. Chauduri et al. [14]. They proposed three different algorithms for compressing SQL workloads: a variant of random sampling (WC-PARTSAMP), a variant of the well known K-Medoids algorithm (WC-KMED) and a greedy algorithm removing queries from the workload as long as the results quality constraint is satisfied (WC-ALLPAIRS). Of these algorithms, only the WC-PARTSAMP requires no more than one pass over the input data, but has several shortcomings typical to random sampling methods. Both the compression ratio and the ISP solution quality can vary significantly for different runs of the program. There is also no way of reasonably setting the sampling rate without knowing the characteristics of the workload in advance. For large workloads with lots of similar queries, the sampling rate should be small, but for workloads with many different queries it should be large enough not to miss the important queries. The WC-KMED and WC-ALLPAIRS algorithms don't have these shortcomings and achieve higher compression ratio than the WC-PARTSAMP, but their time complexity is also much higher. Moreover, they need to be given the whole workload in advance. The WC-KMED algorithm invokes the standard k-Medoids (KMED) algorithm several times to guess the best number of clusters. Each invocation of KMED requires scanning the whole dataset many times, until the medoids stop changing. The WC-ALLPAIRS algorithm has even higher complexity because it must calculate the distance between every pair of queries in the workload. This makes these algorithms unsuitable for solving the continuous ISP.

The paper [14] also proposes an asymmetric query distance function used to estimate the quality loss of the ISP results caused by replacing a given query in the workload with another query. Although they put lot of effort into assuring their metric does not underestimate the quality loss, there are several situations

possible, where it actually does. This is caused by the simplified assumption that the index selection tool would select indexes for the columns used in the predicates with the lowest selectivity. As it is true in many cases, it is not true in general. First, using a clustered index for a less selective predicate can be cheaper than using a non-clustered index for a predicate with a higher selectivity, because accessing the tuples pointed by the clustered index usually requires less I/O operations than accessing the same number of tuples pointed by the non-clustered index [15]. The decision what index should be clustered is usually based on many queries in the workload and is not known at the time of the workload compression. Second, automatic index selection tools evaluate not only the benefits of indexes but also their maintenance costs. Thus, the index on the columns of the most selective predicate may have higher maintenance costs and be discarded. This problem applies also to covering indexes, where having a smaller index beneficial to only one query may be a better solution than having a larger index covering two queries but requiring much more maintenance. We propose a different metric that addresses these shortcomings and additionally has a symmetry property and satisfies the triangle inequality.

3 Workload Compression Algorithm

As shown in [14], the workload compression problem can be treated as a special case of a clustering problem, where each data point is a query in the workload and the distance between two points is a function of predicted ISP results quality loss if replacing one query with the other one. The queries in the workload are clustered and afterwards one query in each cluster is retained. The retained queries form the compressed workload. The number of queries in the compressed workload equals the number of the clusters, so the clustering algorithm should create as few clusters as possible to achieve good compression ratio. On the other hand, the queries in each cluster should be similar to each other. The more similar are the queries in each cluster, the less degradation of the ISP results is expected.

The requirement for incremental compression makes classic clustering algorithms k-Medoids, k-Means and hierarchical algorithms AGNES, DIANA, BIRCH, Chameleon [16] unsuitable for the purpose of online ISP. The density based algorithms like DBSCAN [16] are neither applicable, because they don't limit the maximum distance between the queries in each cluster. This could result in a large degradation of the ISP results.

The architecture of our solution is shown in Fig. 1. The incoming workload W is a stream of SQL queries. Each query q has a weight $w(q)$ assigned to it. The workload compression module supplies a compressed workload W' to the input of the tool that selects indexes. The compressed workload contains a subset of queries from W , but they can be assigned different weights w' so that the total estimated cost of W' is near the estimated cost of W .

The workload compression module uses internally a simple yet efficient algorithm that groups incoming queries in clusters (Fig. 2). A *seed query* is the first query added to the cluster. The first incoming query becomes the seed of the first

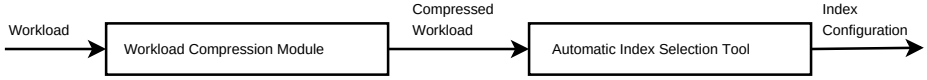


Fig. 1. Automatic index selection improved by workload compression

cluster. For each subsequent incoming query from W , the nearest seed query s is found. If the distance between these two queries exceeds a user-defined limit, then the incoming query forms a new cluster and becomes its seed. Otherwise the query is added to the cluster having seed s . The compressed workload is formed from the cluster seeds. Each cluster seed is assigned a weight evaluated as the sum of the weights of all queries in this cluster. Actually, to improve performance, only the seeds with weights are stored in memory, and instead of adding non-seed queries to the clusters, only the seed weights are updated.

Input: workload W , constraint δ , distance function d , query weights w

Output: compressed workload W' , modified weights w'

1. $W' := \emptyset$
2. For each query $q \in W$ do:
3. Set $w'(q) := w(q)$
4. If $W' = \emptyset$, add q to W' and continue
5. Find the nearest seed query: $s := \arg \min_{q' \in W'} d(q', q)$
6. If $d(s, q) > \delta$, add q to W'
7. Else $w'(s) := w'(s) + w(q)$

Fig. 2. Workload compression algorithm

Note that we do not use the distance function to adjust the weights of the output queries, like it was done in [14]. We argue, this is not needed because of the following reasons:

- Without actually running the index selection tool on both compressed and uncompressed workloads, it is not possible to evaluate the quality loss caused by removing a given query from the workload.
- It is not known in advance, which indexes would be created for the removed query by the index selection tool, so the estimations of the result quality loss may have large errors.
- If the δ constraint is small enough, the queries in each cluster would be very similar to each other and the difference between the estimated costs of the original and compressed workload would be also small.

4 Query Distance Function

The intuition behind the distance function is that the distance between query q_1 and q_2 should be small, if replacing the query q_1 with q_2 in the workload causes small quality loss. For example the queries:

```
SELECT * FROM person WHERE id = 12345
SELECT * FROM person WHERE id = 54321
```

are very similar to each other, assuming they select at most one tuple. Regardless of the index configuration, they will have the same query plans. Leaving one of them out of the compressed workload would cause no quality loss, as they both deliver the same information on the usefulness of the index on the `id` column. However, if the selectivities of the predicates in these queries were different, e.g:

```
SELECT * FROM person WHERE age = 30
SELECT * FROM person WHERE age > 30
```

The queries look similarly, but they contribute different information on the usefulness of the index on the `age` column. The first query would be probably accelerated the most by the hash index on the `age` column. However, this index would not be useful for the second query. Probably the best choice for the second query would be some kind of clustered B-tree index, which could also serve the first query, but the availability of such clustered index is strongly dependent on the other queries in the workload. It is not possible to guess the solution, so it is wise to leave both queries in the compressed workload and let the index selection tool decide. The problem becomes even more complex if queries with more predicates and tables are concerned. The number of useful index configurations for such queries grows exponentially with the number of predicates. Thus, removing any of two queries differing significantly with at least one predicate selectivity may bias the workload against using some good index configurations.

The query distance d between queries q_1 and q_2 is evaluated as follows. If the queries differ in *structure*, that is with anything except constant literals:

$$d(q_1, q_2) = +\infty,$$

else

$$d(q_1, q_2) = \max_{p_1 \in \text{Pred}(q_1), p_2 \in \text{Pred}(q_2), p_1 \sim p_2} \frac{\max\{\text{Sel}(p_1), \text{Sel}(p_2)\}}{\min\{\text{Sel}(p_1), \text{Sel}(p_2)\}} - 1,$$

where:

- $\text{Pred}(q)$ is the set of the selection predicates in the query q
- $\text{Sel}(p)$ is the selectiveness of the predicate p in range from 0 to 1.
- $p_1 \sim p_2$ denotes two corresponding predicates, that differ only with the constant literals.

This metric has several advantages. It is symmetric, satisfies the triangle inequality and is very easy to implement. For most query pairs it is even not needed to evaluate the selectivities of the predicates, because differences in the query structure (set of tables, join predicates, number of selection predicates, column set in the `ORDER BY` or `GROUP BY` clause, etc.) can be easily spotted by shallow text analysis. This is very important, because for large workloads containing lots

of simple queries, the parsing and plan generation process can become a bottleneck. Another advantage is that the proposed metric can leverage the existing support for prepared statements in most modern database systems. The comparison of the structure of queries can be performed at the time of statement preparation, not their execution.

5 Experiments

For the experiments, we used two real-world server side transactional applications: a commercial multiplayer network game with 100,000 users further referred as MG and a mobile web application of one of Polish telecom operators (WA). The MG executed much more update statements than WA, which was mostly read-only (Tab. 1). Besides, MG was a mature application being for over 3 years in production and used 108 tables, while WA application was in its beta-stage and used only 33 tables. Both applications sent EJB-QL queries and employed an object-relational mapping tool to convert them into valid SQL statements. The workloads did not contain any subqueries, however some of the queries required joining up to 6 tables and contained up to 10 selection predicates.

The framework for the workload compression has been build as a standalone application employing an ANTLR generated SQL parser and a histogram based predicate selectivity estimator. The histograms were imported from the statistics gathered by the database optimizer. For some of the queries, we manually compared our predicate selectivity estimation with the estimations made by the database optimizer in the EXPLAIN mode and noticed none or negligible differences.

As the automatic index selection tool we used a prototype tool developed at our institute. The tool can select single and multicolumn B-tree indexes, both clustered or unclustered, and takes index maintenance costs into account. The selection tool uses the same selectivity estimates the workload compression application does.

The measurement of the compression ratio for both workloads shows that the compressed workload size grows very slowly with the size of the input workload (Fig. 3). The index selection tool could not finish the computations in the given 1 hour period for the uncompressed workload of the MG application containing

Table 1. Characteristic of the workloads used in the experiments

Statement type	Share [%]	
	MG	WA
Single-table SELECT	66.95	78.73
Multi-table SELECT	18.93	16.14
Aggregate SELECT	2.84	3.30
INSERT	0.92	1.83
UPDATE	12.71	0.98
DELETE	0.49	2.31



Fig. 3. Compressed workload size as a function of the input workload ($\delta = 0.66$)

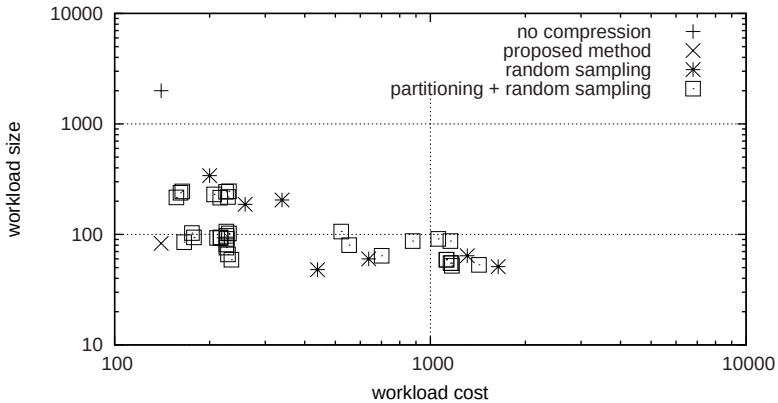


Fig. 4. Compression-ratio vs. ISP results quality for various algorithms for workload of 2000 queries executed by the application MG

25000 queries, but managed to finish the task in less than 2 minutes for the compressed workload.

As stated in Introduction, the workload compression algorithm is useful if both the compression ratio and quality ratio are high. We compared our method with other algorithms that could compress workloads incrementally: the random sampling method, and the random sampling method preceded with partitioning used in [14]. To be able to calculate the quality-loss in a reasonable time, we had to limit the number of queries in the test workloads to 2000. Our compression method resulted in both good compression-ratio and quality-ratio. The results for the MG application are shown in Fig. 4, and the results for WA were very similar. We noticed almost no quality loss ($< 0.1\%$) for both workloads. The other tested methods could achieve sometimes higher compression-ratio than our algorithm but with significant quality-loss. Keeping small quality-loss

required to increase the sampling rate, but it decreased the compression-ratio. As expected, the results of random sampling based methods varied from run to run. In contrast, our method gave stable compression-ratio and quality. Changing the order of queries in the workload did not affect the quality loss and compression ratio by more than 5%.

6 Discussion

The experiments have shown that the presented compression method is useful for compressing transactional workloads and provides acceptable compression ratio and negligible quality loss of index selection results. This enables to deal with orders of magnitude less queries at the input of the index selection tool and to significantly reduce the database tuning time. Besides, the compression ratio can be easily adjusted by the parameter δ . However, it is not the best method when having the whole workload in advance. A k-Medoids variant like the one presented in [14] or some other standard clustering methods [16] can surely achieve better compression ratio within the same quality loss, because they pick the seed queries more carefully. This can lead to either smaller clusters or fewer of them. These algorithms are also usually not sensitive to the order of queries in the workload, unlike our method is.

7 Conclusions

We presented a simple solution of workload compression problem for large transactional database workloads. The main advantage of our method is the possibility to compress workloads incrementally, without storing all queries in memory, which makes our method ideal for usage with online database tuning software. The performed experiments have shown good compression ratio and low quality loss of the method, as opposed to random sampling based approach.

In the future we plan to investigate how the proposed compression method works for complex decision support workloads, e.g. TPC-H standard workload. We predict the compression ratio on such workloads would be worse than for the transactional workloads used in the presented experiments, because the decision support queries usually contain more predicates. Thus, the chances for two queries to differ significantly in at least one predicate selectivity are greater. However, we cannot estimate how much this would affect the overall compression ratio and quality loss without actually measuring them. We are planning to do that as soon as the framework we have built for the predicate selectivity estimation supports subqueries and other structures usually used in the decision support queries e.g. **CASE WHEN**.

Acknowledgements. We would like to thank Marzena Kryszkiewicz for important feedback on the method and experiments.

References

1. Finkelstein, S., Schkolnick, M., Tiberio, P.: Physical database design for relational databases. *ACM Trans. Database Syst.* 13, 91–128 (1988)
2. Ip, M.Y.L., Saxton, L.V., Raghavan, V.V.: On the selection of an optimal set of indexes. *IEEE Trans. Softw. Eng.* 9(2), 135–143 (1983)
3. Whang, K.Y.: Index selection in relational databases. In: FODO, pp. 487–500 (1985)
4. Barcucci, E., Pinzani, R., Sprugnoli, R.: Optimal selection of secondary indexes. *IEEE Trans. Softw. Eng.* 16, 32–38 (1990)
5. Choenni, S., Blanken, H.M., Chang, T.: Index selection in relational databases. In: International Conference on Computing and Information, pp. 491–496 (1993)
6. Chaudhuri, S., Narasayya, V.R.: An efficient cost-driven index selection tool for Microsoft SQL Server. In: VLDB 1997: Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 146–155. Morgan Kaufmann Publishers Inc., San Francisco (1997)
7. Valentin, G., Zulliani, M., Zilio, D.C., Lohman, G., Skelley, A.: DB2 advisor: An optimizer smart enough to recommend its own indexes. In: ICDE 2000: Proceedings of the 16th International Conference on Data Engineering, Washington, DC, USA, p. 101. IEEE Computer Society, Los Alamitos (2000)
8. Zilio, D.C., Zuzarte, C., Lohman, G.M., Pirahesh, H., Gryz, J., Alton, E., Liang, D., Valentin, G.: Recommending materialized views and indexes with IBM DB2 design advisor. In: ICAC 2004: Proceedings of the First International Conference on Autonomic Computing, Washington, DC, USA, pp. 180–188. IEEE Computer Society, Los Alamitos (2004)
9. Elnaffar, S., Powley, W., Benoit, D., Martin, P.: Today's DBMSs: How autonomic are they? In: DEXA 2003: Proceedings of the 14th International Workshop on Database and Expert Systems Applications, Washington, DC, USA, p. 651. IEEE Computer Society, Los Alamitos (2003)
10. Ganek, A.G., Corbi, T.A.: The dawning of the autonomic computing era. *IBM Syst. J.* 42(1), 5–18 (2003)
11. Sattler, K.U., Schallehn, E., Geist, I.: Autonomous query-driven index tuning. In: IDEAS 2004: Proceedings of the International Database Engineering and Applications Symposium (IDEAS 2004), Washington, DC, USA, pp. 439–448. IEEE Computer Society, Los Alamitos (2004)
12. Schnaitter, K., Abiteboul, S., Milo, T., Polyzotis, N.: Colt: continuous on-line tuning. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 793–795. ACM Press, New York (2006)
13. Schnaitter, K., Abiteboul, S., Milo, T., Polyzotis, N.: On-line index selection for shifting workloads. In: ICDE Workshops, pp. 459–468. IEEE Computer Society, Los Alamitos (2007)
14. Chaudhuri, S., Gupta, A.K., Narasayya, V.: Compressing sql workloads. In: SIGMOD 2002: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 488–499. ACM, New York (2002)
15. Garcia-Molina, H., Widom, J., Ullman, J.D.: Database System Implementation. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1999)
16. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2000)

Escaping a Dominance Region at Minimum Cost

Youngdae Kim, Gae-won You, and Seung-won Hwang*

Pohang University of Science and Technology, Korea
{prayer, gwyou, swhwang}@postech.edu

Abstract. Skyline queries have gained attention as an effective way to identify desirable objects that are “not dominated” by another object in the dataset. From market perspective, such objects can be viewed as *marketable*, as each of such objects has at least one competitive edge against all the other objects, or not dominated. In other words, non-skyline objects are not marketable, as there always exists another product excelling in all the attributes. The goal of this paper is, for such non-skyline objects, to identify the cost-minimal enhancement to become a skyline point to gain marketability. More specifically, we abstract this problem as a mixed integer programming problem and develop a novel algorithm for efficiently identifying the optimal solution. Through extensive experiments using synthetic datasets, we show that our proposed framework is both efficient and scalable over extensive experiment settings.

1 Introduction

As Web data extraction technologies evolve, more and more structured data are becoming accessible from Web documents, such as product data automatically extracted from a catalog document published on the Web by merchants. Due to the near-infinite amount of such data, *e.g.*, over 800 million products accessible from products.live.com, there have been efforts to support an effective access to relevant products, from customer’s perspective. To illustrate, Example 1 shows how advanced query semantics such as skyline queries [3] can help customers in such a scenario.

Example 1. Consider a shopping scenario where a customer is looking for a laptop with low price and weight. Considering her preference, one can say that laptop A with lower price and weight than B is more desirable than B , or A “dominates” B . Skyline queries, by identifying the products that are not dominated by any other product (products $p_1, \dots, p_4$ in Fig. 1), identify marketable products.

As illustrated in Example 1, from a customer’s perspective, each skyline product returned from the query corresponds to a *marketable* choice, which outperforms all the other objects in at least one dimension. In contrast, in this paper, we abstract a marketability query from the *merchant* perspective. That is, in the scenario in Example 1, we aim at providing marketability feedbacks, not to customers, but to merchants publishing product catalogs.

* This work was supported by Microsoft Research Asia.

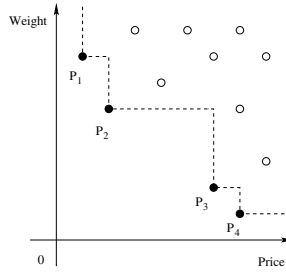


Fig. 1. Skyline (Marketable) products

More specifically, our goal is to provide automatic feedbacks to “non-marketable” merchants whether they are selling some product p that is dominated in the market by some product p' . Such feedbacks are helpful to the provider as p does not have marketability as long as there exists p' in the market. Further, if the cost function for enhancing the product in each attribute is known (*e.g.*, cost of making it lighter by replacing a part with an lighter alternative), feedbacks can be further specified to suggest the cost-minimal enhancement to gain marketability. For example, for the dominated product p to gain marketability, the merchant of p should adjust its attribute values such that at least one of its attribute is better than that of p' .

In particular, we abstract this problem as a mixed integer programming (MIP), which is a linear programming based technique. While there have been off-the-shelf tools for such optimization problem, *e.g.*, CPLEX, applying such tool “as is” incurs a prohibitive cost for identifying the optimal solution, *e.g.*, 780 seconds in 4-dimensional data space from our experiments reported in [8]. In a clear contrast, we devise supporting data structures and algorithms, enabling to answer the same query in 0.2 seconds, without compromising optimality.

2 Related Work

Skylines have been actively studied, pioneered by block nested loop (*BNL*), divide-and-conquer (*D&C*) and B-tree-based algorithms [3]. More recently, nearest neighbor algorithm [4] followed to more effectively prune out dominated objects by partitioning the data with respect to the nearest neighbor objects. Similarly, sort-filter-skyline (*SFS*) [5] algorithm exploits pre-sorting lists to improve existing algorithms.

In a clear contrast to classic skyline queries, our work focuses on (1) how to enhance product objects (2) towards maximizing *marketability*. While this problem has not been studied, the most relevant work is [6] studying queries exploiting not only data attributes, *e.g.*, hotel quality or price, but also *spatial distance*, *e.g.*, how near dominating hotels are. Our work tackles a clearly different problem but shares a slight commonality, in a sense that this work, by adopting our cost function as a distance function, can find a way to gain marketability, though not necessarily the optimal way (as we will empirically show in [8]). In a clear contrast, we aim at finding the cost-optimal way to gain marketability. Another relevant

work [7] considers feature selection from the merchant point of view, while our work clearly distinguishes itself by providing a per-object feedback and the cost-optimal enhancement towards marketability. In particular, this work abstracts the marketability query problem as a linear programming problem, in particular a Mixed-Integer Programming (MIP) problem, which has been actively studied in the area of operation research [9].

3 Preliminaries

Skyline is a set of objects which are not dominated by another object in the dataset. More formally, an object o_1 *dominates* another object o_2 in d -dimensional space if o_1 is no worse than o_2 with respect to every attribute $\forall i : o^i$. That is, in scenarios where smaller attribute values are more desirable, *e.g.*, for attributes such as *price* or *weight*, o_1 dominates o_2 , if $o_1^i \leq o_2^i, \forall i = 1, \dots, d$, and there exists $j, 1 \leq j \leq d$, such that $o_1^j < o_2^j$.

We view this definition of *dominance*, when applied to microeconomic market data, corresponds to *skyline*. That is, product o_1 dominates o_2 in the market, or o_2 does not have marketability as long as o_1 exists, if o_1 is no worse than o_2 in all d attributes. As a result, skyline products can be viewed as *marketable* choices which outperform all the other products in at least one dimension. Meanwhile, the rest, or non-skyline products, are *non-marketable* as there exists a competitor no worse than such products in all the dimensions.

Our goal is to identify, for each non-marketable product p , a marketable counterpart product p' , manufactured by improving p with cost $\mathcal{C}(p, p')$. For instance, for a non-marketable laptop p , one can choose to improve it into a marketable product p' by reducing its weight with the cost overhead of replacing a part with its more expensive but lighter alternative with additional cost $\mathcal{C}(p, p')$. We formally state our goal as follows:

Definition 1 (Marketability Query). *For a non-marketable product p , the marketability query returns a marketable counterpart $\hat{p} = \arg \min_{p'} \mathcal{C}(p, p')$.*

In this paper, the cost function takes the form $\mathcal{C}(p, p') = \sum_{j=1}^d c^j(p^j - p'^j)$ where c^j is the cost enhancing a unit of j -th attribute.

With the problem formally defined, we now visualize this problem in a data space. More specifically, our problem can be viewed as, for a non-skyline point, finding the cost-optimal way to “escape” the region dominated by skyline products, or *dominance region*.

Fig. 2 illustrates the dominance region of skyline set S , abbreviated by DR_S and the *anti-dominance region*, abbreviated by ADR_S , which is not dominated by skyline products. Observe that the boundary line of DR_S is included in the dominance region except the skyline points themselves, as the points on the boundary are dominated by some skyline point in S .

Now our problem can be restated as, for a non-skyline point $x_0 \in DR_S$, finding $\hat{x}$ which satisfies the following. In Equation 1, H denotes the hyper-rectangle with 0 and x_0 as its lower-left and upper-right corners respectively.

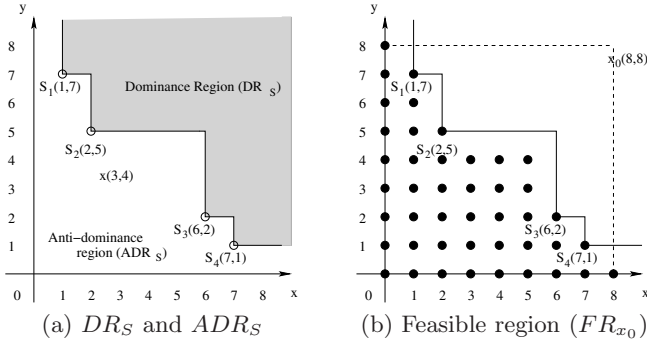


Fig. 2. Dominance/Anti-dominance region and feasible region

$$\hat{x} = \arg \min_{x \in FR_{x_0}} \mathcal{C}(x_0, x), \text{ where } FR_{x_0} = ADR_S \cap H \quad (1)$$

As Equation 1 states, we find an optimal solution among the candidate points in the set FR_{x_0} . The region where candidates can be located for given non-skyline point x_0 is called the *feasible region*, denoted as FR_{x_0} , which we will be our *search space*. Fig. 2(b) illustrates the feasible region, represented by black dots, for the given non-skyline point x_0 in an integer data space. Note that (1) the feasible region consists of finite number of points since the data space consists of integers and (2) the optimal solution satisfying Equation 1 can be found in this feasible region of a finite number of data points.

Hereinafter we assume that our data space consists of integers and we do not consider continuous data space. If the data space is continuous, no optimal solution satisfying Equation 1 can be found since the cost decreases as a point gets closer to the boundary of DR_S , while the boundary itself is not included in our feasible region. Therefore, we need to redefine the optimality of our problem and take different approaches in continuous data space. Due to space limitation, we do not discuss such extension.

4 MIP Modeling

This section presents how we model the problem defined in Section 3. In particular, we divide the problem into the following two sub-problems.

1. Find an optimal point minimizing $\mathcal{C}(x_0, x)$ in the region $FR_{x_0} - S$.
2. Find an optimal point minimizing $\mathcal{C}(x_0, x)$ in the region $S \cap H$.

We then combine the two problems by comparing the costs of the optimal point obtained from each sub-problem and report the one with a cheaper cost. As the second sub-problem can be straightforwardly solved by a nearest neighbor query using our cost function as a distance function, we hereby focus on presenting how to solve the first sub-problem.

In particular, we present the sub-problem 1 as a mixed integer programming (MIP) problem. To convey this idea, we start with presenting the conditions that

hold for the points in $FR_{x_0} - S$. If S consists of a single skyline point s , the region $FR_{x_0} - S$ can be expressed as the union of the sets $\{x : x^j + 1 \leq s^j\}, 1 \leq j \leq d$. Generalizing this for skyline set S consisting of n skyline points, the region $FR_{x_0} - S$ can be represented as follows:

$$FR_{x_0} - S = \bigcap_{i=1}^n \left(\bigcup_{j=1}^d \{x : x^j + 1 \leq s_i^j\} \right) \quad (2)$$

Equation 2 states that point x is in the region $FR_{x_0} - S$ if and only if there exists at least one dimension, say j -th dimension, for each skyline point $s \in S$, such that x^j is less than s^j . Building upon Equation 2, we can express the region $FR_{x_0} - S$ and the objective mathematically as follows:

$$\text{minimize:} \quad \sum_{j=1}^d c^j (x_0^j - x^j) \quad (3)$$

subject to:

$$x^j + 1 \leq s_i^j + M p_{ij} \quad (4)$$

$$\sum_{j=1}^d p_{ij} = d - 1 \quad (5)$$

$$p_{ij} \in \{0, 1\}, x : \text{integer, and } 0 \leq x^j \leq x_0^j, \forall i, \forall j \quad (6)$$

The objective 3 in this MIP model indicates that our objective of minimizing the cost of moving the given non-skyline point x_0 to the point x . Constraints 4, 5, and 6 represent the region in which x can be located, which is identical to region $FR_{x_0} - S$, *i.e.*, this MIP model looks for point x minimizing the objective among the points satisfying the constraints. Constraints 4 and 5 require that for each skyline point s , x have at least one dimension in which the value of x is less than the value of s in that dimension. The value of M is set to have the maximum possible value in the given attribute domain— That is, if M remains in the constraint ($p_{ij} = 1$ for some i and j), that constraint is automatically satisfied regardless of the value of x^j . Constraint 5 forces that for each skyline point there exists only one p_{ij} with value of 0 so that the value of x^j is less than the value of s_i^j in the j th dimension. Constraint 6 restricts x to be located in the hyper-rectangle H with 0 and x_0 as its lower-left and upper-right corners, respectively. Since x should satisfy constraints 4, 5, and 6, our MIP model will identify an optimal point minimizing the cost function $\mathcal{C}(x_0, x)$ among the points in the region $FR_{x_0} - S$.

Summing up, we solve the first and second sub-problem using a MIP model and a nearest neighbor query, respectively, then compare the cost of each solution to report the one with a cheaper cost as our final solution.

A baseline approach to solve the model is to consider all the skyline points in $S \cap H$ are used for constructing our MIP model, which we call *Naive* approach. In a clear contrast, we devise a grid-based cell searching algorithm, which significantly outperforms the *Naive* approach (*e.g.*, reducing the response time from 780 secs to 0.2 sec [8]).

5 Grid-Based Cell Searching Algorithm

As the number of constraints and binary variables in the MIP model in Section 4 is proportional to the number of skyline objects and dimensions, solving the model for a dataset with many skyline objects would incur a prohibitive cost. Since skyline cardinality is exponential to the dimensionality d of dataset, $\Theta((\ln n)^{d-1}/(d-1)! [1])$, the performance of *Naive* approach deteriorates dramatically as dimension increases. Empirical study shows that *Naive* approach incurs a prohibitive cost *e.g.*, 780 seconds in 4-dimensional space.

The main reason of performance degradation of *Naive* approach is that it considers all the skyline points at once that dominate the given non-skyline point to compute the cost-minimal way. However, it turns out that a little preprocessing effort enables us to compute the same optimal solution by considering only a small part of skyline points at once.

Specifically, we precompute the hyper-rectangles H_i 's, $i=1, \dots, m$, that encloses the boundary of dominance region as Fig. 3(a) depicts. Then it is easy to see that the cost $\mathcal{C}(x_0, ur_i)$ where ur_i is the upper-right corner of hyper-rectangle H_i provides the lower bound cost for escaping through the hyper-rectangle H_i from the non-skyline point x_0 .

We now present our approach, which is essentially a best-first search of boundary rectangles, in the ascending order of cost bounds. This can be achieved by considering the hyper-rectangle H_j with the lowest cost bound first, *i.e.*, $\mathcal{C}(x_0, ur_j) \leq \mathcal{C}(x_0, ur_i), \forall i = 1, \dots, m$ and $i \neq j$. We then compute the optimal solution x for escaping the dominance region from the point ur_i within the selected rectangle using the MIP model. The total cost will be $\mathcal{C}(x_0, ur_j) + \mathcal{C}(ur_j, x)$. At each iteration, we record *the best solution* found so far. We repeat this procedure until the next rectangle with the cheapest cost bound $\mathcal{C}(x_0, ur_k)$ exceeds the cost of the best solution found thus far. We can prove that the optimality of this solution is guaranteed [8].

This approach significantly reduces the computational overhead (1) by enabling *early termination* and (2) by considering only the skyline points that dominate ur_i at each iteration, in contrast to *Naive* approach considering all the skyline points, which can be exponential to d .

To implement this framework, we consider the two practical issues in the two following sections respectively.

- **Partitioning:** how to partition the given data space into cells and discard the cells that are guaranteed not to contain the optimal solution.
- **Search:** how to efficiently search for the optimal solution among the promising cells we keep.

Grid Partitioning and Skyline Cell. This section discusses how we divide the given data space into grids. Each cell has side length l , hence the volume of a cell will be l^d . Then we keep only the cells which contain the boundary of the dominance region, called the *boundary cells*, and the rest of cells can be discarded without compromising the optimality of the results. A cell contains

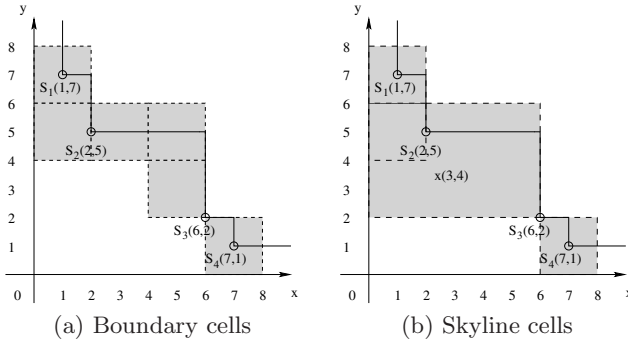


Fig. 3. Boundary and skyline cells

the boundary of the dominance region if and only if the upper-right corner is in DR_S and the lower-left corner is in ADR_S as Fig. 3(a) illustrates.

Among the boundary cells we keep, we only store the *skyline cells* and discard the rest. Skyline cells are the boundary cells whose upper-right corners are not dominated by any other boundary cell's upper-right corner.¹ While calculating the skyline cells, we extend the range of skyline cells to cover the cells it dominates, *i.e.*, to cover the boundary of DR_S . Fig. 3(b) illustrates the skyline cells.

Finding the boundary cells and computing the skyline cells over them can be efficiently implemented by hierarchical recursive partitioning method [8].

Best-first Search. Once the skyline cells are identified, we move on to discuss how to efficiently search for the optimal solution among these cells.

We first discuss how to further prune out the skyline cells with respect to the given non-skyline point x_0 and the skyline cells SCs . Since we only search toward the direction of reducing values of x_0 in each dimension, *i.e.*, into more desirable products for *min* skyline operator, we need to consider only the skyline cells intersected with the hyper-rectangle H with two corner points 0 and x_0 and can discard the rest. By intersecting SCs with H , we obtain the skyline cells to be considered to find the optimal solution. Note that, during the intersection operation, the skyline cells which partially overlap with the H can be chopped to keep only the overlapped partial regions.

After this pruning phase, we now calculate the lower bound cost of each interesting skyline cell, *i.e.*, the cost to reach the upper-right corner of a skyline cell from the non-skyline point x_0 , and store the skyline cell in a heap in the ascending order of the lower bound cost. After that, we iteratively calculate the exact cost for escaping through a specific region, *i.e.*, the boundary of the skyline cell, in an increasing order of its lower bound cost until the lower bound cost of top element in a heap is more expensive than the cheapest cost calculated so far. This procedure is described at line 9-19 in Algorithm 1.

Summing up, our algorithm operates in a best-first manner. We first discard the regions we do not need to consider then visit the rest of the cells in the

¹ Note that, when we compute skyline cells, we use the *max* operator because we use non-skyline point x_0 as the origin.

Algorithm 1. Skyline cell based MIP model solution

Require: non-skyline point x_0 ; the skyline cells SC 's**Ensure:** the optimal solution x

```

1:  $x \leftarrow x_0$ 
2: Construct a hyper-rectangle  $H$  with two corner points 0 and  $x_0$ 
3:  $SC'$ 's  $\leftarrow$  the result of intersecting  $H$  with  $SC$ 's
4: for each skyline cell  $sc$  in  $SC'$  do
5:    $sc.lb \leftarrow$  lower bound cost of  $sc$  from  $x_0$ 
6:    $h.push(sc)$ 
7: end for
8:  $bestCost \leftarrow h.top().lb$ 
9: while  $h.top().lb < bestCost$  do
10:   $sc \leftarrow h.pop()$ 
11:  Set up an MIP model with  $sc$ .skylines
12:   $x' \leftarrow$  the optimal solution of the MIP model
13:   $c \leftarrow$  the cost of the optimal solution
14:   $totalCost \leftarrow sc.lb + c$ 
15:  if  $totalCost < bestCost$  then
16:     $bestCost \leftarrow totalCost$ 
17:     $x \leftarrow x'$ 
18:  end if
19: end while
20: return  $x$ 

```

ascending order of cost bounds. If there exists no further region cheaper than the current cheapest region, we *terminate early*, or else, we examine the next candidate region through which we may escape at the cheapest cost.

References

1. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of ACM* (1978)
2. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest Neighbor Queries. In: *SIGMOD* (1995)
3. Börzsönyi, S., Kossman, D., Stocker, K.: The Skyline Operator. In: *ICDE* (2001)
4. Kossman, D., Ramsak, F., Rost, S.: Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In: *VLDB 2002* (2002)
5. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: *ICDE 2003* (2003)
6. Li, C., Tung, A.K.H., Jin, W., Ester, M.: On Dominating Your Neighborhood Profitably. In: *VLDB 2007* (2007)
7. Miah, M., Das, G., Hristidis, v., Mannila, H.: Standing Out in a Crowd: Selecting Attributes for Maximum Visibility. In: *ICDE 2008* (2008)
8. Kim, Y., You, G.-w., Hwang, S.-w.: Escaping a Dominance Region at Minimum Cost. POSTECH TR, <http://ids.postech.ac.kr>
9. Hiller, F.S., Lieberman, G.J.: *Introduction to Operations Research*. McGraw-Hill, New York (2005)
10. ILOG CPLEX 9.0 User's Manual, <http://www.ilog.com/products/cplex>

Evolutionary Clustering in Description Logics: Controlling Concept Formation and Drift in Ontologies

Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito

Dipartimento di Informatica – Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{fanizzi,claudia.damato,esposito}@di.uniba.it

Abstract. We present a method based on clustering techniques to detect concept drift or novelty in a knowledge base expressed in Description Logics. The method exploits an effective and language-independent semi-distance measure defined for the space of individuals, that is based on a finite number of dimensions corresponding to a committee of discriminating features (represented by concept descriptions). In the algorithm, the possible clusterings are represented as strings of central elements (medoids, w.r.t. the given metric) of variable length. The number of clusters is not required as a parameter; the method is able to find an optimal choice by means of the evolutionary operators and of a fitness function. An experimentation with some ontologies proves the feasibility of our method and its effectiveness in terms of clustering validity indices. Then, with a supervised learning phase, each cluster can be assigned with a refined or newly constructed intensional definition expressed in the adopted language.

1 Introduction

In the context of the Semantic Web (henceforth SW) there is an extreme need of automatizing those activities which are more burdensome for the knowledge engineer, such as ontology construction, matching and evolution. These phases can be assisted by specific learning methods, such as instance-based learning (and analogical reasoning) [5], case-based reasoning [7], inductive generalization [8, 21, 15] and unsupervised learning (clustering) [19, 12] crafted for knowledge bases (henceforth KBs) expressed in the standard representations of the field and complying with their semantics.

In this work, we investigate on the problem of conceptual clustering of semantically annotated resources. The benefits of *conceptual clustering* [24] in the SW context are manifold. Clustering annotated resources enables the definition of new emerging concepts (*concept formation*) on the grounds of the concepts defined in a KB; supervised methods can exploit these clusters to induce new concept definitions or to refine existing ones (*ontology evolution*); intensionally defined groupings may speed-up the task of search and *discovery* [6]; a clustering may also suggest criteria for *ranking* the retrieved resources based on the distance from the centers. Approaches based on incremental learning [9] and clustering have also been proposed [23] to detect *novelties* or track the phenomenon of *concept drift* [25] over time. Most of the clustering methods are based on the application of similarity (or density) measures defined over a fixed set of attributes of the domain objects [16]. Classes of objects are taken as collections

that exhibit low interclass similarity (density) and high intraclass similarity (density). These methods are rarely able to take into account some form of *background knowledge* that could characterize object configurations by means of global concepts and semantic relationships [24]. This hinders the interpretation of the outcomes of these methods that is crucial in the SW perspective which enforces sharing and reusing the produced knowledge to enable semantic interoperability across different KBs and applications.

Conceptual clustering methods can answer these requirements since they have been specifically crafted for defining groups of objects through descriptions based on selected attributes [24]. The expressiveness of the language adopted for describing objects and clusters is extremely important. Related approaches, specifically designed for terminological representations (*Description Logics* [1], henceforth DLs), have recently been introduced [19, 12]. They pursue logic-based methods for attacking the problem of clustering w.r.t. some specific DLs. The main drawback of these methods is that they are language-dependent and cannot scale to standard SW representations that are mapped on complex DLs. Moreover, purely logic methods can hardly handle noisy data.

These problems motivate the investigation on similarity-based clustering methods which can be more noise-tolerant and language-independent. In this paper, an extension of distance-based techniques is proposed. It can cope with the standard SW representations and profit by the benefits of a randomized search for optimal clusterings. The method is intended for grouping similar resources w.r.t. a notion of similarity, coded in a distance measure, which fully complies with the semantics KBs expressed in DLs. The individuals are gathered around cluster centers according to their distance. The choice of the best centers (and their number) is performed through an evolutionary approach [13, 20]. From a technical viewpoint, upgrading existing distance-based algorithms to work on multi-relational representations, like the concept languages used in the SW, requires similarity measures that are suitable for such representations and their semantics. A theoretical problem is posed by the *Open World Assumption* (OWA) that is generally made on the language semantics, differently from the *Closed World Assumption* (CWA) which is standard in other contexts. Moreover, as pointed out in a seminal paper on similarity measures for DLs [3], most of the existing measures focus on the similarity of atomic concepts within hierarchies or simple ontologies. Recently, dissimilarity measures have been proposed for some specific DLs [5]. Although they turned out to be quite effective for specific inductive tasks, they were still partly based on structural criteria which makes them fail to fully grasp the underlying semantics and hardly scale to more complex ontology languages. We have devised a family of dissimilarity measures for semantically annotated resources, which can overcome the aforementioned limitations [10]. Following the criterion of semantic discernibility of individuals, a family of measures is derived that is suitable for a wide range of languages since it is merely based on the discernibility of the input individuals w.r.t. a fixed committee of features represented by a set of concept definitions. In this setting, instead of the notion of *centroid* that characterizes the distance-based algorithms descending from K-MEANS [16], originally developed for numeric or ordinal features, we recur to the notion of *medoids* [18]. The proposed clustering algorithm employs genetic programming as a search schema. The evolutionary problem is modeled by considering populations made up of strings of medoids with different lengths. The medoids are computed according to the semantic

measure mentioned above. On each generation, the strings in the current population are evolved by mutation and cross-over operators, which are also able to change the number of medoids. Thus, this algorithm is also able to suggest autonomously a promising number of clusters. Accordingly, the fitness function is based both on the optimization of a cluster cohesion index and on the penalization of lengthy medoid strings.

We propose the exploitation of the outcomes of the clustering algorithm for detecting the phenomena of concept drift or novelty from the data in the KB. Indeed ontologies evolve over the time (because new assertions are added or because new concepts are defined). Specifically, the occurrence of new assertions can provoke the introduction of new concepts (defined only by the extensions) or can transform existing concepts into more general or more specific ones. We consider the set of new assertions as a candidate cluster and we evaluate its nature w.r.t. the computed clustering model; namely we assess if the candidate cluster is a *normal* cluster, a *new* concept or a *drift* concept. Hence, new concepts could be induced and/or existing ones could be refined.

The remainder of the paper is organized as follows. Sect. 2 presents the basics of the target representation and the semantic similarity measure adopted with the clustering algorithm which is presented in Sect. 3. In Sect. 4 we report an experiment aimed at assessing the validity of the method on some ontologies available in the Web. The utility of clustering in the logic of ontology evolution is discussed in Sect. 5. Conclusions and extensions of the work are examined in Sect. 6.

2 Semantic Distance Measures

In the following, we assume that resources, concepts and their relationship may be defined in terms of a generic ontology language that may be mapped to some DL language with the standard model-theoretic semantics (see the DLs handbook [1] for a thorough reference). In the intended framework setting, a *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. $\mathcal{T}$ is a set of concept definitions. The complexity of such definitions depends on the specific DL language constructors. $\mathcal{A}$ contains *assertions* (ground facts) on *individuals* (domain objects) concerning the current world state, namely: *class-membership* $C(a)$ which means that a is an instance of concept C ; *relations* $R(a, b)$ which means that a is R -related to b . The set of the individuals referenced in the assertions ABox $\mathcal{A}$ will be denoted with $\text{Ind}(\mathcal{A})$. The *unique names assumption* can be made on the ABox individuals¹ therein.

As regards the required inference services, the measure requires performing *instance-checking*, which amounts to determine whether an individual, say a , belongs to a concept extension, i.e. whether $C(a)$ holds for a certain concept C . Note that, differently from the standard DB settings, due to the OWA, the reasoner might be unable to provide a definite answer. Hence one has to cope with this form of uncertainty.

Following some techniques for distance induction in clausal spaces developed in ILP [22], we propose the definition of totally semantic distance measures for individuals in the context of a KB which is also able to cope with the OWA. The rationale of the new measure is to compare individuals on the grounds of their behavior w.r.t. a given set of

¹ Each individual can be assumed to be identified by its own URI, however this is not bound to be a one-to-one mapping.

features, that is a collection of concept descriptions, say $F = \{F_1, F_2, \dots, F_m\}$, which stands as a group of discriminating *features* expressed in the considered DL language. A family of dissimilarity measures for individuals inspired to the Minkowski's distances (L_p) can be defined as follows [10]:

Definition 2.1 (family of dissimilarity measures). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base. Given set of concept descriptions $F = \{F_1, F_2, \dots, F_m\}$, a family of functions $\{d_p^F\}_{p \in \mathbb{N}}$ with $d_p^F : \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) \mapsto [0, 1]$ is defined as follows: $\forall a, b \in \text{Ind}(\mathcal{A})$

$$d_p^F(a, b) := \frac{L_p(\pi(a), \pi(b))}{m} = \frac{1}{m} \left(\sum_{i=1}^m |\pi_i(a) - \pi_i(b)|^p \right)^{\frac{1}{p}}$$

where $p > 0$ and $\forall a \in \text{Ind}(\mathcal{A})$ the projection function π_i is defined by:

$$\pi_i(a) = \begin{cases} 1 & \mathcal{K} \models F_i(a) \\ 0 & \mathcal{K} \models \neg F_i(a) \\ 1/2 & \text{otherwise} \end{cases}$$

The superscript F will be omitted when the set of features is fixed. The functions $\{d_p^F\}_{p \in \mathbb{N}}$ are semi-distance measures (see [10] for more details). The case $\pi_i(a) = 1/2$ occurs when a reasoner cannot give the truth value for a certain membership query. This is due to the OWA normally made in this context. Differently from other DLs measures [4, 17], the presented measure is able to measure dissimilarity between individuals and moreover it does not depend on the constructors of a specific language. It requires only the instance-checking service that is used for deciding whether an individual that is asserted in the KB belongs to a concept extension. Such information can be also pre-computed in order to speed-up the computation of the dissimilarity values and consequently also the clustering process. The underlying idea in the measure definition is that similar individuals should exhibit the same behavior w.r.t. the concepts in F . Here, we make the assumption that the feature-set F represents a sufficient number of (possibly redundant) features that are able to discriminate really different individuals. Preliminary experiments, where the measure has been exploited for instance-based classification (*Nearest Neighbor* algorithm) and similarity search [26], demonstrated the effectiveness of the measure using the very set of both primitive and defined concepts found in the KBs.

However, the choice of the concepts to be included in the committee F is crucial and may be the object of a preliminary learning problem to be solved (*feature selection for metric learning*). We have devised specific optimization algorithms [11] founded in *genetic programming* and *simulated annealing* (whose presentation goes beyond the scope of this work) which are able to find optimal choices of discriminating concept committees. Differently from the goal of the this paper, in [11], the problem of managing novelties and concept drift in an ontology has not been considered. Since the measure is very dependent on the concepts included in the committee of features F , two immediate heuristics can be derived: 1) control the number of concepts of the committee, including especially those that are endowed with a real discriminating power; 2) finding optimal sets of discriminating features, by allowing also their composition employing the specific constructors made available by the DL of choice.

3 Evolutionary Clustering Procedure

Many similarity-based clustering algorithms [16] can be applied to semantically annotated resources stored in a KB, exploiting the measures discussed in the previous section even if, for the best of our knowledge, very few (conceptual) clustering algorithms for coping with DL representations have been proposed in the literature. We focussed on the techniques based on evolutionary methods which are able to determine also an optimal number of clusters, instead of requiring it as a parameter (although the algorithm can be easily modified to exploit this information that greatly reduces the search-space). Conceptual clustering requires also to provide a definition for the detected groups, which may be the basis for the formation of new concepts inductively elicited from the KB. Hence, the conceptual clustering procedure consists of two phases: one that detects the clusters in the data and the other that finds an intensional definition for the groups of individuals detected in the former phase. The first phase of the clustering process is presented in this section. The concept formation process is presented in Sect. 5.2.

The first clustering phase implements a genetic programming learning scheme, where the designed representation for the competing genomes is made up of strings (lists) of individuals of different lengths, with each gene standing as prototypical for a cluster. Specifically, each cluster will be represented by its prototype recurring to the notion of *medoid* [18, 16] on a categorical feature-space w.r.t. the distance measure previously defined. Namely, the medoid of a group of individuals is the individual that has the minimal distance w.r.t. the others. Formally, in this setting:

Definition 3.1 (medoid). *Given a cluster of individuals $C = \{a_1, a_2, \dots, a_n\} \subseteq \text{Ind}(\mathcal{A})$, the medoid of the cluster is defined:*

$$\text{medoid}(C) := \underset{a \in C}{\operatorname{argmin}} \sum_{j=1}^n d(a, a_j)$$

In the proposed evolutionary algorithm, the population will be made up of genomes represented by a list of medoids $G = \{m_1, \dots, m_k\}$ of variable lengths. The algorithm performs a search in the space of possible clusterings of the individuals, optimizing a fitness measure that maximizes the discernibility of the individuals of the different clusters (inter-cluster separation) and the intra-cluster similarity measured in terms of the d_p^F pseudo-metric. On each generation those strings that are considered as best w.r.t. a fitness function are selected for passing to the next generation. Note that the algorithm does not prescribe a fixed length of the genomes (as, for instance in K-MEANS and its extensions [16]), hence it searches a larger space aiming at determining an optimal number of clusters for the data at hand. In the following, a sketch of the algorithm, named ECM, *Evolutionary Clustering around Medoids* is reported.

medoidVector ECM(maxGenerations)

input: maxGenerations: max number of iterations;

output: medoidVector: list of medoids

static: offsprings: vector of generated offsprings

fitnessVector: ordered vector of fitness values

generationNo: generation number

```

INITIALIZE(currentPopulation, popLength)
generationNo = 0
while (generationNo < maxGenerations)
    begin
        offsprings = GENERATEOFFSPRINGS(currentPopulation)
        fitnessVector = COMPUTEFITNESS(offsprings)
        currentPopulation = SELECT(offsprings, fitnessVector)
        ++generationNo
    end
return currentPopulation[0] // fittest genome

```

After the call to the INITIALIZE() function returning (to currentPopulation) a randomly generated initial population of popLength medoid strings, the algorithm essentially consists of the typical generation loop of genetic programming, where a new population is computed and then evaluated for deciding on the best genomes to be selected for survival to the next generation. On each iteration, new offsprings of current best clusterings in currentPopulation are computed. This is performed by suitable genetic operators explained in the following. The fitnessVector recording the quality of the various offsprings (i.e. clusterings) is then updated, and then the best offsprings are selected for the next generation. The fitness of a single genome $G = \{m_1, \dots, m_k\}$ is computed by distributing all individuals among the clusters ideally formed around the medoids in that genome. For each medoid m_i ($i = 1, \dots, k$), let C_i be such a cluster. Then, the fitness is computed by the function:

$$\text{FITNESS}(G) = \left(\lambda(k) \sum_{i=1}^k \sum_{x \in C_i} d_p(x, m_i) \right)^{-1}$$

The factor $\lambda(k)$ is introduced to penalize those clusterings made up of too many clusters that could enforce the minimization in this way (e.g. by proliferating singletons). A suggested value is $\lambda(k) = \sqrt{k+1}$ which was used in the experiments (see Sect. 4).

The loop condition is controlled by the maximal number of generation (the maxGenerations parameter) ensuring that eventually it may end even with a suboptimal solution to the problem. Besides other parameters can be introduced for controlling the loop based on the best fitness value obtained so far or on the gap between the fitness of best and of the worst selected genomes in currentPopulation. Eventually, the best genome of the vector (supposed to be sorted by fitness in descending order) is returned.

It remains to specify the nature of the GENERATEOFFSPRINGS procedure and the number of such offsprings, which may as well be another parameter of the ECM algorithm. Three mutation and one crossover operators are implemented:

```

DELETION( $G$ )  drop a randomly selected medoid:  $G := G \setminus \{m\}, m \in G$ 
INSERTION( $G$ ) select  $m \in \text{Ind}(\mathcal{A}) \setminus G$  that is added to  $G$ :  $G := G \cup \{m\}$ 
REPLACEMENTWITHNEIGHBOR( $G$ ) randomly select  $m \in G$  and replace it with  $m' \in \text{Ind}(\mathcal{A}) \setminus G$  s.t.  $\forall m'' \in \text{Ind}(\mathcal{A}) \setminus G$   $d(m, m') \leq d(m, m'')$ :  $G' := (G \setminus \{m\}) \cup \{m'\}$ 
CROSSOVER( $G_A, G_B$ ) select subsets  $S_A \subset G_A$  and  $S_B \subset G_B$  and exchange them between the genomes:  $G_A := (G_A \setminus S_A) \cup S_B$  and  $G_B := (G_B \setminus S_B) \cup S_A$ 

```

The representation of centers by means of medoids has two advantages. First, it presents no limitations on attributes types, and, second, the choice of medoids is dictated by the location of a predominant fraction of points inside a cluster and, therefore, it is less sensitive to the presence of outliers. In K-MEANS case a cluster is represented by its centroid, which is a mean (usually weighted average) of points within a cluster. This works conveniently only with numerical attributes and can be negatively affected even by a single outlier.

A (10+60) selection strategy has been implemented, with the numbers indicating, resp., the number of parents selected for survival and the number of their offsprings generated employing the mutation operators presented above.

4 Evaluation

The feasibility of the clustering algorithm has been evaluated with an experimentation on KBs selected from standard repositories. For testing our algorithm we preferred using populated ontologies (which may be more difficult to find) rather than randomly generating assertions for artificial individuals, which might have biased the procedure.

4.1 Experimental Setup

A number of different OWL ontologies, selected from various sources², have been considered for the experimentation: FSM, SURFACEWATERMODEL, TRANSPORTATION, NEWTESTAMENTNAMES, and FINANCIAL. Table 1 summarizes details concerning such ontologies. Of course, the number of individuals gives only a partial indication of the number of assertions concerning them which affects both the complexity of reasoning and distance assessment.

Table 1. Ontologies employed in the experiments

Ontology	DL lang.	#concepts	#obj.prop.	#data prop.	#individuals
FSM	$\mathcal{SO}\mathcal{F}(D)$	20	10	7	37
SURFACEWATERMODEL	$\mathcal{ALCO}\mathcal{F}(D)$	19	9	1	115
TRANSPORTATION	$\mathcal{ALC}$	44	7	0	331
NEWTTESTAMENTNAMES	$\mathcal{SHIF}(D)$	47	27	8	676
FINANCIAL	$\mathcal{ALCIF}$	60	16	0	1000

In the computation of the distances between individuals all concepts in the KB have been used for the committee of features, thus guaranteeing meaningful measures with high redundancy. The PELLET reasoner³ was employed to perform the instance-checking that were necessary to compute the projections.

² See the Protégé library: <http://protege.stanford.edu/plugins/owl/owl-library> and the website: <http://www.cs.put.poznan.pl/alawrynowicz/financial.owl>

³ <http://pellet.owldl.com>

The experimentation consisted of 10 runs of the algorithm per knowledge base. The indexes which were chosen for the experimental evaluation were: the *generalized* R-Squared (modRS), the *generalized* Dunn's index, the average Silhouette index, and the number of clusters obtained. We will consider a generic partition $P = \{C_1, \dots, C_k\}$ of n individuals in k clusters. The indexes are formally defined as follows.

The R-Squared index [14] is a measure of cluster separation, ranging in $[0,1]$. Instead of the cluster means, we generalize the measure by computing it w.r.t. their medoids, namely:

$$RS(P) := \frac{SS_b(P)}{SS_b(P) + SS_w(P)}$$

where SS_b is the *between clusters Sum of Squares* defined as $SS_b(P) := \sum_{i=1}^k d(\bar{m}, m_i)^2$ where $\bar{m}$ is the medoid of the whole dataset and SS_t is the *within cluster Sum of Squares* that is defined as $SS_w(P) := \sum_{i=1}^k \sum_{a \in C_i} d(a, m_i)^2$

The generalized Dunn's index is a measure of both compactness (within clusters) and separation (between clusters). The original measure is defined for numerical feature vectors in terms of centroids and it is known to suffer from the presence of outliers. To overcome these limitations, we adopt a generalization of Dunn's index [2] that is modified to deal with medoids. The new index can be defined:

$$V_{GD}(P) := \min_{1 \leq i \leq k} \left\{ \min_{\substack{1 \leq j \leq k \\ i \neq j}} \left\{ \frac{\delta_p(C_i, C_j)}{\max_{1 \leq h \leq k} \{\Delta_p(C_h)\}} \right\} \right\}$$

where δ_p is the Hausdorff distance for clusters derived⁴ from d_p , while the cluster diameter measure Δ_p is defined as $\Delta_p(C_h) := \frac{2}{|C_h|} \sum_{c \in C_h} d_p(c, m_h)$. It is more noise-tolerant w.r.t. the original measure. It ranges in $[0, +\infty[$ and has to be maximized.

The average Silhouette index [18] is a measure ranging in the interval $[-1,1]$, thus suggesting an absolute best value for the validity of a clustering. For each individual $x_i, i \in \{1, \dots, n\}$, the average distance to other individuals within the same cluster $C_j, j \in \{1, \dots, k\}$, is computed: $a_i := \frac{1}{|C_j|} \sum_{x \in C_j} d_p(a_i, x)$ Then the average distance to the individuals in other clusters is also computed: $b_i := \frac{1}{|C_j|} \sum_{x \in C_h}^{h \neq j} d_p(a_i, x)$ Hence, the Silhouette value for the considered individual is obtained as follows:

$$s_i := \frac{(b_i - a_i)}{\max(a_i, b_i)}$$

The average Silhouette value s for the whole clustering is computed: $s := \frac{1}{k} \sum_{i=1}^k s_i$

We also considered the average number of clusters resulting from the repetitions of the experiments on each KB. A stable algorithm should return almost the same number of clusters on each repetition. It is also interesting to compare this number to the one of the primitive and defined concepts in each ontology (see Tab. 1), although this is not a hierarchical clustering method.

⁴ δ_p is defined $\delta_p(C_i, C_j) := \max\{d_p(C_i, C_j), d_p(C_j, C_i)\}$, where $d_p(C_i, C_j) := \max_{a \in C_i} \{\min_{b \in C_j} \{d_p(a, b)\}\}$.

4.2 Results

The experiment consisted in 10 runs of the evolutionary clustering procedure with an optimized feature set (computed in advance). Each run took from a few minutes to 41 mins on a 2.5GhZ (512Mb RAM) machine. These timings include the pre-processing phase needed to compute the distance values between all couples of individuals. The elapsed time for the core clustering algorithm is actually very short (max 3 minutes). The outcomes of the experiments are reported in Tab. 2. For each KB and for each index, the average values observed along the various repetitions is considered. The standard deviation and the range of minimum and maximum values are also reported.

Table 2. Results of the experiments: for each index, average value ($\pm$ standard deviation) and [min,max] interval of values are reported

Ontology	R-Squared	Dunn's	Silhouette	#clusters
FSM	.39 ($\pm$.07)	.72 ($\pm$.10)	.77 ($\pm$.01)	4 ($\pm$.00)
	[.33,.52]	[.69,1.0]	[.74,.78]	[4,4]
SURFACEWATERMODEL	.45 ($\pm$.15)	.99 ($\pm$.03)	.999 ($\pm$.000)	12.9 ($\pm$.32)
	[.28,.66]	[.9,1.0]	[.999,.999]	[12,13]
TRANSPORTATION	.33 ($\pm$.04)	.67 ($\pm$.00)	.975 ($\pm$.004)	3 ($\pm$.00)
	[.26,.40]	[.67,.67]	[.963,.976]	[3,3]
NEWTTESTAMENTNAMES	.46 ($\pm$.08)	.79 ($\pm$.17)	.985 ($\pm$.008)	29.2 ($\pm$ 2.9)
	[.35,.59]	[.5,1.0]	[.968,.996]	[25,32]
FINANCIAL	.37 ($\pm$.06)	.88 ($\pm$ 1.16)	.91 ($\pm$.03)	8.7 ($\pm$.95)
	[.29,.45]	[.57,1.0]	[.87,.94]	[8,10]

The R-Squared index values denotes an acceptable degree of separation between the various clusters. We may interpret the outcomes observing that clusters present a higher degree of compactness (measured by the SS_w component). It should also pointed out that flat clustering penalizes separation as the concepts in the knowledge base are not necessarily disjoint. Rather, they naturally tend to form subsumption hierarchies. Observe also that the variation among the various runs is very limited.

Dunn's index measures both compactness and separation; the rule in this case is *the larger the better*. Results are good for the various bases. These outcomes may serve for further comparisons to the performance of other clustering algorithms. Again, note that the variation among the various runs is very limited, so the algorithm was quite stable, despite its inherent randomized nature.

For the average Silhouette measure, that has a precise range of values, the performance of our algorithm is generally very good, with a degradation with the increase of individuals taken into account. Besides, the largest KB (in terms of its population) is also the one with the maximal number of concepts which provided the features for the metric. Thus in the resulting search space there is more freedom in the choice of the ways to make one individual discernible from the others. Surprisingly, the number of clusters is limited w.r.t. the number of concepts in the KB, suggesting that many individuals gather around a restricted subset of the concepts, while the others are only

complementary (they can be used to discern the various individuals). Such subgroups may be detected extending our method to perform hierarchical clustering.

As regards the overall stability of the clustering procedure, we may observe that the main indices (and the number of clusters) show very little variations along the repetitions (see the standard deviation values), which suggests that the algorithm tends to converge towards clusterings of comparable quality with generally the same number of clusters. As such, the optimization procedure does not seem to suffer from being caught in local minima. However, the case needs a further investigation.

Other experiments (whose outcomes are not reported here) showed that sometimes the initial genome length may have an impact to the resulting clustering, thus suggesting the employment of different randomized search procedures (e.g. again simulated annealing or tabu search) which may guarantee a better exploration of the search space.

5 Automated Concept Evolution in Dynamic Ontologies

In this section we illustrate the utility of clustering in the process of the automated evolution of dynamic ontologies. Namely, clustering may be employed to detect the possible evolution of some concepts in the ontology as reflected by new incoming resources as well as the emergence of novel concepts. These groups of individuals may be successively employed by supervised learning algorithms to induce the intensional description of revised or newly invented concepts.

5.1 Incrementality and Automated Drift and Novelty Detection

As mentioned in the introduction, conceptual clustering enables a series of further activities related to dynamic settings: 1) concept drift [25]: i.e. the change of known concepts w.r.t. the evidence provided by new annotated individuals that may be made available over time; 2) novelty detection [23]: isolated clusters in the search space that require to be defined through new emerging concepts to be added to the knowledge base.

The algorithms presented above are suitable for an online unsupervised learning implementation. Indeed as soon as new annotated individuals are made available these may be assigned to the *closest* clusters (where closeness is measured as the distance to the cluster medoids or to the minimal distance to its instances). Then, new runs of the evolutionary algorithm may yield a modification of the original model (clustering) both in the clusters composition and in their number.

Following [23], the model representing the starting concepts is built based on the clustering algorithm. For each cluster, the maximum distance between its instances and the medoid is computed. This establishes a decision boundary for each cluster. The union of the boundaries of all clusters is the global decision boundary which defines the current model. A new unseen example that falls inside this global boundary is consistent with the model and therefore considered *normal*; otherwise, a further analysis is needed. A single such individual should not be considered as novel, since it could simply represent noise. Due to lack of evidence, these individuals are stored in a short-term memory, which is monitored for the formation of new clusters that might indicate

two conditions: novelty and concept drift. Using the clustering algorithm on individuals in the short-term memory generates candidate clusters. For a candidate cluster to be considered valid, i.e. likely a concept in our approach, the following algorithm can be applied.

(decision, NewClustering) DRIFT_NOVELTY_DETECTION(Model, CCluster)

input: Model: current clustering; CandCluster: candidate cluster;

output: (decision, NewClustering);

$m_{CC} := \text{medoid}(\text{CandCluster});$

for each $C_j \in \text{Model}$ **do** $m_j := \text{medoid}(C_j);$

$d_{\text{overall}} := \frac{1}{|\text{Model}|} \sum_{C_j \in \text{Model}} \left(\frac{1}{|C_j|} \sum_{a \in C_j} d(a, m_j) \right);$

$d_{\text{candidate}} := \frac{1}{|\text{CandCluster}|} \sum_{a \in \text{CandCluster}} d(a, m_{CC});$

if $d_{\text{overall}} \geq d_{\text{candidate}}$ **then** // valid candidate cluster

begin

$\overline{m} := \text{medoid}(\{m_j \mid C_j \in \text{Model}\});$ // global medoid

$d_{\text{max}} := \max_{m_j \in \text{Model}} d(\overline{m}, m_j);$

if $d(\overline{m}, m_{CC}) \leq d_{\text{max}}$ **then**

return (drift, replace(Model, CandCluster))

else return (novelty, Model $\cup$ CandCluster)

end

else return (normal, integrate(Model, CandCluster))

The candidate cluster CandCluster is considered valid⁵ for drift or novelty detection when the average mean distance between medoids and the respective instances for all clusters of the current model is greater than the average distance of the new instances to the medoid of the candidate cluster. Then a threshold for distinguishing between concept drift and novelty is computed: the maximum distance between the medoids of the model and the global one⁶. When the distance between overall medoid and the medoid of the candidate cluster exceeds the maximum distance then the case is of concept drift and the candidate cluster is merged with the current model. Otherwise (novelty case) the clustering is simply extended. Finally, when the candidate cluster is made up of normal instances these can be integrated by assigning them to the closest clusters.

The main differences from the original method [23], lie in the different representational setting (simple numeric tuples were considered) which allows for the use of off-the-shelf clustering methods such as k-MEANS [16] based on a notion of centroid which depend on the number of clusters required as a parameter. In our categorical setting, medoids substitute the role of medoids and, more importantly, our method is able to detect an optimal number of clusters autonomously, hence the influence of this parameter is reduced.

⁵ This aims at choosing clusters whose density is not lower than that of the model.

⁶ Clusters which are closer to the boundaries of the model are more likely to appear due to a drift occurred in the normal concept. On the other hand, a validated cluster appearing far from the normal concept may represent a novel concept.

5.2 Conceptual Clustering for Concept Formation

The next step may regard the refinement of existing concepts as a consequence of concept drift or the invention of new ones to account for emerging clusters of resources. The various cluster can be considered as training examples for a supervised algorithm aimed at finding an intensional DL definition for one cluster against the counterexamples, represented by individuals in different clusters [19, 12].

Each cluster may be labeled with an intensional concept definition which characterizes the individuals in the given cluster while discriminating those in other clusters [19, 12]. Labeling clusters with concepts can be regarded as a number of supervised learning problems in the specific multi-relational representation targeted in our setting [15]. As such it deserves specific solutions that are suitable for the DL languages employed. A straightforward solution may be found, for DLs that allow for the computation of (an approximation of) the *most specific concept* (msc) and *least common subsumer* (lcs) [1] (such as $\mathcal{ALC}$). The first operator, given the current knowledge base and an individual, provides (an approximation of) the most specific concept that has the individual as one of its instances. This would allow for lifting individuals to the concept level. The second operator computes minimal generalizations of the input concept descriptions. Indeed, concept formation can be cast as a supervised learning problem: once the two clusters at a certain level have been found, where the members of a cluster are considered as positive examples and the members of the dual cluster as negative ones. Then any concept learning method which can deal with this representation (and semantics) may be utilized for this new task. Given these premises, the learning process can be described through the following steps:

let C_j be a cluster of individuals

1. **for each** individual $a_i \in C_j$
do compute $M_i := \text{msc}(a_i)$ w.r.t. $\mathcal{A}$;
2. **let** $\text{mscs}_j := \{M_i \mid a_i \in C_j\}$;
3. **return** $\text{lcs}(\text{mscs}_j)$

As an alternative, more complex algorithms for learning concept descriptions expressed in DLs may be employed such as YINYANG [15] or other systems based on refinement operators [21]. Their drawback is that they cannot deal with the most complex DL languages. The concepts resulting from conceptual clustering can be used for performing weak forms of abduction that may be used to update the ABox; namely, the membership of an individual to a cluster assessed by means of the metric, may yield new assertions that do not occur in the ABox may be added (or presented to the knowledge engineer as candidates to addition). Induced assertions coming for newly available individuals may trigger further supervised learning sessions where concepts are refined by means of the aforementioned operators.

6 Conclusions and Extensions

This work has presented a framework for evolutionary conceptual clustering that can be applied to standard relational representations for KBs in the SW context. Its intended

usage is for discovering interesting groupings of semantically annotated resources and can be applied to a wide range of concept languages. Besides, the induction of new concepts may follow from such clusters, which allows for accounting for them from an intensional viewpoint. The method exploits a dissimilarity measure that is based on the undelying resource semantics w.r.t. a committee of features represented by a group of concept descriptions in the chosen language. A preliminary learning phase, based on randomized search, can be used to optimize the choice of the most discriminating features. The evolutionary clustering algorithm is an extension of distance-based clustering procedures employing medoids as cluster prototypes so to deal with complex representations of the target context. Variable-length strings of medoids yielding different partitions are searched guided by a fitness function based on cluster separation. The algorithm can also determine the length of the list, i.e. an optimal number of clusters.

As for the metric induction part, a promising research line, for extensions to match-making, retrieval and classification, is *retrieval by analogy* [5]: a search query may be issued by means of prototypical resources; answers may be retrieved based on local models (intensional concept descriptions) for the prototype constructed (on the fly) based on the most similar resources. The presented algorithm may be the basis for the model construction activity. The distance measure may also serve as a ranking criterion. The natural extensions of the clustering algorithm that may be foreseen are towards incrementality and hierarchical clustering. The former may be easily achieved by assigning new resources to their most similar clusters, and restarting the whole algorithm when some validity measure crosses a given threshold. The latter may be performed by wrapping the algorithm within a level-wise procedure starting with the whole dataset and recursively applying the partitive method until a criterion based on quality indices determines the stop. This may produce more meaningful concepts during the next supervised phase. Better fitness functions may be also investigated for both distance optimization and clustering. For instance, some clustering validity indices can be exploited in the algorithm as measures of compactness and separation.

References

- [1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
- [2] Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics* 28(3), 301–315 (1998)
- [3] Borgida, A., Walsh, T.J., Hirsh, H.: Towards measuring similarity in description logics. In: Horrocks, I., Sattler, U., Wolter, F. (eds.) *Working Notes of the International Description Logics Workshop*, Edinburgh, UK. *CEUR Workshop Proc.*, vol. 147 (2005)
- [4] d'Amato, C., Fanizzi, N., Esposito, F.: A dissimilarity measure for *ALC* concept descriptions. In: *Proceedings of the 21st Annual ACM Symposium of Applied Computing, SAC2006*, Dijon, France, vol. 2, pp. 1695–1699. ACM, New York (2006)
- [5] d'Amato, C., Fanizzi, N., Esposito, F.: Reasoning by analogy in description logics through instance-based learning. In: Tummarello, G., et al. (eds.) *Proc. of Workshop on Semantic Web Applications and Perspectives, SWAP 2006*. *CEUR*, vol. 201 (2006)
- [6] d'Amato, C., Staab, S., Fanizzi, N., Esposito, F.: Efficient discovery of services specified in description logics languages. In: *Proc. of the ISWC Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web* (2007)

- [7] d'Aquin, M., Lieber, J., Napoli, A.: Decentralized case-based reasoning for the Semantic Web. In: Gill, Y., et al. (eds.) Proc. of the 4th Int. Semantic Web Conf., ISWC 2005. LNCS, vol. 3279, pp. 142–155. Springer, Heidelberg (2005)
- [8] Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 441–455. Springer, Heidelberg (2004)
- [9] Esposito, F., Ferilli, S., Fanizzi, N., Basile, T.M.A., Di Mauro, N.: Incremental learning and concept drift in INTHELEX. Jour. of Intelligent Data Analysis 8(1/2), 133–156 (2004)
- [10] Fanizzi, N., d'Amato, C., Esposito, F.: Induction of optimal semi-distances for individuals based on feature sets. In: Calvanese, D., et al. (eds.) Working Notes of the 20th International Description Logics Workshop, DL 2007. CEUR, vol. 250 (2007)
- [11] Fanizzi, N., d'Amato, C., Esposito, F.: Randomized metric induction and evolutionary conceptual clustering for semantic knowledge bases. In: Silva, M.J., et al. (eds.) Proc. of the 16th ACM Conf. on Information and Knowledge Management, pp. 51–60. ACM, New York (2007)
- [12] Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Concept formation in expressive description logics. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 99–110. Springer, Heidelberg (2004)
- [13] Ghoeil, A., Fogel, D.B.: Discovering patterns in spatial data using evolutionary programming. In: Koza, J.R., et al. (eds.) Genetic Programming 1996: Proc. of the 1st Annual Conf., pp. 521–527. MIT Press, Cambridge (1996)
- [14] Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2-3), 107–145 (2001)
- [15] Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence 26(2), 139–159 (2007)
- [16] Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
- [17] Janowicz, K.: Sim-dl: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCN}$ in geographic information retrieval. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1681–1692. Springer, Heidelberg (2006)
- [18] Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, Chichester (1990)
- [19] Kietz, J.-U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning 14(2), 193–218 (1994)
- [20] Lee, C.-Y., Antonsson, E.K.: Variable length genomes for evolutionary algorithms. In: Whitley, L., et al. (eds.) Proc. of the Genetic and Evolutionary Computation Conference, GECCO 2000, p. 806. Morgan Kaufmann, San Francisco (2000)
- [21] Lehmann, J.: Concept learning in description logics. Master's thesis, Dresden University of Technology (2006)
- [22] Sebag, M.: Distance induction in first order logic. In: Džeroski, S., Lavrač, N. (eds.) ILP 1997. LNCS, vol. 1297, pp. 264–272. Springer, Heidelberg (1997)
- [23] Spinosa, E.J., de Leon, A.P., de Carvalho, F., Gama, J.: OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams. In: Proc. of the Annual ACM Symposium of Applied Computing, vol. 1, pp. 448–452. ACM, New York (2007)
- [24] Stepp, R.E., Michalski, R.S.: Conceptual clustering of structured objects: A goal-oriented approach. Artificial Intelligence 28(1), 43–69 (1986)
- [25] Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)
- [26] Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search – The Metric Space Approach. Advances in database Systems. Springer, Heidelberg (2007)

Model–Driven, View–Based Evolution of Relational Databases^{*}

Eladio Domínguez¹, Jorge Lloret¹, Ángel L. Rubio²,
and María A. Zapata¹

¹ Dpto. de Informática e Ingeniería de Sistemas.
Facultad de Ciencias. Edificio de Matemáticas.
Universidad de Zaragoza. 50009 Zaragoza. Spain
`{noesis,jlloret,mazapata}@unizar.es`

² Dpto. de Matemáticas y Computación. Edificio Vives.
Universidad de La Rioja. 26004 Logroño. Spain
`arubio@dmc.unirioja.es`

Abstract. Among other issues, database evolution includes the necessity of propagating the changes inside and between abstraction levels. There exist several mechanisms in order to carry out propagations from one level to another, that are distinguished on the basis of when and how the changes are performed. The strict mechanism, which implies the immediate realization of modifications, is a time-consuming process. In this paper we propose a solution that is closer to the lazy and logical mechanisms, in which changes are delayed or not finally realized, respectively. This solution makes use of the notion of view. The use of views allows the data not to be changed if it is not necessary and facilitates carrying out changes when required.

1 Introduction

There are two main interrelated issues that are very frequently highlighted in existing database evolution literature: information consistency and propagation of modifications. For example, both issues are described in [10] and included within a more general activity called *managing core schema evolution*. Information consistency is concerned with the crucial point of ensuring the lossless of any semantic information when an evolution task is carried out. Propagation of modifications must be considered along two dimensions with regard to abstraction layers. On the one hand, within a *horizontal* dimension the changes in some part of a schema can (and usually must) be conveyed to other parts of the schema. On the other hand, within a *vertical* dimension the changes in a schema situated in an abstraction layer must be propagated to the corresponding schema in other abstraction layers (for instance, from the relational level downwards to the extensional level).

^{*} This work has been partially supported by DGI, project TIN2005-05534, by the Ministry of Industry, Tourism and Commerce, project SPOCS (FIT-340001-2007-4), by the Government of Aragon and by the European Social Fund.

In [2] we have presented a proposal in which information consistency is preserved by using a database evolution architecture, called MeDEA, framed in a forward engineering context. This architecture has been developed from a model-driven perspective, since it has been recognized that schema evolution is one of the applications of model management [1]. With respect to propagation of modifications, different authors classify several ways of carrying out propagations. For instance, Roddick in [8] distinguishes three kinds of *data conversion mechanisms*: strict, lazy and logical. In the *strict* mechanism changes in the schema are propagated to the data immediately. In the *lazy* mechanism, routines that can perform the changes in the data are assembled, but they are used only when required. Lastly, a system of screens is created to translate the data to the appropriate format at access time (without converting data) in the *logical* mechanism. Another classification is proposed in [7] where screening, conversion and filtering techniques are described. Although this second classification is tied to object-based systems, a rather close parallel can be drawn between the *screening* technique and the lazy data conversion mechanism, the *conversion* technique and the strict mechanism, and the *filtering* technique and the logical mechanism. Our above-mentioned architecture, as presented in [2], uses the strict data conversion mechanism.

In any case, there exists a ‘common line’ in the literature that states that the best solution for the propagation of modifications implies making a balance between avoiding the modification of data until it is indispensable and not increasing excessively the complexity of the database. As a step to obtain such a solution, in this paper we propose a new application of our architecture in which views are used. The use of views allows the modification of data not to be initially realized, and therefore, our approach fits into the logical procedure. A noteworthy contribution of this idea is that when, for some reason, the modifications have to be translated into the physical database, this task is facilitated since views *codify* the changes undergone at the logical level. In this sense, our adopted solution is closer to the lazy mechanism and the screening technique.

The structure of the paper is as follows. Section 2 first reviews the basic ideas of our architecture for database evolution, and then presents the adaptation of the architecture to the view-based approach. In Section 3 we discuss the effect of the conceptual changes when a view-based approach is followed. We devote Section 4 to detailing the technical aspects of the view based propagation algorithm. Finally, related work, some conclusions and future work appear in Sections 5 and 6.

2 View-Based Database Evolution

2.1 Brief Description of MeDEA

MeDEA is a MEtamodel-based Database Evolution Architecture we have presented in [2]. In this section, we describe briefly the meaning and purpose of the components of MeDEA (see [2] for details).

MeDEA has four components (see Figure 1): *conceptual component*, *translation component*, *logical component* and *extensional component*. The *conceptual component* captures machine-independent knowledge of the real world. In this work, it deals with EER schemas. The *logical component* captures tool-independent knowledge describing the data structures in an abstract way. In this paper, it deals with schemas from the relational model by means of standard SQL. The *extensional component* captures tool dependent knowledge using the implementation language. Here, this component deals with the specific database in question, populated with data, and expressed in the SQL of Oracle. One of the main contributions of our architecture is the *translation component*, that not only captures the existence of a transformation from elements of the conceptual component to others of the logical one, but also stores explicit information about the way in which specific conceptual elements are translated into logical ones.

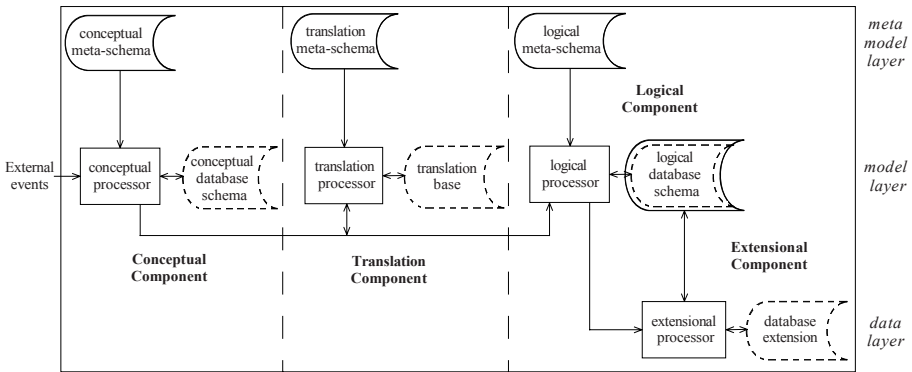


Fig. 1. MeDEA Database Evolution Architecture

It must be noted that three different abstraction levels are involved in the architecture. On the one hand, the (meta-)schemas of the three former components are situated at the most abstract level (metamodel layer according to [14]) and, on the other hand, the information base which stores the population of the database is situated at the least abstract level (user data layer [14]). All the other elements are situated at the model layer [14].

In order to apply MeDEA to the specific situation of this paper (that is, to EER and relational schemas) nine artifacts must be established: three metamod- els (EER, translation and relational), a translation algorithm from EER schemas to relational schemas, three sets of evolution transformations (one set for each metamodel), one set of propagation rules and one set of correspondence rules.

Once these nine artifacts are given (see Section 2.2) the way of working of the architecture is as follows: given an EER schema in the conceptual component, the translation algorithm is applied to it and creates: (1) the logical database schema, (2) a set of elementary translations in the translation component and (3) the physical database schema.

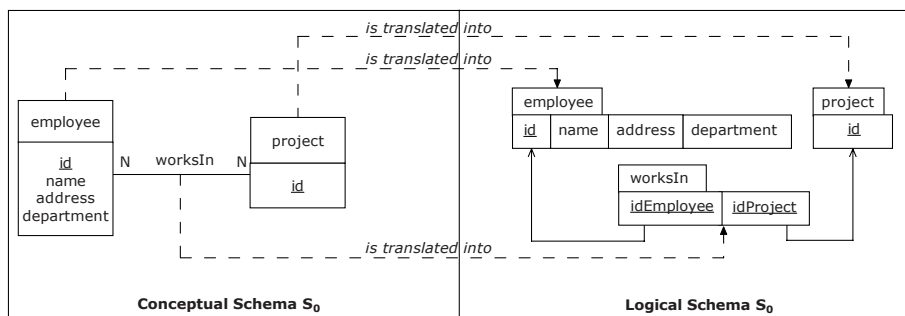


Fig. 2. Conceptual and logical schemas

For instance, as running example, we will consider an initial EER schema S_0 (see Figure 2) with employees and projects. Each employee has an **id**, **name**, **address** and **department** where (s)he works. Each project has an **id** and many employees can work in many projects. The application of the translation algorithm to this EER schema gives rise to: (1) the relational schema with three relation schemas shown in Figure 2, (2) elementary translations storing which EER elements are translated into relational elements (some of the generated elementary translations are represented as dashed lines in Figure 2) and (3) the SQL sentences of Oracle that create the relational schema.

Then, the physical database is populated with data. For various reasons, the data structure may need to be changed. In this case, the data designer must issue the appropriate evolution transformations to the EER schema. The propagation algorithm propagates the changes to the other components. To be precise, the EER schema changes are propagated to the translation and relational components by means of the propagation rules. Afterwards, the relational schema changes are propagated to the Oracle database by means of the correspondence rules.

In Sections 3 and 4, an evolution example, using the EER schema S_0 as starting point, is shown. Four evolution transformations are issued which are propagated to the relational schema, the physical schema and the data.

2.2 Application of MeDEA Using a View-Based Evolution Approach

In [2] we applied MeDEA to EER and relational schemas following a strict data conversion mechanism. The main contribution of this paper is the application of this architecture using a lazy and logical data conversion mechanism, so that the changes are delayed as far as possible. In order to reach this goal, views are used for evolving relational databases whenever the conceptual changes allow this to be done, avoiding the overhead of strict data conversion. However, if the number of views is increased too much, query processing performance can be downgraded.

This change in the way of managing the evolution processes has led us to change some of the artifacts proposed in [2]. To be precise, the EER meta-model, the translation algorithm and the set of conceptual changes remain the same. It should be noted that, during the execution of the translation algorithm, the user may choose the translation rules which are applied to each element of the EER schema giving place to relational structures. For example, for relationship types the user can choose, among others, between the **relSchemaPerRelType** translation rule ('a relation schema for each relationship type') or **relSchemaPerNNRelType** translation rule ('a relation schema for the N-N relationship type and foreign key for other cardinalities of the relationship'). Additional details about the unchanged artifacts can be found in [2].

The rest of the artifacts proposed in [2] have been changed in order to allow us to perform a lazy and logical data conversion mechanism. In particular, the relational metamodel has been modified including in it the concept of **view** and distinguishing between **base** and **derived** attributes (see Figure 3). A view is described by its **name**, a **predicate** and a **checkOption**. Let us notice that a view can be based on relation schemas and views, and that its attributes can be base or derived attributes. These changes in the logical metamodel oblige us to include new logical transformations regarding the new concepts (see Table 1).

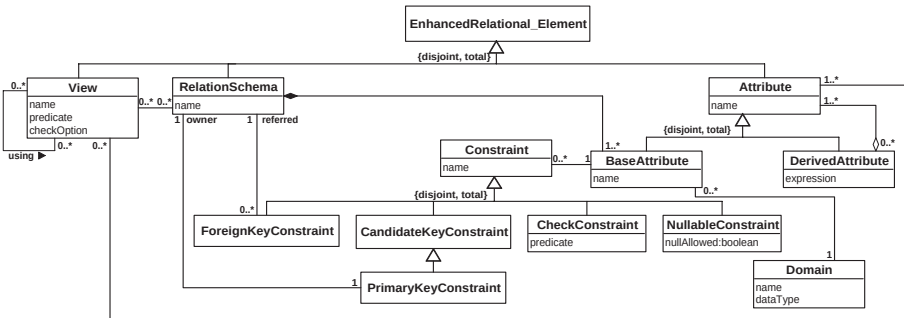


Fig. 3. Graphical representation of the relational Metamodel

The modification of the logical metamodel also implies the inclusion in the translation component of new types of elementary translations. In this way, the **EntityType2View** type reflects the translation of an entity type into a view and the **Attribute2ViewAttribute** type reflects the translation of a conceptual attribute into an attribute of a view. Moreover, the **RelType2View** type reflects the translation of a relationship type into a view. Here we have a constraint according to which for each entity or relationship type there is only one relation schema or view in the logical schema. As a consequence, there is only one elementary translation for each entity or relationship type.

The rest of the paper is devoted to studying the effects of the conceptual changes, to explain the view based propagation algorithm and the propagation rules, bringing to light the advantages of this new proposal.

Table 1. New logical transformations

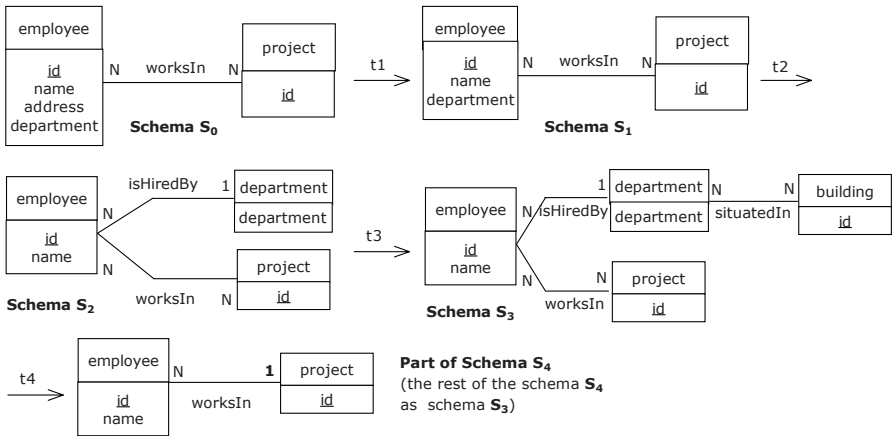
createView(*N*: name; [*LR*: listOfRelationSchemas]; [*LV*: listOfViews]; *LA*: listOfAttributes; *P*: predicate; *CO*: checkOption)
createDerivedAttribute(*N*: name; *LA*: listOfAttributes; *E*:expression)
deleteView(*V*: view)
deleteDerivedAttribute(*DA*: derivedAttribute)
renameView(*V*: view; *N*: name)
renameDerivedAttribute(*DA*: derivedAttribute; *N*: name)
includeRelationSchemaInView(*V*: view; *R*: relationSchema)
includeViewInView(*V*: view; *W*: view)
includeAttributeInView(*V*: view; *A*: attribute)
includeAttributeInDerivedAttribute(*DA*: derivedAttribute; *A*: attribute)
removeRelationSchemaInView(*V*: view; *R*: relationSchema)
removeViewInView(*V*: view; *W*: view)
removeAttributeInView(*V*: view; *A*: attribute)
removeAttributeInDerivedAttribute(*DA*: derivedAttribute; *A*: attribute)
changePredicate(*V*: view; *P*: predicate)
changeCheckOption(*V*: view; *CO*: checkOption)
changeExpression(*DA*: derivedAttribute; *E*: expression)

3 Study of the Effects of the Conceptual Changes

For strict database conversion, we have proposed in [2] a propagation algorithm. Its input is a set of conceptual transformations which change the conceptual schema. Its output is a set of SQL sentences which add, drop or modify tables, columns and constraints as well as a set of data load procedures. The SQL sentences and data load procedures are executed against the physical database so that the new physical database schema conforms with the new conceptual schema.

With the aim of applying the idea of delaying as far as possible the strict conversion of data, we have studied how the propagation algorithm should be modified (giving place to the *view based propagation algorithm*). For this purpose, we have considered the set of MeDEA conceptual transformations which can be applied to a conceptual schema and we have determined how to change the effect of these transformations in the components of the architecture when the concept of view is used. The modifications we have found have been incorporated into the view based propagation algorithm which we describe in the next section.

So as to illustrate our study, we will use the evolution example of Figure 4 in which, starting from the schema S_0 , the attribute **address** is deleted, the attribute **department** is transformed into an entity type **department**, a new entity type **building** and a new relationship type **situatedIn** between **department** and **building** are set. Finally, the cardinality of the relationship type **worksIn** is changed and an employee can work at most in one project.



```

t1: deleteAttribute(employee.address)
t2: turnAttributeIntoEntityType(employee.department,department, true, 'isHiredBy')
t3: createEntityType('building', true)
    createAttribute('building.id',INTEGER)
    addAttributeToEntityType(building.id,building)
    createKeyOfStrongEntityType(building,(id))
    createRelationshipTypeAndRoles('situatedIn',(department, building), null,((0,N),(0,N)), true)
t4: changeCardinalityOfRole(rEmployeeWorksIn, (0,1))

```

Fig. 4. Evolved schemas

Revision of the effects of the conceptual transformations. We briefly describe the effects of some of the conceptual changes and we only describe in depth the transformation for adding a relationship type and for changing the cardinality of a role.

Add an attribute to an entity type. If the entity type is transformed into a relation schema, add an attribute to the relation schema. If the entity type is transformed into a view, add the corresponding attribute to the original relation schema of the entity type as well as to the view.

Delete an attribute from an entity type. Mark as deleted the elementary translations of the attribute. Add a new view or modify an existing one in such a way that the corresponding attribute does not appear.

Add an entity type. Create a new relation schema for the entity type.

Drop an entity type. Marked as deleted the elementary translations of the elements of the entity type as well as those of the relationship types in which the entity type participates.

Add a relationship type. The transformation is `createRelationshipTypeAndRoles(n,le,ln,lc,ident?)` and it creates a new relationship type with name *n*, the list *le* of participants, the list *ln* of names of roles, the list *lc* of cardinalities and information about whether the relationship type is an identifying

Table 2. Representations of relationship types

Case	e	f	rt	Example
	has been transformed into...		is transformed into...	
1	r1	r2	r3, new relation schema or relation schema from one of the participants	transf. t_3
2	v1	r2	r3, new relation schema	transf. t_4
3	v1	r2	v3, view based on v1	
4	v1	r2	r3=r2	
5	v1	v2	r3, new relation schema	transf. t_2
6	v1	v2	v3, view based on v1 or v2	

relationship type. Let us suppose that the participants are **e** and **f**. The different representations of the relationship type are shown in Table 2 where **vx** stands for view and **rx** stands for relation schema. Each one of these cases is contemplated in at least one propagation rule.

The first case of Table 2 will be treated by a propagation rule of the initial propagation algorithm because both of the participants have been transformed into relation schemas and no views are involved. For the rest of the cases, new propagation rules are defined, which contemplate the fact that at least one of the participants is transformed into a view. These rules represent the relationship types as a relation schema (case 2, 4 and 5) or as a view (cases 3, 6).

The first situation (relationship type as a relation schema) happens when we need to augment the information capacity of the schema. For example, in transformation t_3 , we add a new relationship type **situatedIn** for having the information about the departments and buildings where they are located. As the relationship type is N–N, we need a new relation schema to store these relationships.

The second situation (relationship type as a view) appears when we can embed the relationship type into the base relation schema of some of the participants. For example, in transformation t_2 , the relationships between the employees and the projects can be seen through a view on relation schema **employee**. In this respect, we have found two cases:

1. The view for the relationship type is the same as the view for one of the participants, but adequately modified. For example, this happens in transformation t_4 where the view for the relationship type **worksIn** is the same as the view for the entity type **employee**.
2. The view for the relationship type is based on the view for one of the participants but does not modify this view. For example, this happens in transformation t_2 where the view for the entity type **department** is built on the view for the entity type **employee**.

Drop a relationship type. Mark as deleted the elementary translations of the elements of the relationship type.

Change the cardinality of a relationship type. If we change the cardinality of a relationship type from N-1 to N-N two situations can appear. If previously there is a relation schema for the relationship type, no change must be made other than modifying the elementary translation for the relationship type. The new logical element for the relationship type is the previously existing relation schema. Otherwise, the information capacity of the logical schema must be increased and a new relation schema must be created. Views can not be used in either of the two situations.

However, for changing the cardinality of a relationship type from N-N to N-1 when the data allow this to be done, we can create a view for the relationship type and add a unique constraint in the base relation schema of the relationship type to cope with the new cardinality constraint. Database operations continue to be done on the original relation schema of the N-N relationship type. An example can be seen in transformation t_4 .

Remarks. When a conceptual transformation drops a conceptual element, the corresponding elementary translations are marked as deleted but the relational structures are not dropped. The corresponding columns are no longer accessible from the conceptual level, and the columns or tables will be dropped when restructuring database tasks are performed.

The restructuring consists of changing the logical schema in such a way that only tables are left and the consistency between levels is kept. The key point here, and one of the most noteworthy contributions of this paper, is that the use of view converts this task into an easy one. The reason is that the views keep a memory of the changes undergone by the logical schema. Then, creating a new table for an entity or relationship type will basically consist of transforming the view into a table.

For example, the view `vEmployee`, after transformation t_4 , is gathering the information of the entity type `employee(attributes id, name)`, of the relationship type `isHiredBy` (by means of the attributes `id, department`) and of the relationship type `worksIn` (by means of the attributes `id, idProject`).

When this view has to be transformed into a table, two steps related with employees must be made at the physical level:

1. The `department` column values are replaced by identifiers for the departments
2. The view `vEmployee` is easily transformed into a table with a SQL sentence like `CREATE TABLE employee2 AS SELECT * FROM vEmployee`

The analysis we have just done is integrated into the existing propagation rules or into new ones, giving place to the view based propagation algorithm (see next section).

4 View Based Propagation Algorithm

In our new perspective of delaying the conversion of the data, the changes in the EER schema are propagated to the physical database by means of the view

elementary_translation				Legend for elementary translations
id	type	conceptual_element	logical_item	
 1	RelType2RelSchema	worksIn	worksIn	 marked as deleted
 2	Attribute2Attribute	employee.id	worksIn.idEmployee	 affected
 3	Attribute2Attribute	project.id	worksIn.idProject	
 4	RelType2View	worksIn	vEmployee	 added

Fig. 5. Some elementary translations involved in the `changeCardinalityOfRole` example

Table 3. Sketch of the subalgorithm for the logical level

```

INPUT: Set of changes in the translation component and information
about the conceptual changes
OUTPUT: Changes on the logical component
For each translation change c of the INPUT
  Find the logical propagation rule r of the
  view based logical propagation rules set (if there is one) such that:
    - the event of r matches the translation change c AND
    - the condition of r is met
  Execute the action of r
end For
If such a rule not exists, repeat the same with the rules of the
original set of propagation rules
End For

```

based propagation algorithm. This is split into subalgorithms for the translation, for the logical and for the extensional component.

Subalgorithm for the translation component. This receives as input a set of changes in the conceptual component. Its output is a set of added, marked as deleted and affected elementary translations.

The initial subalgorithm for the translation component has been modified in order to give place to ‘marked as deleted’ elementary translations instead of deleted elementary translations.

For instance, when the `changeCardinalityOfRole` transformation of Figure 4 is executed, the elementary translation 1(Figure 5) is marked as deleted, the elementary translations 2 and 3 are affected, and the elementary translation 4 is added.

Subalgorithm for the logical component. The changes in the translation component are the input for the propagation subalgorithm for the logical component (see the sketch in Table 3).

It is based on the set of existing logical propagation rules together with the new view based logical propagation rules. Both kinds of rules are ECA rules [4] and the latter have the following meaning for its components:

Table 4. An example of logical propagation rule

1. **Name:** modifyView
 2. **Event:** newElementaryTranslation(e)
 3. **Condition:** (the conceptual change is `changeCardinalityOfRole(r,(0,1))`)
AND (the type of e is `RelType2View`) AND (the conceptual element of e is the
relationship type from which r is a role)
 4. **Action:**

```

et←getEntityType(r)
rt←getRelType(r)
source←getRelSchemaOrView(et)
target←getRelSchemaOrView(rt)
targetAttributes←getAttribs(rt,et)
sourceJoinAttributes←getEntityTypeID(et)
targetJoinAttributes←getRelTypeID(rt,et)
predicate←equality(sourceJoinAttributes,targetJoinAttributes)
includeRelationSchemaInView(source, target)
includeAttributeInView(source, targetAttributes)
changePredicate(source, predicate)

```
1. The event that triggers the rule. This is a transformation on the translation component.
 2. The condition that determines whether the rule action should be executed. This is expressed in terms of the conceptual and logical elements involved in the affected elementary translations and in terms of the conceptual change.
 3. The actions. These include procedures which create, delete or modify logical views.

An example of a view based logical propagation rule can be seen in Table 4.

The subalgorithm for the logical component has two sets of propagation rules. One of them is the set of logical propagation rules coming from the initial propagation algorithm. The second set is the set of newly created propagation rules with the purpose of delaying as far as possible the strict conversion of data.

For each elementary translation (whether it is added, marked as deleted or affected) the subalgorithm searches the only rule (if there is one) in the second set whose event is the same as the event of the elementary translation and whose condition evaluates to true. Then, the subalgorithm executes the actions of the rule changing the logical component. If there is no rule in the second set satisfying the above conditions, then a rule in the first set is searched for. If found and its condition evaluates to true, the actions of the rule are applied. If there is no rule in any of the sets, no rule is fired and the logical component is not changed for this elementary translation.

Our algorithm at work. When the conceptual transformation t_4 `changeCardinalityOfRole('rEmployeeWorksIn', (0,1))` is applied to the schema S_3 of Figure 4, the subalgorithm for the translation component, among other things, (1) adds a new elementary translation (number 4 in Figure 5) with type

Table 5. Logical changes after applying the conceptual transformations

- (1) **Logical changes for** deleteAttribute(employee.address)
 (a) createView('vEmployee', employee, null, (id, name, department), null, null);
- (2) **Logical changes for** turnAttributeIntoEntityType(employee.department, department, true, 'isHiredBy')
 (b) createView('vDepartment', null, vEmployee, department, 'department IS NOT NULL', null);
- (3) **Logical changes for** createEntityType('building', true), createAttribute('building.id', INTEGER), addAttributeToEntityType(building.id, building) **and** createKeyOfStrongEntityType(building, (id))
 (c) createRelationSchema('building', 'id', INTEGER, 'id')
- (4) **Logical changes for** createRelationshipTypeAndRoles('situatedIn', (department, building), null, ((0,N),(0,N)), true)
 (d) createRelationSchema('situatedIn', ('department', 'idBuilding'), (VARCHAR(30), INTEGER), ('department', 'idBuilding'))
 (e) createConstraint('fk3', 'fk', situatedIn.idBuilding, building.id)
- (5) **Logical changes for** changeCardinalityOfRole(rEmployeeWorksIn, (0,1))
 (f) includeRelationSchemaInView(vEmployee, worksIn)
 (g) includeAttributeInView(vEmployee, worksIn.idProject)
 (h) changePredicate(vEmployee, 'employee.id=worksIn.idEmployee')

Table 6. Generated SQL sentences which apply a view based evolution

- a. CREATE OR REPLACE VIEW vEmployee AS SELECT id, name, department FROM employee
- b. CREATE OR REPLACE VIEW vDepartment AS SELECT DISTINCT department FROM vEmployee WHERE department IS NOT NULL
- c. CREATE TABLE building(id INTEGER, name VARCHAR2(30), CONSTRAINT pk4 PRIMARY KEY (id))
- d. CREATE TABLE situatedIn(department VARCHAR2(20), idBuilding INTEGER, CONSTRAINT pk5 PRIMARY KEY(department, idBuilding))
- e. ALTER TABLE situatedIn ADD CONSTRAINT fk3 FOREIGN KEY (idBuilding) REFERENCES building(id)
- f. ALTER TABLE worksIn ADD CONSTRAINT uniq1 UNIQUE(idEmployee)
- g. CREATE OR REPLACE VIEW vEmployee AS SELECT id, name, department, idProject FROM employee, worksIn WHERE employee.id=worksIn.idEmployee

RelType2View to store the fact that the relationship type `worksIn` is now being converted into a view, (2) marks as deleted the elementary translation 1 in Figure 5 and (3) detects the affected elementary translations numbers 2 and 3. These elementary translations are the input for the subalgorithm for the logical component. For each one of them, the subalgorithm searches the logical rule (if any) in the second set with the same event as the event of the translation change and for which the condition of the rule evaluates to true.

For the newly added elementary translation number 4, the logical propagation rule `modifyView` is fired because its event is the same as the event of the added elementary translation and its condition evaluates to true. As a consequence, the view `vEmployee` is modified as can be seen in (f) to (h) of Table 5. With respect to the affected and marked as deleted elementary translations no changes are made at the logical level because no rule is fired.

Finally, the extensional propagation subalgorithm (not shown here) takes as input the changes in the logical component and changes the extensional component by means of SQL sentences. For the complete running example, the SQL sentences generated by this subalgorithm can be seen in Table 6. Sentence (a) corresponds to the conceptual transformation `deleteAttribute`, sentence (b) to transformation `turnAttributeIntoEntityType`, sentence (c) to transformations `createEntityType`, `createAttribute`, `addAttributeToEntityType` `createKeyOfStrongEntityType`, sentences (d) and (e) to transformation `createRelationshipTypeAndRoles` and sentences (f) and (g) to transformation `change CardinalityOfRole`.

5 Related Work

The development of techniques that enable schema changes to be accommodated as seamlessly and as painlessly as possible is an apparent theme in recent schema evolution research [9]. This is the main aim that guides the proposal we present in this paper. We started from a schema evolution proposal we presented in [2], which follows a strict data conversion mechanism. The problem of the strict (or eager, or early) conversion method is that it takes longer schema modification time [8] causing a heavy load to the system [12]. With the goal of avoiding this problem, in this paper we have modified our previous proposal by using a lazy and logical data conversion mechanism. The lazy (or deferred) and logical mechanism has the advantage that changes can be made more rapidly [8].

We propose to achieve the lazy and logical conversion mechanism by means of views, so that a logical modification of data is performed whenever the conceptual changes allow this to be done. Views have been proposed by several authors [10] as a way of performing different schema evolution processes. The most common use of views has been to simulate schema versioning [13]. View mechanism has also been employed to create personal schemas when a database is shared by many users [11]. However, these approaches lack the consideration of a conceptual level which allows the designer to work at a higher level of abstraction.

The novelty of our approach is that the schema evolution is performed at a conceptual level so that users interact with schema changes at a high level, such as is demanded in [9]. To our knowledge the approaches that propose a conceptual level (for example [5]) do not propose a lazy and logical conversion mechanism. In [6] views are used within a framework with conceptual level, but they are involved during the translation process instead of during the evolution process as in our case. For this reason, as far as we are aware, this is the first time that views are used for achieving a lazy and logical conversion mechanism within a framework where different levels of abstraction are involved following a metamodel perspective [3].

6 Conclusions and Future Work

One of the main problems that arise when database evolution tasks must be realized is that of propagation of modifications. In particular, the propagation in a

vertical sense (from conceptual to logical, from logical to extensional levels) can be carried out following different data conversion mechanisms. The strict mechanism, in which changes are propagated immediately, is the most frequently used. For instance, this is the method we used in [2] where we have presented MeDEA, a database evolution architecture with traceability properties. However, it has been recognized that the strict mechanism takes longer schema modification time and causes a heavy workload to the system. As a step to solve this problem, in the present paper we have proposed a new application of MeDEA, by making use of the notion of view. The main advantage of using views is that it is not necessary to use the strict mechanism. On the contrary, the changes in data do not have to be initially realized, which sets a scenario closer to the logical data conversion mechanism. Furthermore, views codify internally the changes, in such a way that these changes can be realized explicitly, if required, in a similar way to the lazy mechanism.

In the present paper we have used the MeDEA architecture by fixing EER and Relational as techniques for the conceptual and logical levels, respectively. This decision has allowed us to use views in a natural way. A possible future line of work is the study of other kind of techniques where views or similar concepts can be used to carry out evolution tasks.

References

1. Bernstein, P.A.: Applying Model Management to Classical Meta Data Problems. In: First Biennial Conference on Innovative Data Systems Research- CIDR 2003, Online Proceedings (2003)
2. Domínguez, E., Lloret, J., Rubio, A.L., Zapata, M.A.: MeDEA: A database evolution architecture with traceability. *Data & Knowledge Engineering* 65, 419–441 (2008)
3. Domínguez, E., Zapata, M.A.: Noesis: Towards a Situational Method Engineering Technique. *Information Systems* 32(2), 181–222 (2007)
4. Elmasri, R.A., Navathe, S.B.: *Fundamentals of Database Systems*, 4th edn. Addison-Wesley, Reading (2003)
5. Hick, J.M., Hainaut, J.L.: Database application evolution: A transformational approach. *Data and Knowledge Engineering* 59(3), 534–558 (2006)
6. Mork, P., Bernstein, P.A., Melnik, S.: Teaching a Schema Translator to Produce O/R Views. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) *ER 2007*. LNCS, vol. 4801, pp. 102–119. Springer, Heidelberg (2007)
7. Peters, R.J., Tamer Özsu, M.: An Axiomatic Model of Dynamic Schema Evolution in Objectbase Systems. *ACM Trans. on Database Systems* 22(1), 75–114 (1997)
8. Roddick, J.F.: A survey of schema versioning issues for database systems. *Information Software Technology* 37(7), 383–393 (1995)
9. Roddick, J.F., de Vries, D.: Reduce, Reuse, Recycle: Practical Approaches to Schema Integration, Evolution and Versioning. In: Roddick, J.F., Benjamins, V.R., Si-said Cherfi, S., Chiang, R., Claramunt, C., Elmasri, R.A., Grandi, F., Han, H., Hepp, M., Lytras, M., Mišić, V.B., Poels, G., Song, I.-Y., Trujillo, J., Vangenot, C. (eds.) *ER Workshops 2006*. LNCS, vol. 4231, pp. 209–216. Springer, Heidelberg (2006)

10. Ram, S., Shankaranarayanan, G.: Research issues in database schema evolution: the road not taken, working paper # 2003-15 (2003)
11. Ra, Y.G., Rundensteiner, E.A.: A Transparent Schema-Evolution system based on object-oriented view technology. *IEEE Transactions of Knowledge and Data Engineering* 9(4), 600–624 (1997)
12. Tan, L., Katayama, T.: Meta Operations for Type Management in Object-Oriented Databases – A Lazy Mechanism for Schema Evolution. In: Kim, W., et al. (eds.) *Proceedings of the First International Conference on Deductive and Object-Oriented Databases*, pp. 241–258. North Holland, Amsterdam (1990)
13. Tresch, M., Scholl, M.H.: Schema Transformation without Database Reorganization. *ACM SIGMOD Record* 22(1), 21–27 (1993)
14. OMG, UML 2.1.2 Superstructure Spec., formal/ 2007-11-02 (2007), www.omg.org

Inventing Less, Reusing More, and Adding Intelligence to Business Process Modeling

Lucinéia H. Thom¹, Manfred Reichert¹, Carolina M. Chiao²,
Cirano Iochpe², and Guillermo N. Hess²

¹ Institute of Databases and Information Systems,
Ulm D-89069 – Ulm, Germany

{lucineia.thom,manfred.reichert}@uni-ulm.de

² Institute of Informatics, Federal University of Rio Grande do Sul,
Av. Bento Gonçalves, 9500, 91501-970 – Porto Alegre, RS, Brazil
{cchiao,ciochpe,hess}@inf.ufrgs.br

Abstract. Recently, a variety of workflow patterns has been proposed focusing on specific aspects like control flow, data flow, and resource assignments. Though these patterns are relevant for implementing Business Process Modeling (BPM) tools and for evaluating the expressiveness of BPM languages, they do not contribute to reduce redundant specifications of recurrent business functions when modeling business processes. Furthermore, contemporary BPM tools do not support process designers in defining, querying, and reusing activity patterns as building blocks for process modeling. Related to these problems this paper proposes a set of activity patterns, evidences their practical relevance, and introduces a BPM tool for the modeling of business processes based on the reuse of these activity patterns. Altogether our approach fosters reuse of business function specifications and helps to improve the quality and comparability of business process models.

Keywords: business function, activity pattern, business process modeling.

1 Introduction

Business processes help organizations to better align their business goals with the needs of their customers; i.e., business processes constitute the glue between the strategic and the operational level of the organization [1]. To stay competitive in their market, organizations and companies are increasingly interested in improving the quality and the efficiency of their business processes as well as their interactions with customers and business partners [2].

Process-aware information systems (PAISs) offer promising perspectives to realize these goals, and a growing interest in aligning information systems (IS) in a process-oriented way can be observed. To allow for more flexibility, PAISs introduce an additional layer when compared to traditional information systems, which provides an explicit description of the process logic. In general, this logic is represented as a process model which can be created using a business process modeling (BPM) tool.

The introduction of PAISs and the adoption of BPM tools offer promising perspectives: (a) companies obtain a precise and unambiguous description of their business processes; (b) the definition of new business processes and new process models respectively can be speed up significantly; (c) the work between different actors can be coordinated more effectively; (d) real-time data about in-progress processes can be gathered and visualized; and (e) business processes can be standardized. Through Web Service technology, in addition, the benefits of BPM can be applied to cross-organizational business processes as well [3], [4].

Business processes comprise different *business functions* with specific and well-defined semantics, which can be considered as self-contained building blocks. Generally, a particular business function may occur several times within one or multiple business process models [3]. As example consider the process for approving the contents of a newsletter (cf. Fig 1). This simple business process includes three activities with following order: (a) The *author* sends a request for approving the article to be published to the *editor* of the current newsletter edition. (b) The *editor* reviews the contents of the article; she either approves it or requests changes from the *author*. (c) If the newsletter article is not sent to the editor after a certain period of time, the author will receive a respective notification. Obviously, this process comprises business functions with generic semantics, which recur in numerous business processes: Task Execution Request (a), Approval (b), and Notification (c). As we will discuss later such recurrent business functions can be described in terms of activity patterns in order to foster their reuse and to improve the quality and comparability of business process models.

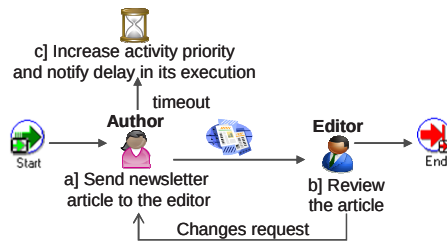


Fig. 1. Approval process for newsletter content

Recently a variety of *workflow patterns* has been proposed focusing on different process aspects like control flow [5], resource assignments [6], data flow [7], exception handling [8], domain-specific ontologies [9], service interactions [10], process change [11,30], and process compliance [12]. Though all these patterns are useful for implementing BPM tools and for evaluating the expressiveness of BPM languages, they do not contribute to avoid redundant specifications of recurrent business functions when modeling business processes. Consequently, business process design becomes inefficient, complex and time-consuming when being confronted with a large collection of business functions and business process models. To our best knowledge there exists no comprehensive work evidencing the existence of *activity patterns* for defining such recurrent business functions within business process models. Furthermore, no efforts have been spent on investigating the need, benefits and completeness

of activity patterns with respect to business process modeling. Finally, contemporary BPM tools like Intalio, ARIS Toolset and WBI Modeler do not support process designers in defining, querying and reusing activity patterns as building blocks for business process modeling.

Related to these problems we proposed a set of seven *workflow activity patterns* and corresponding design choices (i.e., pattern variants) in previous work [3], [13]. Each of these activity patterns captures a recurrent business function as we can find it in numerous business processes like the one shown in Fig. 1. Combined with existing control flow patterns (e.g., sequence, multi instance activity), these activity patterns are suited to design a large variety of process models in different domains.

In this paper we briefly report on the results of an empirical study in which we analyze the relative frequency of activity patterns in a collection of 214 real-world process models from domains like quality management, software access control, and electronic change management. For selected process categories, we further discuss results of an additional analysis in which we investigate the frequency of co-occurring activity patterns. The results of this second analysis are also utilized for developing a BPM tool, which shall foster the modeling of business processes based on the reuse of activity patterns. Given some additional information about the kind of process to be designed, the results of our analysis can be further used by this tool to suggest a ranking of the activity patterns suited best to succeed the last pattern applied during process modeling.

One of the basic pillars of this BPM tool constitutes an ontology to describe the activity patterns. Particularly, this ontology allows to store and retrieve the patterns (together with their properties and constraints) during process modeling. To obtain machine-readable specifications, we suggest using a standard ontology language (e.g., OWL). Based on this, any BPM model based on the activity patterns can be easily and automatically transformed to conventional process modeling notations (e.g., BPMN) or languages (e.g., WS-BPEL, XPD). Finally, using an ontology allows to make the relationships between the different activity patterns more explicit, which provides useful information for process designers.

The remainder of this paper is organized as follows: Section 2 gives an overview of workflow activity patterns. Exemplarily, we discuss basic principles taking the *Approval Pattern*. Section 3 presents the results of our empirical study in which we investigate the occurrence and relevance of activity patterns in practice by analyzing 214 real-world process models. Section 4 sketches our process modeling tool and discusses how the user interacts with it. In this context we also describe the ontology representing the patterns in detail. Section 5 discusses related work and Section 6 concludes with a summary and outlook.

2 Workflow Activity Patterns

In this paper we use the term *Workflow Activity Pattern* (WAP or *activity pattern* for short) to refer to the description of a recurrent business function as it frequently occurs in business process models. Examples include *notification*, *decision*, and *approval*. Initially, we derived seven activity patterns based on an extensive literature study [3], [13]. Table 1 gives an overview of these patterns. Generally, these activity

patterns are close to the abstraction level or vocabulary used within an organization. This, in turn, fosters their reuse when modeling business processes, and therefore contributes to more standardized and better comparable business process models.

Each activity pattern is characterized by a short *description*, an *example*, a description of the *problem* context in which it can be applied, and relevant *issues*. Our framework considers additional attributes as well like *design choices* (pattern variants), *related patterns* and *pattern implementation*. However, these attributes are outside the scope of this paper and are omitted here. Fig. 2 gives a simple example of the description for the APPROVAL activity pattern (here restricted to a particular pattern variant, namely *single approval*; i.e., approval is required from exactly one role).

Table 1. Selected variants of activity patterns representing business functions

WAP - Name	Description
WAP I: <i>Approval</i>	An object (e.g., a business document) has to be approved by one or more organizational roles.
WAP II: <i>Question-Response</i>	A question which emerges during process enactment has to be answered. This pattern allows to formulate the question, to identify the organizational role(s) who shall answer it, to send the question to the respective role(s), and to wait for the response(s) (<i>single-question-response</i>).
WAP III: <i>Unidirectional Performative</i>	A sender requests the execution of a particular task from another process participant. The sender continues process execution immediately after having sent the request for performing the activity.
WAP IV: <i>Bi-directional Performative</i>	A sender requests the execution of a particular task from another process actor. The sender waits until this actor notifies him that the requested task has been performed.
WAP V: <i>Notification</i>	The status or result of an activity execution is communicated to one or more process participants.
WAP VI: <i>Informative</i>	An actor requests certain information from a process participant. He continues process execution after having received the requested information.
WAP VII: <i>Decision</i>	This pattern can be used to represent a decision activity in the flow with different connectors to subsequent execution branches. Those branches will be selected for execution whose transition conditions evaluate to true.

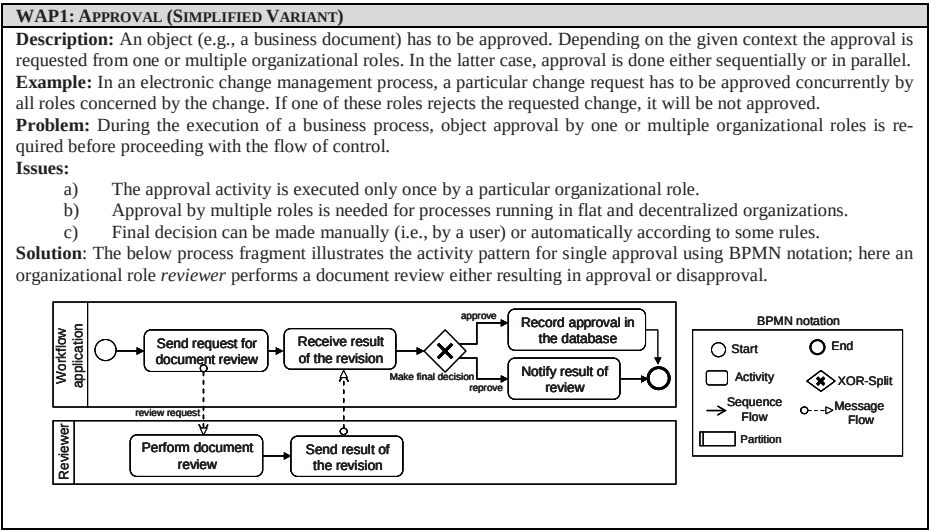


Fig. 2. Approval pattern

3 Evidencing the Existence and Relevance of Activity Patterns in Practice

To identify activity patterns in real workflow applications we analyzed 214 process models. These process models have been modeled either with the Oracle Builder tool or an UML modeler. Our analysis has been based on process models instead of event logs, since we consider the semantics and the context of process activities as being fundamental for identifying activity patterns. Altogether, the analyzed process models stem from 13 different organizations – private as well as governmental ones – and are related to different applications like Total Quality Management (TQM), software access control, document management, help desk services, user feedback, document approval, and electronic change management. In all organizations the respective process models have been operationalized, i.e. they were supported by a PAIS. Table 2 summarizes information about involved organizations and analyzed process models.

Table 2. Characteristics of the analyzed process models

Size and Number of Companies	Kind of Decision-making	Examples of Analyzed Process Models	Number of Analyzed Models
1 x small	Decentralized	Management of Internal Activities	17
1 x large	Decentralized	TQM; Management of Activities	11
6 x large	Centralized	TQM; Control of Software Access; Document Management	133
4 x large	No information available	Help Desk Services, User Feedback; Document Approval	29
1 x large	Centralized	Electronic Change Management	24

3.1 Method Used to Analyze the Process Models

To our knowledge there exist no mining techniques to extract activity patterns from real-world process models; i.e., contemporary process mining tools like ProM analyze the event logs (e.g., execution or change logs) related to process execution and do not extract information related to the semantics and the (internal) logic of process activities [14], [5], [15]. Therefore, we perform a manual analysis in order to identify relevant activity patterns as well as their co-occurrences within the 214 process models.

For each workflow activity pattern WAP^* we calculate its support value S_{WAP^*} , which represents the relative frequency of the respective activity pattern within the set of analyzed process models; i.e., $S_{WAP^*} = \text{Freq}(WAP^*)/214$ where $\text{Freq}(WAP^*)$ denotes the absolute frequency of WAP^* within the collection of the analyzed 214 models; for each process model we count at most one occurrence of a particular pattern.

First, we manually identify and annotate activity patterns in all process models analyzed. Following this, we determine the absolute frequency of each activity pattern WAP^* as described above. The obtained results are divided by the total number of analyzed process models (i.e., 214 in our case).

3.2 Frequency of Activity Patterns in Real-World Process Models

Our analysis has shown that five out of the seven activity patterns (cf. Table 1) are not dependent on a specific application domain or on a particular organizational structure

(e.g., the degree of centralization in decision making or the standardization of work abilities). More precisely, this applies to the following five patterns (cf. Table 1): UNIDIRECTIONAL and BI-DIRECTIONAL PERFORMATIVE (WAP III+IV), DECISION (WAP VII), NOTIFICATION (WAP V), and INFORMATIVE (WAP VI). We could identify these five patterns with high frequency in almost all process models we had analyzed. The latter also applies to the APPROVAL pattern, which can be explained by the high degree of centralization regarding decision-making within the considered organizations (cf. Table 2). This high centralization implies the use of approval activities [16]. By contrast, most of the analyzed process models do not contain QUESTION-RESPONSE activities. Figure 3 graphically illustrates the relative frequency of each activity pattern with respect to the set of analyzed process models.

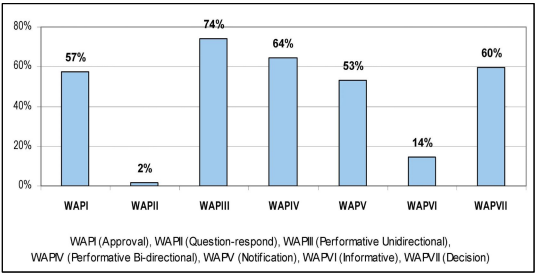


Fig. 3. Frequency of activity patterns in real process models

3.3 Identifying Co-occurrences of Activity Patterns

One of the use cases for the *ontology* of our BPM tool (cf. Section 4) is based on a mechanism that gives design time recommendations with respect to the activity patterns best suited to be combined with the last used pattern. This mechanism utilizes statistical data we gathered during our empirical study. We summarize these statistical findings in this section. To obtain the frequencies for pattern co-occurrences, we analyze the sequences of activity patterns in 154¹ out of 214 studied process models.

Before performing the analysis we classified the business process models into human-oriented models (i.e., processes with human interventions during their execution) and fully automated models (i.e., processes without any human intervention). We verified that certain activity patterns can be find more often in one of the two categories [17]. This analysis has been inspired by a classification provided by Le Chair who distinguishes between *system-intensive* and *human-intensive* business processes [18]. System-intensive processes are characterized by being handled on a straight-through basis, i.e., there is minimal or no human intervention during process execution and only few exceptions occur. Human-intensive processes (similar to methods engineering [19]), in turn, require people to get work done by relying on

¹ When performing this analysis we had access to only 154 out of the 214 studied process models.

business applications, databases, documents, and other people as well as extensive interactions with them. This type of process requires human intuition or judgment for decision-making during individual process steps.

When classifying a subset of the studied process models, for which respective information is available, into these two categories, we obtain 123 human-intensive and 31 system-intensive process models respectively. Note that in this earlier analysis we consider only 154 of the 214 process models studied in total. In a next step we evidence the occurrence of the seven activity patterns with respect to the two categories of process models. Figure 4 shows the support value (i.e., the relative frequency) of the activity patterns in both the *system-intensive* and the *human-intensive* process models. As can be seen, some of the patterns (i.e., APPROVAL (WAP I), INFORMATIVE (WAP VI), and QUESTION-RESPONSE (WAP II)) do not appear in system-intensive process models at all. Obviously, these patterns are usually related to human activities; i.e. they are executed by an organizational role.

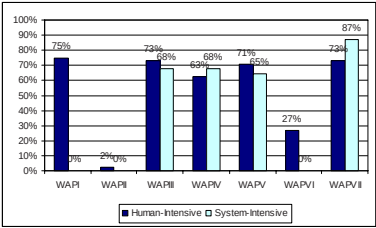


Fig. 4. Frequency of activity patterns in human- and system-intensive process models

In another analysis we have identified frequent and recurrent co-occurrences of activity patterns within process models. Relying on the results of this analysis, our knowledge base and BPM tool respectively display to the process designer a ranking of the activity patterns which most frequently follow the pattern the user applied before during process design. For example, our analysis has shown that the pattern pair DECISION → NOTIFICATION occurs more often in *system-intensive* than in *human-intensive* business processes (cf. Fig. 5). Opposed to this, pattern pair DECISION → APPROVAL occurs more frequently in *human-intensive* process models.

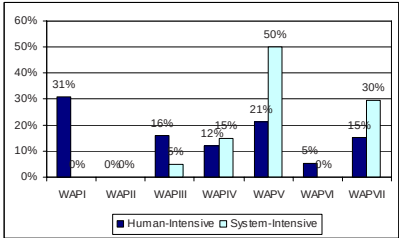


Fig. 5. How often does an activity pattern directly follow the DECISION pattern (regarding both system- and human-intensive processes)?

4 Towards a Pattern-Based Process Modeling Tool

This section presents basic concepts of our BPM tool, which allows to design process models based on the reuse of activity patterns. The latter are described by means of an ontology. In principle, basic concepts behind this BPM tool can be added as extension to existing BPM components as well (e.g., Intalio [20], Aris Toolset [21], or ADEPT Process Composer [22]).

Core functionalities of our BPM tool are as follows:

Assisting users in designing process models. First, the process designer selects the kind of business process (e.g., human intensive) to be modeled, which is then matched to a set of business functions as maintained in the ontology; i.e., the BPM tool adapts a set of business functions to be used for process modeling in the given context. Following this, the process designer chooses a business function and provides contextual data (e.g., about the organization). This information is then matched to an activity pattern as maintained in the aforementioned ontology. Furthermore, the BPM tool recommends to use the respective activity pattern and to apply corresponding design choices; i.e., to configure a concrete *pattern variant*. Afterwards, the tool recommends the most suitable activity patterns to be used together with the activity pattern applied before. In addition, it informs the user about how frequently each pair of activity pattern (i.e., the previously applied activity pattern plus the recommended activity pattern) was used in earlier modeling. This module is developed based on the analysis results presented in Section 3.3.

Construction of an ontology for activity patterns. The ontology for activity patterns does not only maintain the patterns themselves, but also the frequency with which each pattern has co-occurred with a previously used pattern. Through the analyses of additional process models (e.g., from the automotive as well as the health-care domain) we aim at increasing the support value of such pairs of activity patterns (cf. Section 3.3). Thus, at design time the pattern pairs being recommended will help to design a process model which is closer to the business process being manually executed in the organization.

4.1 Architecture of the Process Modeling Tool

Core components of our BPM Tool are as follows (cf. Fig. 6):

- **Query Component:** It provides a query mechanism for *matching* the activity patterns maintained by the ontology with the given kind of business process (e.g., human intensive), business function (e.g., approval), organizational context (e.g., level of centralization in decision-making), and corresponding design choice as chosen by the user (if not set automatically).
- **Ontology Manager:** It comprises an *ontology* and a *query mechanism* (Business Function Query + WAP Query). The ontology describes the activity patterns (cf. Fig. 6) and their properties (e.g., attributes and relationships with other patterns). Our query as well as update mechanisms give design time recommendations with respect to the most suited activity patterns to be combined with an already used one. An example of a query would be the selection of the

business functions which occur more frequently in system-intensive process models. In addition, our update mechanism has to be used to adapt relative frequency of each pattern pair (e.g., based on the analysis of new process models) as identified in our process model analysis.

- Scheme Translation:** This component is responsible for translating a process model (based on translation algorithms) which uses activity patterns as building blocks (stored in XML code) to either a conventional notation (e.g., BPMN) or an existing process execution language (e.g., BPEL, XPD). The use of this translation component is optional, i.e., it will be only applicable if the user wants the respective process model being translated to another notation and process execution language respectively.

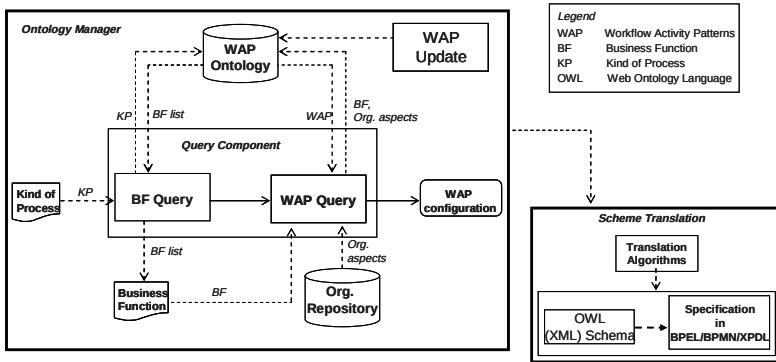


Fig. 6. Architecture of the ProWAP process modeling tool

4.2 Interacting with the Process Modeling Tool

We sketch basic steps of our pattern-based modeling approach: First, the user specifies the kind of process to be designed (e.g., system- vs. human-intensive) (cf. Step 1 in Fig. 7). Taking this information, in Step 2 the BPM tool presents a set of business functions relevant in the given context (cf. Section 3.2). In Step 3, the user selects a business function (e.g., approval) and additionally specifies information about the organizational context within which this function is executed (e.g., organizational unit with centralized decision-making).

Based on this information and on the defined ontology, the best suited activity pattern (incl. corresponding design choices) is queried (cf. Step 4 in Fig. 7). More precisely, the query results in an instance of an activity pattern and a set of related attributes (e.g., kind of approval, list of reviewers in case of concurrent or iterative approvals, application-specific details about the approval like approval conditions, etc.). In Step 5, the user then customizes the activity pattern to the given context (e.g., *kind of approval* (iterative approval by a hierarchy of organizational roles); *list of reviewers* (#id): 101, 106, 200; *Approval activity description*: paper review; *conditions to approve*: at least 2 strong acceptances). In Step 6, the BPM tool creates an instance of the corresponding pattern according to this customization. Afterwards, the BPM tool recommends a list of activity patterns (with corresponding statistics regarding the use

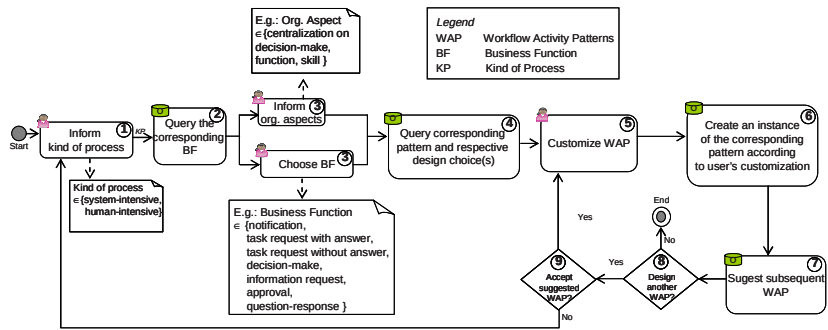


Fig. 7. Procedure showing how the user interacts with the BPM tool

of these patterns in other process models) to follow the previously modeled activity (cf. Step 7 in Fig. 7). For example, APPROVAL is followed by NOTIFICATION in 24% of the 154 process models we analyzed. The user must then state whether another activity pattern shall be designed and whether the pattern suggested by the BPM tool shall be used (cf. Steps 8 and 9).

4.3 Ontology of Workflow Activity Patterns

We now introduce our ontology for workflow activity patterns (cf. Fig.8). Particularly, this ontology has been used when implementing the recommendation mechanism of our BPM Tool. The ontology represents the structure of the activity patterns, related design choices (e.g., single and iterative approval), and their relationships. Altogether, the main goal of this ontology is to better define the structure of activity patterns, their attributes, and their relationships. In addition, the ontology maintains the use statistics for each activity pattern (in the context of process modeling) as well as the co-occurrences of pattern pairs based on the empirical study reported in Section 3.3. Our ontology also integrates information about the organizational context in which the activity patterns are usually applied. Such information is matched with the business function to identify the most suitable activity patterns to be used together with the activity pattern designed before.

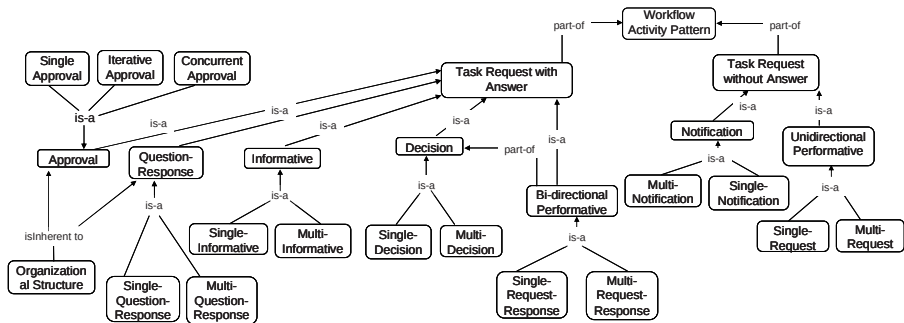


Fig. 8. Ontology of Workflow Activity Patterns (simplified and partial views)

The diagram depicted in Fig. 8 represents the Workflow Activity Patterns ontology (partial view). It describes the taxonomy of the activity patterns, i.e. a conceptual hierarchical structure that relates classes and sub-classes to each other. The most general class is WORKFLOW ACTIVITY PATTERN. This concept has two main components: TASK REQUEST WITH ANSWER and TASK REQUEST WITHOUT ANSWER. The first component is composed by the following concepts, which represent the Workflow Activity Patterns: APPROVAL, QUESTION-RESPONSE, INFORMATIVE, DECISION and BIDIRECTIONAL PERFORMATIVE. The TASK REQUEST WITHOUT ANSWER concept is composed by the two patterns NOTIFICATION and UNIDIRECTIONAL PERFORMATIVE.

The APPROVAL and QUESTION-RESPOND patterns are related to organizational structure aspects. In the context of an APPROVAL pattern, an object approval is always performed by specific roles inside an organization. The QUESTION-RESPONSE pattern is applied when an actor might have a question in order to work on the process or on a particular process activity. The question is forwarded to a specific organizational role that has the appropriate expertise to answer it.

As aforementioned each workflow activity pattern has specific design choices. For example, the design choices of task execution pattern allow expressing at design time if the task execution is requested from a single actor or from multiple actors. In Fig. 8 such variants are represented as subclasses of the pattern concepts, i.e. as specializations. The *Approval* concept, for example, has three specializations: *Single Approval*, *Iterative Approval* and *Concurrent Approval*. They represent the Approval pattern design choices. *Single Approval* is associated with exactly one reviewer. *Iterative Approval* is handled by a loop based on a list of reviewers. This list can be often related to vertical organizations where there is a hierarchical organizational structure. Regarding *Concurrent Approval*, the approval request is sent to the whole group of reviewers simultaneously. The final decision is then made after all reviewers have performed their evaluation.

Our ontology (complete view) also considers *system-* and *human-intensive* processes. In addition, we have defined attributes for each class and sub-class of our ontology; e.g., way of notification (e.g., by e-mail or as work item in the participants' worklists), organizational role, and activity status.

5 Related Work

Recently, numerous approaches on workflow patterns have been proposed to support both the conceptual and the implementation level of business processes. One of the first contributions in this respect was a set of process patterns to be used in the context of software processes within an organization [23].

Russell proposes 43 workflow patterns for describing process behavior [24], [5]. Each pattern represents a routing element (e.g., sequential, parallel and conditional routing) which can be used in process definitions. In the meantime these workflow patterns have been additionally used for evaluating process specification languages and process modeling tools [25], [26].

A set of data patterns is proposed by [7]. These patterns are based on data characteristics that occur repeatedly in different workflow modeling paradigms. Examples are *data visibility* and *data interaction*. In another work, Russell presents a set of

resource patterns of which each one describes a way through which resources can be represented and utilized in process models [6]. A resource is considered as an entity capable of doing work. It can be either human (e.g., a worker) or non-human (e.g., equipment). Examples of resource patterns include *Direct Allocation* and *Role-Based Allocation*. Finally, process change patterns and change support features have emerged to effectively deal with (dynamic) process changes [11], [30].

Recently, Russell has presented a pattern-based classification framework for characterizing *exception handling* in workflow management systems [8]. This framework has been used to examine the capabilities of workflow management and BPM systems and to evaluate process specification as well as process execution languages. As a result, the authors emphasize the limited support for exception handling in existing workflow management systems.

Barros proposes a set of *service interaction patterns*, which allow for service interactions, pertaining to choreography and orchestration, to be benchmarked against abstracted forms of representative scenarios [10]. As example consider the *Send* pattern and the *Send/Receive* pattern. Altogether Russell and Barros provide a thorough examination of the various perspectives that need to be supported by a workflow language and BPM tool respectively. However, none of these approaches investigate which are the most frequent patterns recurrently used during process modeling and in which way the introduction of such activity patterns eases process modeling.

SAP has developed a cross-application engine called SAP Business Workflow [27]. This tool enables the process-oriented integration of business objects and applications including a workflow wizard with workflow templates and process reference models. Related to that is the Supply-Chain Operations Reference-model (SCOR) which is based on best practices as well [28].

Finally, PICTURE proposes a set of 37 domain-specific process building blocks. More precisely, these building blocks are used by end users in public administrations to capture and model the process landscape. The building blocks have been evaluated in practice [29].

6 Summary and Outlook

In this paper, we identified seven workflow activity patterns necessary and in many cases also sufficient to model a large variety of process models. Moreover we discussed results of an empirical study where we had analyzed how often activity patterns as well as co-occurrences of them (i.e. pairs of activity patterns) are present in a large collection of real process models.

We additionally proposed an approach for process modeling based on workflow activity patterns. Basic to this tool is the reuse of the presented activity patterns. Respective functionality can be added as extension to existing process modeling components as well. Our goal is to increase the reuse of recurrent business functions and to better assist users in the design of high-quality process models; i.e., by providing context-specific recommendations on which patterns to use for process modeling next. It is important to mention that our patterns can be used together with other patterns related to control flow [5] or process change [11], [30]. In this paper we have given first insights into the architecture of our BPM tool as it is implemented in the ProWAP project.

The main advantages of our approach can be summarized as follows: (a) the completeness and necessity of the activity patterns for process design have been empirically evidenced; (b) a small set of parameterizable activity patterns is sufficient to model a large variety of processes, which reduces the complexity with respect to pattern usage; (c) the concepts realized in our process modeler are tool-independent and can be adapted for any process modeling tool; and (d) the sketched recommendation mechanisms can be useful to reduce complexity in process design as well as to improve semantical model correctness.

In future work, we intend to identify variants of each pattern concerning specific application domains. For example, we want to figure out what kind of approvals occur most frequently in the healthcare and in the automotive domain. Furthermore, we will perform additional analyses considering process models from different application domains (e.g., health insurance and automotive engineering). Our goal is to identify more common occurrences of pairs of activity patterns. We also aim at extending the tool with a mechanism to learn from designer actions. Last but not least, we intend to investigate how to transform process models defined with our tool (and being based on the activity patterns) into conventional notations and languages respectively.

Acknowledgements

The authors would like to acknowledge the Coordination for the Improvement of Graduated students (CAPES), the Institute of Databases and Information Systems of the University of Ulm (Germany), and the Informatics Institute of Federal University of Rio Grande do Sul (Brazil).

References

1. Rummler, G., Brache, A.: *Improving Performance: How to Manage the White Space on Organizational Chart*. Jossey-Bass, San Francisco (1990)
2. Lenz, R., Reichert, M.: IT Support for Healthcare Processes – Premises, Challenges, Perspectives. *Data and Knowledge Engineering* 61, 39–58 (2007)
3. Thom, L., Iochpe, C., Amaral, V., Viero, D.: Toward Block Activity Patterns for Reuse in Workflow Design. In: *Workflow Handbook 2006*, pp. 249–260 (2006)
4. Mutschler, B., Reichert, M., Bumiller, J.: Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications. *IEEE Transactions on Systems, Man, and Cybernetics (Part C)* 38(3), 280–291 (2008)
5. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.: Workflow Patterns. *Distributed and Parallel Databases* 14(3), 5–51 (2003)
6. Russel, N., Aalst, W., Hofstede, A., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) *CAiSE 2005*. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
7. Russel, N., Hofstede, A., Edmond, D.: Workflow Data Patterns. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) *ER 2005*. LNCS, vol. 3716, pp. 353–368. Springer, Heidelberg (2005)
8. Russel, N., Aalst, W., Hofstede, A.: Workflow Exception Patterns. In: Dubois, E., Pohl, K. (eds.) *CAiSE 2006*. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
9. Ohnmacht, A.: *Development of a Collaborative Process Modeling Methodology for Domain Experts*. Master Thesis, University of Ulm (2007)

10. Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) *BPM 2005*. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
11. Weber, B., Rinderle, S., Reichert, M.: Change Patterns and Change Support Features in Process-Aware Information Systems. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) *CAiSE 2007*. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
12. Namiri, K., Stoganovic, N.: Pattern-Based Design and Validation of Business Process Compliance. In: *Proc. CoopIS 2007*, pp. 59–76 (2007)
13. Thom, L.: Applying Block Activity Patterns in Workflow Modeling. In: *Proc. 8th Int'l Conf. on Enterprise Information Systems (ICEIS 2006)*, Paphos, Cyprus, pp. 457–460 (2006)
14. Ellis, C.: Workflow Mining: Definitions, Techniques, Future Directions. In: Fischer, L. (ed.) *Workflow Handbook 2006*. Lighthouse Point: Future Strategies, pp. 213–228 (2006)
15. Günther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P., Recker, J.: Using Process Mining to Learn from Process Changes in Evolutionary Systems. *Int'l Journal of Business Process Integration and Management* (2008)
16. Davis, M.R., Weckler, D.A.: *A Practical Guide to Organization Design*. Crisp Publications, Boston (1996)
17. Chiao, C., Thom, L.H., Iochpe, C., Reichert, M.: Verifying Existence, Completeness and Sequences of Workflow Activity Patterns in Real Process Models. In: *IV Brazilian Symposium of Information Systems (SBSI)*, Rio de Janeiro, Brazil (2008)
18. Le Clair, C., Teubner, C.: *The Forrester Wave: Business Process Management for Document Processes, Q3* (2007)
19. Wiley, J.: *Methods Engineering*, United States of America (1962)
20. Intalio. *Creating Process Flows* (2006), <http://bpms.intalio.com>
21. IDS Scheer: *Aris Platform: Product Brochure* (2007), http://www.ids-scheer.com/set/82/PR_09-07_en.pdf
22. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive Process Management with ADEPT2. In: *Proc. Int'l Conf. on Data Engineering (ICDE 2005)*, Tokyo, Japan, pp. 1113–1114. IEEE Computer Society Press, Los Alamitos (2005)
23. Ambler, S.W.: *An Introduction to Process Patterns* (1998)
24. Russell, N., Hofstede, A.H.M., ter Aalst, W.M.P., van der Mulyar, N.: *Workflow Control Flow Patterns: A Revised View*. BPM Center Report BPM-06-22, BPM center.org (2006)
25. van der Aalst, W.M.P.: *Patterns and XPD: A Critical Evaluation of the XML Process Definition Language*. QUT Technical report, FIT-TR-2003-06, Queensland University of Technology, Brisbane (2003)
26. Wohed, P., Aalst, W.M.P., van der Dumas, M., ter Hofstede, A.H.M., Russell, N.: *Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives*. BPM Center Report BPM-06-17, BPMcenter.org (2006)
27. SAP. *SAP Business Workflow* (2008), <http://www.sap.com>
28. SCOR. *Supply-Chain Operations Reference-model* (2008)
29. Becker, J., Pfeiffer, D., Räckers, M.: Domain Specific Process Modelling in Public Administrations – The PICTURE-Approach. In: Wimmer, M.A., Scholl, J., Grönlund, Å. (eds.) *EGOV*. LNCS, vol. 4656, pp. 68–79. Springer, Heidelberg (2007)
30. Weber, B., Reichert, M., Rinderle-Ma, S.: *Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems*. Data and Knowledge Engineering (accepted for publication, 2008)

Author Index

Abdulsalam, Hanady 643
 Ahn, Sungwoo 41
 Ali, Akhtar 676
 Alkobaisi, Shayma 33
 An, Yoo Jung 73
 Antonellis, Panagiotis 537
 Appice, Annalisa 283, 701

Bae, Wan D. 33
 Bakiras, Spiridon 158
 Balke, Wolf-Tilo 610
 Baltzer, Oliver 340
 Banek, Marko 65
 Belkhatir, Mohammed 465
 Bertino, Elisa 390
 Bertolotto, Michela 390
 Bessho, Katsuji 124
 Bique, Stephen 269
 Borjas, Livia 753
 Bosc, Patrick 652
 Breddin, Tino 635
 Brown, David 269

Camossi, Elena 390
 Cavaliere, Federico 718
 Ceci, Michelangelo 283, 701
 Chakravarthy, Sharma 684
 Chen, Arbee L.P. 225
 Chen, Hanxiong 693
 Chen, Lijun 241
 Chen, Qiming 106
 Chen, Yangjun 97
 Chevalletinst, Jean-Pierre 142
 Chiao, Carolina M. 837
 Cho, Chung-Wen 225
 Chun, Soon Ae 73
 Crolotte, Alain 596
 Cui, Bin 241
 Cuzzocrea, Alfredo 348

d'Amato, Claudia 808
 Dai, Bi-Ru 5
 Decker, Hendrik 89
 Dehne, Frank 340
 Deng, Qiao 19

Domínguez, Eladio 822
 du Mouza, Cédric 81
 Dyreson, Curtis 479

Eder, Johann 668
 Endo, Hiroki 186
 Esposito, Floriana 808

Falquet, Gilles 142
 Fanizzi, Nicola 808
 Farré, Carles 660
 Feng, Jianhua 19
 Fernández, Alejandro 753
 Florez, Omar U. 57
 Furfaro, Filippo 348
 Furtado, Pedro 779
 Furuse, Kazutaka 693

Geller, James 73
 Ghazal, Ahmad 596
 Ghinita, Gabriel 158
 Gkoulalas-Divanis, Aris 49
 Groch, Matthias 635
 Gruenwald, Le 172
 Guerrini, Giovanna 718

Hambruch, Susanne 340
 He, Zhenying 566
 Hess, Guillermo N. 837
 Hong, Bonghee 41
 Hou, Wen-Chi 376
 Hsu, Meichun 106
 Hsueh, Yu-Ling 419
 Huang, Kuo-chuan 73
 Hunter, Nick 172
 Hwang, Seung-won 610, 800

Ichikawa, Toshikazu 210
 Iochpe, Cirano 837
 Ishida, Kozue 255
 Iwaihara, Mizuho 552

Jiang, Zhewei 376
 Jin, Hao 479
 Jomier, Geneviève 581
 Jouini, Khaled 581

- Kabore, Patrick 581
 Kalnis, Panos 158
 Kataoka, Ryoji 124
 Kato, Yuka 186
 Kim, Bum-Soo 362
 Kim, Jinho 362
 Kim, Seon Ho 33
 Kim, Youngdae 800
 Kitagawa, Hiroyuki 255
 Kitsuregawa, Masaru 134, 196
 Köhn, Dagmar 745
 Kołaczowski, Piotr 791
 Ku, Wei-Shinn 419
 Kuo, Ya-Ping 5
 Kurashima, Takeshi 124

 Lammari, Nadira 81
 Larriba-Pey, Josep-L. 625
 Laurent, Anne 710
 Lee, Jongwuk 610
 Leutenegger, Scott T. 33
 Li, Dong (Haoyuan) 710
 Li, Jiang 523
 Li, Lin 134
 Liang, Jianyu 269
 Liang, Wenxin 508
 Lim, SeungJin 57
 Lin, Pai-Yu 5
 Liu, Chengfei 727
 Liu, Yu 19
 Lloret, Jorge 822
 Lokoč, Jakub 312
 Lu, Yang 241
 Luo, Cheng 376
 Lv, Dapeng 19

 Ma, Z.M. 116
 Maatuk, Abdelsalam 676
 Madria, Sanjay Kumar 196
 Makris, Christos 537
 Malerba, Donato 283, 701
 Manjarrez-Sanchez, Jorge 326
 Marteau, Pierre-François 150
 Martin, Patrick 643
 Martinez, José 326
 Mazzeo, Giuseppe M. 348
 McKenna, Bill 596
 Métais, Elisabeth 81
 Meersman, Robert 434
 Mesiti, Marco 718

 Miki, Takeshi 508
 Miranker, Daniel 450
 Mondal, Anirban 196
 Moon, Yang-Sae 362
 Muhammad Fuad, Muhammad Marwan 150

 Namnandorj, Sansarkhuu 693

 Ohbo, Nobuo 693
 Orleans, Luís Fernando 779
 Ortiz, Jorge 753
 Otsuka, Shingo 134

 Palpanas, Themis 770
 Papas, Marios 625
 Pivert, Olivier 652
 Poncelet, Pascal 710
 Pradhan, Subhesh 684
 Pramanik, Sakti 404

 Qian, Gang 404
 Qin, Yongrui 566

 Radhouani, Saïd 142
 Rau-Chaplin, Andrew 340
 Reichert, Manfred 837
 Roche, Mathieu 710
 Rolland, Colette 1
 Rossiter, Nick 676
 Rubio, Ángel L. 822
 Rull, Guillem 660
 Rundensteiner, Elke 269

 Sacco, Giovanni Maria 297
 Sairamesh, Jakka 770
 Sakr, Sherif 735
 Sakurai, Yasushi 210
 Sayre, Brian 269
 Seid, Dawit 596
 Selke, Joachim 610
 Seok, Hyun-Jeong 404
 Sequeda, Juan 450
 Sheybani, Ehsan 269
 Shibata, Hiroaki 186
 Sisson, Richard 269
 Skillicorn, David B. 643
 Skopal, Tomáš 312
 Strömbäck, Lena 745
 Sun, Weiwei 566

Tahamtan, Amirreza 668
 Tang, Yan 434
 Teniente, Ernest 660
 Thom, Lucinéia H. 837
 Tineo, Leonid 753
 Tirmizi, Syed Hamid 450
 Tjoa, A Min 65
 Toda, Hiroyuki 124
 Tomisawa, Satoshi 186
 Toyoda, Machiko 210
 Trancoso, Pedro 625
 Turi, Antonio 701

 Uchiyama, Toshio 124
 Urpí, Toni 660

 Valduriez, Patrick 326
 Vallur, Anita 172
 van Bommel, Martin 761
 Vanthienen, Jan 434
 Varde, Aparna 269
 Verykios, Vassilios S. 49
 Vojtěchovský, Petr 33
 Vrdoljak, Boris 65
 Vyahhi, Nikolay 158

Walzer, Karen 635
 Wang, Chi 19
 Wang, Hailong 116
 Wang, Junhu 493, 523, 727
 Wang, Ping 761
 Wu, Yi-Hung 225

 Yan, Feng 376
 Yang, Dongqing 241
 Ye, Yang 19
 Yokota, Haruo 508
 Yonei, Yumi 552
 Yoshikawa, Masatoshi 552
 You, Gae-won 610, 800
 Yu, Jeffrey Xu 493, 727
 Yu, Ping 566

 Zapata, María A. 822
 Zhang, Zhuoyao 566
 Zhao, Jiakui 241
 Zheng, Ying 225
 Zhou, Rui 727
 Zhu, Qiang 376, 404
 Zimbrão, Geraldo 779
 Zimmermann, Roger 419