

Pablo García Bringas  
Abdelkader Hameurlain  
Gerald Quirchmayr (Eds.)

LNCS 6261

# Database and Expert Systems Applications

21st International Conference, DEXA 2010  
Bilbao, Spain, August/September 2010  
Proceedings, Part I

**1**  
Part I

**DEXA 2010**

 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

*TU Dortmund University, Germany*

Madhu Sudan

*Microsoft Research, Cambridge, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbruecken, Germany*

Pablo García Bringas  
Abdelkader Hameurlain  
Gerald Quirchmayr (Eds.)

# Database and Expert Systems Applications

21st International Conference, DEXA 2010  
Bilbao, Spain, August 30 - September 3, 2010  
Proceedings, Part I

Volume Editors

Pablo García Bringas  
University of Deusto, DeustoTech Computing  
Avda. Universidades, 24, 48007 Bilbao, Spain  
E-mail: pablo.garcia.bringas@deusto.es

Abdelkader Hameurlain  
Paul Sabatier University, Institut de Recherche  
en Informatique de Toulouse (IRIT)  
118, route de Narbonne, 31062 Toulouse Cedex, France  
E-mail: hameur@irit.fr

Gerald Quirchmayr  
University of Vienna, Faculty of Computer Science  
Department of Distributed Systems and Multimedia Systems  
Liebiggasse 4/3-4, 1010 Vienna, Austria  
E-mail: gerald.quirchmayr@univie.ac.at

Library of Congress Control Number: 2010932618

CR Subject Classification (1998): H.4, I.2, H.3, C.2, H.5, J.1

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

ISSN 0302-9743  
ISBN-10 3-642-15363-1 Springer Berlin Heidelberg New York  
ISBN-13 978-3-642-15363-1 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.com

© Springer-Verlag Berlin Heidelberg 2010  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper 06/3180

## Preface

We welcome you to the proceedings of the 21<sup>st</sup> International Conference on Database and Expert Systems Applications held in Bilbao. With information and database systems being a central topic of computer science, it was to be expected that the integration of knowledge, information and data is today contributing to the again rapidly increasing attractiveness of this field for researchers and practitioners.

Since its foundation in 1990, DEXA has been an annual international conference, located in Europe, which showcases state-of-the-art research activities in these areas. DEXA 2010 continued this tradition and provided a forum for presenting and discussing research results in the area of database and intelligent systems and advanced research topics, applications and practically relevant issues related to these areas. It offered attendees the opportunity to extensively discuss requirements, problems, and solutions in the field in the pleasant atmosphere of the city of Bilbao, which is known for its driving industriousness, its top cultural venues and its rich and inspiring heritage and lifestyle. The University of Deusto with its great educational and research traditions, and the hospitality which the university and the city are so famous for, set the stage for this year's DEXA conference.

This volume contains the papers selected for presentation at the DEXA conference. DEXA 2010 attracted 197 submissions, and from these the Program Committee, based on the reviews, accepted two categories of papers: 45 regular papers and 36 short papers. Regular papers were given a maximum of 15 pages in the proceedings to report their results. Short papers were given an 8-page limit. Decisions made by members of the Program Committee were not always easy, and due to limited space a number of submissions had to be left out.

We would like to thank all those who contributed to the success of DEXA 2010: the hard work of the authors, the Program Committee, the external reviewers, and all the institutions (University of Deusto and University of Linz/FAW) that actively supported this conference and made it possible. Our special thanks go to Gabriela Wagner, manager of the DEXA organization, for her valuable help and efficiency in the realization of this conference.

We thank the DEXA Association and the University of Deusto for making DEXA 2010 a successful event. Without the continuous efforts of the General Chair, Pablo Garcia Bringas and his team, and the active support of José Luis del Val from Deusto University, this conference would not have been able to take place in the charming city of Bilbao.

June 2009

Abdelkader Hameurlain  
Gerald Quirchmayr



Danielle Boulanger	University of Lyon, France
Omar Boussaid	University of Lyon, France
Stephane Bressan	National University of Singapore, Singapore
Patrick Brezillon	University of Paris VI, France
Pablo Garcia Bringas	Deusto University Bilbao, Spain
Yingyi Bu	Microsoft, China
Luis M. Camarinha-Matos	Universidade Nova de Lisboa + Uninova, Portugal
Yiwei Cao	RWTH Aachen University, Germany
Barbara Carminati	Università degli Studi dell'Insubria, Italy
Silvana Castano	Università degli Studi di Milano, Italy
Barbara Catania	Università di Genova, Italy
Michelangelo Ceci	University of Bari, Italy
Wojciech Cellary	University of Economics at Poznan, Poland
Sharma Chakravarthy	The University of Texas at Arlington, USA
Badrish Chandramouli	Microsoft Research , USA
Cindy Chen	University of Massachusetts Lowell, USA
Phoebe Chen	Deakin University, Australia
Shu-Ching Chen	Florida International University, USA
Hao Cheng	University of Central Florida, USA
James Cheng	Nanyang Technological University, Singapore
Reynold Cheng	The University of Hong Kong, China
Max Chevalier	IRIT - SIG, Université de Toulouse, France
Byron Choi	Hong Kong Baptist University, Hong Kong
Henning Christiansen	Roskilde University, Denmark
Soon Ae Chun	City University of New York, USA
Eliseo Clementini	University of L'Aquila, Italy
Martine Collard	University of Nice, France
Gao Cong	Microsoft Research Asia, China
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Bin Cui	Peking University, China
Carlo A. Curino	Politecnico di Milano, Italy
Emiran Curtmola	University of California, San Diego, USA
Alfredo Cuzzocrea	University of Calabria, Italy
Deborah Dahl	Conversational Technologies
Violeta Damjanovic	Salzburg Research Forschungsgesellschaft m.b.H., Austria
Jérôme Darmont	Université Lumière Lyon 2, France
Valeria De Antonellis	Università di Brescia, Italy
Andre de Carvalho	University of Sao Paulo, Brazil
Vincenzo De Florio	University of Antwerp, Belgium
Guy De Tré	Ghent University, Belgium
Olga De Troyer	Vrije Universiteit Brussel, Belgium
Roberto De Virgilio	Università Roma Tre, Italy
John Debenham	University of Technology, Sydney, Australia
Hendrik Decker	Universidad Politécnica de Valencia, Spain

Zhi-Hong Deng	Peking University, China
Vincenzo Deufemia	Università degli Studi di Salerno, Italy
Claudia Diamantini	Università Politecnica delle Marche, Italy
Juliette Dibie-Barthélemy	AgroParisTech, France
Ying Ding	Indiana University, USA
Zhiming Ding	Chinese Academy of Sciences, China
Gillian Dobbie	University of Auckland, New Zealand
Peter Dolog	Aalborg University, Denmark
Dejing Dou	University of Oregon, USA
Cedric du Mouza	CNAM, France
Johann Eder	University of Vienna, Austria
Suzanne M. Embury	The University of Manchester, UK
Christian Engelmann	Oak Ridge National Laboratory, USA
Jianping Fan	University of North Carolina at Charlotte, USA
Cécile Favre	University of Lyon, France
Bettina Fazzinga	University of Calabria, Italy
Leonidas Fegaras	The University of Texas at Arlington, USA
Yaokai Feng	Kyushu University, Japan
Stefano Ferilli	University of Bari, Italy
Eduardo Fernandez	Florida Atlantic University, USA
Filomena Ferrucci	Università di Salerno, Italy
Flavius FrasinCAR	Erasmus University Rotterdam, The Netherlands
Hiroaki Fukuda	Keio University, Japan
Benjamin Fung	Concordia University, Canada
Steven Furnell	University of Plymouth, UK
Aryya Gangopadhyay	University of Maryland Baltimore County, USA
Sumit Ganguly	Indian Institute of Technology, Kanpur, India
Yunjun Gao	Zhejiang University, China
Dragan Gasevic	Athabasca University, Canada
Manolis Gergatsoulis	Ionian University, Greece
Bernard Grabot	LGP-ENIT, France
Fabio Grandi	University of Bologna, Italy
Carmine Gravino	University of Salerno, Italy
Nathan Griffiths	University of Warwick, UK
Sven Groppe	Lübeck University, Germany
Crina Grosan	Babes-Bolyai University Cluj-Napoca, Romania
William Grosky	University of Michigan, USA
Volker Gruhn	Leipzig University, Germany
Stephane Grumbach	INRIA, France
Jerzy Grzymala-Busse	University of Kansas, USA
Francesco Guerra	Università degli Studi Di Modena e Reggio Emilia, Italy
Giovanna Guerrini	University of Genova, Italy
Levent Gurgen	CEA-LETI Minattec, France
Antonella Guzzo	University of Calabria, Italy
Abdelkader Hameurlain	Paul Sabatier University, Toulouse, France



Ibrahim Hamidah	Universiti Putra Malaysia, Malaysia
Wook-Shin Han	Kyungpook National University, Korea
Takahiro Hara	Osaka University, Japan
Theo Härder	TU Kaiserslautern, Germany
Saven He	Microsoft Research at Asia, China
Francisco Herrera	University of Granada, Spain
Birgit Hofreiter	University of Liechtenstein, Liechtenstein
Steven Hoi	Nanyang Technological University, Singapore
Estevam Rafael Hruschka Jr	Carnegie Mellon University, USA
Wynne Hsu	National University of Singapore, Singapore
Yu Hua	Huazhong University of Science and Technology, China
Jimmy Huang	York University, Canada
Xiaoyu Huang	South China University of Technology, China
San-Yih Hwang	National Sun Yat-Sen University, Taiwan
Ionut Emil Iacob	Georgia Southern University, USA
Renato Iannella	National ICT Australia (NICTA), Australia
Sergio Ilarri	University of Zaragoza, Spain
Abdessamad Imine	University of Nancy, France
Yoshiharu Ishikawa	Nagoya University, Japan
Mizuho Iwaihara	Waseda University, Japan
Adam Jatowt	Kyoto University, Japan
Peiquan Jin	University of Science and Technology, China
Jan Jurjens	Open University and Microsoft Research, UK
Ejub Kajan	State University of Novi Pazar, Serbia
Ken Kaneiwa	National Institute of Information and Communications Technology(NICT), Japan
Anne Kao	Boeing Phantom Works, USA
Dimitris Karagiannis	University of Vienna, Austria
Stefan Katzenbeisser	Technical University of Darmstadt, Germany
Yiping Ke	Chinese University of Hong Kong, Hong Kong
Myoung Ho Kim	KAIST, Korea
Sang-Wook Kim	Hanyang University, Korea
Markus Kirchberg	Institute for Infocomm Research, A*STAR, Singapore
Hiroyuki Kitagawa	University of Tsukuba, Japan
Carsten Kleiner	University of Applied Sciences&Arts Hannover, Germany
Ibrahim Korpeoglu	Bilkent University, Turkey
Harald Kosch	University of Passau, Germany
Michal Krátký	VSB-Technical University of Ostrava, Czech Republic
Petr Kroha	Technische Universität Chemnitz-Zwickau, Germany
Arun Kumar	IBM India Research Lab., India
Ashish Kundu	Purdue University, USA

Josef Küng	University of Linz, Austria
Lotfi Lakhali	University of Marseille, France
Kwok-Wa Lam	University of Hong Kong, Hong Kong
Nadira Lammari	CNAM, France
Gianfranco Lamperti	University of Brescia, Italy
Andreas Langegger	University of Linz, Austria
Anne Laurent	LIRMM, University of Montpellier 2, France
Mong Li Lee	National University of Singapore, Singapore
Young-Koo Lee	Kyung Hee University, Korea
Alain Toiron Leger	Orange - France Telecom R&D, France
Daniel Lemire	Université du Québec à Montréal, Canada
Scott Leutenegger	University of Denver, USA
Pierre Lévy	Public Health Department, France
Lenka Lhotska	Czech Technical University, Czech Republic
Wenxin Liang	Dalian University of Technology, China
Lipyew Lim	IBM T. J. Watson Research Center, USA
Hong Lin	University of Houston-Downtown, USA
Tok Wang Ling	National University of Singapore, Singapore
Sebastian Link	Victoria University of Wellington, New Zealand
Volker Linnemann	University of Lübeck, Germany
Chengfei Liu	Swinburne University of Technology, Australia
Chuan-Ming Liu	National Taipei University of Technology, Taiwan
Fuyu Liu	University of Central Florida, USA
Hong-Cheu Liu	Diwan University, Taiwan
Hua Liu	Xerox Research Center at Webster, USA
Jorge Lloret Gazo	University of Zaragoza, Spain
Miguel Ángel López Carmona	University of Alcalá de Henares, Spain
Peri Loucopoulos	Loughborough University, UK
Chang-Tien Lu	Virginia Tech, USA
James J. Lu	Emory University, Atlanta, USA
Jianguo Lu	University of Windsor, Canada
Alessandra Lumini	University of Bologna, Italy
Qiang Ma	Kyoto University, Japan
Stéphane Maag	TELECOM & Management SudParis, France
Nikos Mamoulis	University of Hong Kong, Hong Kong
Vladimir Marik	Czech Technical University, Czech Republic
Elio Masciari	ICAR-CNR, Italy
Norman May	SAP Research Center, Germany
Jose-Norberto Mazon	University of Alicante in Spain, Spain
Dennis McLeod	University of Southern California, USA
Brahim Medjahed	University of Michigan - Dearborn, USA
Carlo Meghini	ISTI-CNR, Italy
Rosa Meo	University of Turin, Italy
Farid Meziane	Salford University, UK
Harekrishna Misra	Institute of Rural Management Anand, India

Jose Mocito	INESC-ID/FCUL, Portugal
Lars Mönch	FernUniversität in Hagen, Germany
Anirban Mondal	University of Tokyo, Japan
Hyun Jin Moon	UCLA Computer Science, USA
Yang-Sae Moon	Kangwon National University, Korea
Reagan Moore	San Diego Supercomputer Center, USA
Franck Morvan	IRIT, Paul Sabatier University, Toulouse, France
Yi Mu	University of Wollongong, Australia
Mirco Musolesi	University of Cambridge, UK
Tadashi Nakano	University of California, Irvine, USA
Ullas Nambiar	IBM India Research Lab, India
Ismael Navas-Delgado	University of Málaga, Spain
Wilfred Ng	University of Science & Technology, Hong Kong
Christophe Nicolle	University of Burgundy, France
Javier Nieves Acedo	Deusto University, Spain
Selim Nurcan	University of Paris 1 Pantheon Sorbonne, France
Joann J. Ordille	Avaya Labs Research, USA
Mehmet Orgun	Macquarie University, Australia
Luís Fernando Orleans	Federal University of Rio de Janeiro, Brazil
Mourad Oussalah	University of Nantes, France
Gultekin Ozsoyoglu	Case Western Reserve University, USA
George Pallis	University of Cyprus, Cyprus
Christos Papatheodorou	Ionian University, Corfu, Greece
Paolo Papotti	Università Roma Tre, Italy
Marcin Paprzycki	Polish Academy of Sciences, Warsaw Management Academy, Poland
Oscar Pastor	Universidad Politecnica de Valencia, Spain
Jovan Pehcevski	MIT University, Skopje, Macedonia
David Pinto	BUAP University, Mexico
Clara Pizzuti	CNR, ICAR, Italy
Jaroslav Pokorny	Charles University in Prague, Czech Republic
Giuseppe Polese	University of Salerno, Italy
Pascal Poncelet	LIRMM, France
Elaheh Pourabbas	National Research Council, Italy
Xiaojun Qi	Utah State University, USA
Fausto Rabitti	ISTI, CNR Pisa, Italy
Claudia Raibulet	Università degli Studi di Milano-Bicocca, Italy
Isidro Ramos	Technical University of Valencia, Spain
Praveen Rao	University of Missouri Kansas City, USA
Jan Recker	Queensland University of Technology, Australia
Manjeet Rege	Rochester Institute of Technology, USA
Rodolfo F. Resende	Federal University of Minas Gerais, Brazil
Claudia Roncancio	Grenoble University / LIG, France
Kamel Rouibah	College of Business Administration, Kuwait
Edna Ruckhaus	Universidad Simon Bolivar, Venezuela
Massimo Ruffolo	University of Calabria, Italy

Igor Ruiz Agúndez	Deusto University, Spain
Giovanni Maria Sacco	University of Turin, Italy
Fereidoon (Fred) Sadri	University of North Carolina at Greensboro, USA
Simonas Saltenis	Aalborg University, Denmark
Jose Samos	Universidad de Granada, Spain
Demetrios G. Sampson	University of Piraeus, Greece
Carlo Sansone	Università di Napoli "Federico II", Italy
Paolo Santi	Istituto di Informatica e Telematica, Italy
Igor Santos Grueiro	Deusto University, Spain
Ismael Sanz	Universitat Jaume I, Spain
N.L. Sarda	I.I.T. Bombay, India
Marinette Savonnet	University of Burgundy, France
Raimondo Schettini	Università degli Studi di Milano-Bicocca, Italy
Erich Schweighofer	University of Vienna, Austria
Florence Sedes	IRIT Toulouse, France
Nazha Selmaoui	University of New Caledonia, France
Patrick Siarry	Université Paris 12 (LiSSi), France
Gheorghe Cosmin Silaghi	Babes-Bolyai University of Cluj-Napoca, Romania
Hala Skaff-Molli	Université Henri Poincaré, France
Giovanni Soda	University of Florence, Italy
Leonid Sokolinsky	South Ural State University, Russia
MoonBae Song	Samsung Electronics, Korea
Adrian Spalka	CompuGROUP Holding AG, Germany
Bala Srinivasan	Monash University, Australia
Umberto Straccia	Italian National Research Council, Italy
Darijus Strasunskas	Norwegian University of Science and Technology (NTNU), Norway
Martin J. Strauss	Michigan University, USA
Lena Stromback	Linköpings Universitet, Sweden
Aixin Sun	Nanyang Technological University, Singapore
Raj Sunderraman	Georgia State University, USA
Ashish Sureka	Infosys Technologies Limited, India
Jun Suzuki	University of Massachusetts, Boston, USA
Jie Tang	Tsinghua University, China
David Taniar	Monash University, Australia
Cui Tao	Brigham Young University, USA
Maguelonne Teisseire	LIRMM, University of Montpellier 2, France
Sergio Tessaris	Free University of Bozen-Bolzano, Italy
Olivier Teste	IRIT, University of Toulouse, France
Stephanie Teufel	University of Fribourg, Switzerland
Jukka Teuhola	University of Turku, Finland
Taro Tezuka	Ritsumeikan University, Japan
J.M. Thevenin	University of Toulouse, France
Philippe Thiran	University of Namur, Belgium
Helmut Thoma	University of Basel, Switzerland

A Min Tjoa	Technical University of Vienna, Austria
Andreas Tolk	Old Dominion University, USA
Vicenc Torra	Universitat Autònoma de Barcelona, Spain
Traian Truta	Northern Kentucky University, USA
Christos Tryfonopoulo	Max-Planck Institute for Informatics, Germany
Vassileios Tsetso	National and Kapodistrian University of Athens, Greece
Theodoros Tzouramanis	University of the Aegean, Greece
Roberto Uribeetxebarria	Mondragon University, Spain
Marc van Kreveld	Utrecht University, The Netherlands
Genoveva Vargas-Solar	LSR-IMAG, France
Maria Vargas-Vera	The Open University, UK
Krishnamurthy Vidyasankar	Memorial University of Newfoundland, Canada
Marco Vieira	University of Coimbra, Portugal
Johanna Völker	University of Karlsruhe, Germany
Jianyong Wang	Tsinghua University, China
Junhu Wang	Griffith University, Brisbane, Australia
Kewen Wang	Griffith University, Brisbane, Australia
Wei Wang	University of New South Wales, Sydney, Australia
Wendy Hui Wang	Stevens Institute of Technology, USA
Andreas Wombacher	University Twente, The Netherlands
Songhua Xing	University of Southern California, USA
Jianliang Xu	Hong Kong Baptist University, Hong Kong
Lai Xu	SAP Research, Switzerland
Hsin-Chang Yang	National University of Kaohsiung, Taiwan
Ming Hour Yang	Chung Yuan Christian University, Taiwan
Xiaochun Yang	Northeastern University, China
Haruo Yokota	Tokyo Institute of Technology, Japan
Zhiwen Yu	Northwestern Polytechnical University, China
Gian Piero Zarri	University of Paris IV, Sorbonne, France
Xiao-Jun Zeng	University of Manchester, UK
Zhigang Zeng	Huazhong University of Science and Technology, China
Xiuzhen (Jenny) Zhang	RMIT University Australia, Australia
Yanchang Zhao	University of Technology, Sydney, Australia
Yu Zheng	Microsoft Research Asia, China
Xiao Ming Zhou	Sybase Engineering
Qiang Zhu	The University of Michigan, USA
Yi Zhuang	Zhejiang Gongshang University, China
Ester Zumpano	University of Calabria, Italy
Urko Zurutuza	Mondragon University, Spain

## External Reviewers

Shafiq Alam	Ming Fang	Michele Melchiori
Amin Anjomshoaa	Nikolaos Fousteris	Sumit Mittal
Yuki Arase	Filippo Furfaro	Riad Mokadem
Zahoua Aoussat	George Giannopoulos	Sébastien Nedjar
M. Asif Naeem	Sergio A. Gómez	Afonso Araujo Neto
Naser Ayat	Shen Ge	Ermelinda Oro
Zhifeng Bao	Alberto Gemelli	Ruggero G. Pensa
Nicolas Béchet	Massimiliano Giordano	Yoann Pitarch
Tom Heydt-Benjamin	Wang Hao	Domenico Potena
Jorge Bernardino	Ben He	Han Qin
Vineetha Bettaiah	Yukai He	Julien Rabatel
Paulo Alves Braz	Tok Wee Hyong	Hassan Sanaifar
Souad Boukheddouma	Vivian Hu	Federica Sarro
Laurynas Biveinis	Dino Ienco	Pasquale Savino
Guillaume Cabanac	Evan Jones	Wei Shen
Marc Chastand	Christine Julien	Chad M.S. Steel
Himanshu Chauhan	Hideyuki Kawashima	Umberto Straccia
Yi Chen	Nikos Kiourtis	Jafar Tanha
Anna Ciampi	Cyril Labbé	Manolis Terrovitis
Camelia Constantin	Maria Laura Maag	Ronan Tournier
Lucie Copin	Ki Yong Lee	Antoine Veillard
Nadine Cullot	Jianxin Li	Daya Wimalasuriya
Theodore Dalamagas	Jing Li	Huayu Wu
Claus Dabringer	Wei Li	Kefeng Xuan
Enrique de la Hoz	Weiling Li	Jun Zhang
Kuntal Dey	Haishan Liu	Geng Zhao
Raimundo F. Dos Santos	Xutong Liu	Rui Zhou
Fabrizio Falchi	Ivan Marsa-Maestre	

# Table of Contents – Part I

## Web, Semantics and Ontologies

Vi-DIFF: Understanding Web Pages Changes . . . . .	1
<i>Zeynep Pehlivan, Myriam Ben-Saad, and Stéphane Gançarski</i>	
Automatic Fuzzy Semantic Web Ontology Learning from Fuzzy Object-Oriented Database Model . . . . .	16
<i>Fu Zhang, Z.M. Ma, Gaofeng Fan, and Xing Wang</i>	
Combining Fuzzy Logic and Semantic Web to Enable Situation-Awareness in Service Recommendation . . . . .	31
<i>Alessandro Ciaramella, Mario G.C.A. Cimino, Francesco Marcelloni, and Umberto Straccia</i>	

## Information Integration and Process Modelling

Source Selection in Large Scale Data Contexts: An Optimization Approach . . . . .	46
<i>Alexandra Pomares, Claudia Roncancio, Van-Dat Cung, José Abásolo, and María-del-Pilar Villamil</i>	
Semi-automatic Discovery of Web Services Driven by User Requirements . . . . .	62
<i>María Pérez, Ismael Sanz, Rafael Berlanga, and María José Aramburu</i>	
An Open Platform for Business Process Modeling and Verification . . . . .	76
<i>Antonio De Nicola, Michele Missikoff, Maurizio Proietti, and Fabrizio Smith</i>	

## Streams and Multimedia Databases

Discovering Highly Informative Feature Sets from Data Streams . . . . .	91
<i>Chongsheng Zhang and Florent Massegia</i>	
Continuous Probabilistic Skyline Queries over Uncertain Data Streams . . . . .	105
<i>Hui Zhu Su, En Tzu Wang, and Arbee L.P. Chen</i>	
DBOD-DS: Distance Based Outlier Detection for Data Streams . . . . .	122
<i>Md. Shiblee Sadik and Le Gruenwald</i>	

Effective Bitmap Indexing for Non-metric Similarities . . . . .	137
<i>Claus A. Jensen, Ester M. Mungure, Torben Bach Pedersen, Kenneth Sørensen, and François Delière</i>	

## Data Management Algorithms and Performance

Efficient Incremental Near Duplicate Detection Based on Locality Sensitive Hashing . . . . .	152
<i>Marco Fisichella, Fan Deng, and Wolfgang Nejdl</i>	
Efficient Fuzzy Top- $\kappa$ Query Processing over Uncertain Objects . . . . .	167
<i>Chuanfei Xu, Yanqiu Wang, Shukuan Lin, Yu Gu, and Jianzhong Qiao</i>	
Performance and Power Evaluation of Flash-Aware Buffer Algorithms . . . . .	183
<i>Yi Ou, Theo Härder, and Daniel Schall</i>	
Towards Efficient Concurrent Scans on Flash Disks . . . . .	198
<i>Chang Xu, Lidan Shou, Gang Chen, Wei Hu, Tianlei Hu, and Ke Chen</i>	

## Decision Support Systems and Performance (Short Papers)

Enhanced Foundry Production Control . . . . .	213
<i>Javier Nieves, Igor Santos, Yoseba K. Peña, Felix Brezo, and Pablo G. Bringas</i>	
The Profile Distance Method: Towards More Intuitive Multiple Criteria Decision Making in Information Systems . . . . .	221
<i>Edward Bernroider, Nikolaus Obwegeser, and Volker Stix</i>	
Hybrid In-Memory and On-Disk Tables for Speeding-Up Table Accesses . . . . .	231
<i>Joan Guisado-Gómez, Antoni Wolski, Calisto Zuzarte, Josep-Lluís Larriba-Pey, and Victor Muntés-Mulero</i>	
Low Power Management of OLTP Applications Considering Disk Drive Power Saving Function . . . . .	241
<i>Norifumi Nishikawa, Miyuki Nakano, and Masaru Kitsuregawa</i>	

## Data Streams (Short Papers)

A Disk-based, Adaptive Approach to Memory-Limited Computation of Windowed Stream Joins . . . . .	251
<i>Abhirup Chakraborty and Ajit Singh</i>	



Prediction Functions in Bi-temporal Datastreams . . . . .	261
<i>André Bolles, Marco Grawunder, Jonas Jacobi, Daniela Nicklas, and H. -Jürgen Appelrath</i>	
A Load Shedding Framework for XML Stream Joins . . . . .	269
<i>Ranjan Dash and Leonidas Fegaras</i>	
Supporting Multi-criteria Decision Support Queries over Time-Interval Data Streams . . . . .	281
<i>Nam Hun Park, Venkatesh Raghavan, and Elke A. Rundensteiner</i>	
<b>XML Databases, Programming Language and Cooperative Work (Short Papers)</b>	
Faster Algorithms for Searching Relevant Matches in XML Databases . . . . .	290
<i>Rung-Ren Lin, Ya-Hui Chang, and Kun-Mao Chao</i>	
Efficient SLCA-Based Keyword Search on XML Databases: An Iterative-Skip Approach . . . . .	298
<i>Jiaqi Zhang, Zhenying He, Yue Tao, Xiansheng Wang, Hao Hu, and Wei Wang</i>	
SFL: A Structured Dataflow Language Based on SQL and FP . . . . .	306
<i>Qiming Chen and Meichun Hsu</i>	
Incorporating Data Provenance in a Medical CSCW System . . . . .	315
<i>Konrad Stark, Christian Koncilia, Jonas Schulte, Erich Schikuta, and Johann Eder</i>	
<b>Query Processing and Optimization (Short Papers)</b>	
Optimization of Object-Oriented Queries through Rewriting Compound Weakly Dependent Subqueries . . . . .	323
<i>Michał Bleja, Tomasz Kowalski, and Kazimierz Subieta</i>	
A Utilization of Schema Constraints to Transform Predicates in XPath Query . . . . .	331
<i>Dung Xuan Thi Le, Stéphane Bressan, Eric Pardede, David Taniar, and Wenny Rahayu</i>	
Merging Views Containing Outer Joins in the Teradata DBMS . . . . .	340
<i>Ahmad Ghazal, Dawit Seid, and Alain Crotte</i>	
A Hybrid Index Structure for Set-Valued Attributes Using Itemset Tree and Inverted List . . . . .	349
<i>Shahriyar Hossain and Hasan Jamil</i>	

Optimization of Disk Accesses for Multidimensional Range Queries . . . . .	358
<i>Peter Chovanec, Michal Krátký, and Radim Bača</i>	

## Data Privacy and Security

A Privacy-Enhancing Content-Based Publish/Subscribe System Using Scalar Product Preserving Transformations . . . . .	368
<i>Sunoh Choi, Gabriel Ghinita, and Elisa Bertino</i>	

Synthesizing: Art of Anonymization . . . . .	385
<i>Jun Gu, Yuxian Chen, Junning Fu, Huanchun Peng, and Xiaojun Ye</i>	

Anonymizing Transaction Data to Eliminate Sensitive Inferences . . . . .	400
<i>Grigorios Loukides, Aris Gkoulalas-Divanis, and Jianhua Shao</i>	

## Temporal, Spatial, and High Dimensional Databases

Interval-Orientation Patterns in Spatio-temporal Databases . . . . .	416
<i>Dhaval Patel</i>	

Efficient K-Nearest Neighbor Search in Time-Dependent Spatial Networks . . . . .	432
<i>Ugur Demiryurek, Farnoush Banaei-Kashani, and Cyrus Shahabi</i>	

Hybrid Indexing and Seamless Ranking of Spatial and Textual Features of Web Documents . . . . .	450
<i>Ali Khodaei, Cyrus Shahabi, and Chen Li</i>	

Understanding the Human Genome: A Conceptual Modeling-Based Approach (Extended Abstract) . . . . .	467
<i>Oscar Pastor</i>	

## Semantic Web and Ontologies (Short Papers)

Learning Semantic N-Ary Relations from Wikipedia . . . . .	470
<i>Marko Banek, Damir Jurić, and Zoran Škočir</i>	

RIF Centered Rule Interchange in the Semantic Web . . . . .	478
<i>Xing Wang, Z.M. Ma, Fu Zhang, and Li Yan</i>	

f-SPARQL: A Flexible Extension of SPARQL . . . . .	487
<i>Jingwei Cheng, Z.M. Ma, and Li Yan</i>	

Geo Linked Data . . . . .	495
<i>Francisco J. Lopez-Pellicer, Mário J. Silva, Marcirio Chaves, F. Javier Zarazaga-Soria, and Pedro R. Muro-Medrano</i>	

Approximate Instance Retrieval on Ontologies . . . . .	503
<i>Tuvshintur Tserendorj, Stephan Grimm, and Pascal Hitzler</i>	
Removing the Redundancy from Distributed Semantic Web Data . . . . .	512
<i>Ahmad Ali Iqbal, Maximilian Ott, and Aruna Seneviratne</i>	
<b>Author Index</b> . . . . .	521

## Table of Contents – Part II

### Data Mining Systems

Mining and Explaining Relationships in Wikipedia . . . . .	1
<i>Xinpeng Zhang, Yasuhito Asano, and Masatoshi Yoshikawa</i>	
Publishing Time-Series Data under Preservation of Privacy and Distance Orders . . . . .	17
<i>Yang-Sae Moon, Hea-Suk Kim, Sang-Pil Kim, and Elisa Bertino</i>	
Efficient Discovery of Generalized Sentinel Rules . . . . .	32
<i>Morten Middelfart, Torben Bach Pedersen, and Jan Krogsgaard</i>	

### Parallelism and Query Planning

Compound Treatment of Chained Declustered Replicas Using a Parallel Btree for High Scalability and Availability . . . . .	49
<i>Min Luo, Akitsugu Watanabe, and Haruo Yokota</i>	
Query Reuse Based Query Planning for Searches over the Deep Web . . .	64
<i>Fan Wang and Gagan Agrawal</i>	
Efficient Parallel Data Retrieval Protocols with MIMO Antennae for Data Broadcast in 4G Wireless Communications . . . . .	80
<i>Yan Shi, Xiaofeng Gao, Jiaofei Zhong, and Weili Wu</i>	

### Data Warehousing and Decision Support Systems

Inferring Aggregation Hierarchies for Integration of Data Marts . . . . .	96
<i>Dariusz Riazati, James A. Thom, and Xiuzhen Zhang</i>	
Schema Design Alternatives for Multi-granular Data Warehousing . . . . .	111
<i>Nadeem Iftikhar and Torben Bach Pedersen</i>	
An Agent Model of Business Relationships . . . . .	126
<i>John Debenham and Carles Sierra</i>	

### Temporal, Spatial and High Dimensional Databases (Short Papers)

Pivot Selection Method for Optimizing both Pruning and Balancing in Metric Space Indexes . . . . .	141
<i>Hisashi Kurasawa, Daiji Fukagawa, Atsuhiko Takasu, and Jun Adachi</i>	

Minimum Spanning Tree on Spatio-Temporal Networks . . . . .	149
<i>Viswanath Gunturi, Shashi Shekhar, and Arnab Bhattacharya</i>	

**Data Warehousing and Data Mining Algorithms  
(Short Papers)**

Real-Time Temporal Data Warehouse Cubing . . . . .	159
<i>Usman Ahmed, Anne Tchounikine, Maryvonne Miquel, and Sylvie Servigne</i>	

PAGER: Parameterless, Accurate, Generic, Efficient <i>k</i> NN-Based Regression . . . . .	168
<i>Himanshu Singh, Aditya Desai, and Vikram Pudi</i>	

B2R: An Algorithm for Converting Bayesian Networks to Sets of Rules . . . . .	177
<i>Bartłomiej Śnieżyński, Tomasz Łukasik, and Marek Mierzwa</i>	

Automatic Morphological Categorisation of Carbon Black Nano-aggregates . . . . .	185
<i>Juan López-de-Uralde, Iraide Ruiz, Igor Santos, Agustín Zubillaga, Pablo G. Bringas, Ana Okariz, and Teresa Guraya</i>	

**Data Mining Algorithms (I)**

Towards Efficient Mining of Periodic-Frequent Patterns in Transactional Databases . . . . .	194
<i>R. Uday Kiran and P. Krishna Reddy</i>	

Lag Patterns in Time Series Databases . . . . .	209
<i>Dhaval Patel, Wynne Hsu, Mong Li Lee, and Srinivasan Parthasarathy</i>	

An Efficient Computation of Frequent Queries in a Star Schema . . . . .	225
<i>Cheikh Tidiane Dieng, Tao-Yuan Jen, and Dominique Laurent</i>	

**Information Retrieval and Database Systems  
(Short Papers)**

Evaluating Evidences for Keyword Query Disambiguation in Entity Centric Database Search . . . . .	240
<i>Elena Demidova, Xuan Zhou, Irina Oelze, and Wolfgang Nejdl</i>	

Typicality Ranking of Images Using the Aspect Model . . . . .	248
<i>Taro Tezuka and Akira Maeda</i>	

Plus One or Minus One: A Method to Browse from an Object to Another Object by Adding or Deleting an Element . . . . .	258
<i>Kosetsu Tsukuda, Takehiro Yamamoto, Satoshi Nakamura, and Katsumi Tanaka</i>	
Using Transactional Data from ERP Systems for Expert Finding . . . . .	267
<i>Lars K. Schunk and Gao Cong</i>	
A Retrieval Method for Earth Science Data Based on Integrated Use of Wikipedia and Domain Ontology . . . . .	277
<i>Masashi Tatedoko, Toshiyuki Shimizu, Akinori Saito, and Masatoshi Yoshikawa</i>	

## Query Processing and Optimization

Consistent Answers to Boolean Aggregate Queries under Aggregate Constraints . . . . .	285
<i>Sergio Flesca, Filippo Furfaro, and Francesco Parisi</i>	
Identifying Interesting Instances for Probabilistic Skylines . . . . .	300
<i>Yinian Qi and Mikhail Atallah</i>	
GPU-WAH: Applying GPUs to Compressing Bitmap Indexes with Word Aligned Hybrid . . . . .	315
<i>Witold Andrzejewski and Robert Wrembel</i>	
Containment of Conjunctive Queries with Negation: Algorithms and Experiments . . . . .	330
<i>Khalil Ben Mohamed, Michel Leclère, and Marie-Laure Mugnier</i>	

## Application of DB Systems, Similarity Search and XML

Ranking Objects Based on Attribute Value Correlation . . . . .	346
<i>Jaehui Park and Sang-goo Lee</i>	
Efficiently Finding Similar Objects on Ontologies Using Earth Mover’s Distance . . . . .	360
<i>Mala Saraswat</i>	
Towards a “More Declarative” XML Query Language . . . . .	375
<i>Xuhui Li, Mengchi Liu, and Yongfa Zhang</i>	
Reducing Graph Matching to Tree Matching for XML Queries with ID References . . . . .	391
<i>Huayu Wu, Tok Wang Ling, Gillian Dobbie, Zhifeng Bao, and Liang Xu</i>	

**Data Mining Algorithms (II)**

Improving Alternative Text Clustering Quality in the Avoiding Bias Task with Spectral and Flat Partition Algorithms . . . . . 407  
*M. Eduardo Ares, Javier Parapar, and Álvaro Barreiro*

An Efficient Similarity Join Algorithm with Cosine Similarity Predicate . . . . . 422  
*Dongjoo Lee, Jaehui Park, Junho Shim, and Sang-goo Lee*

An Efficient Algorithm for Reverse Furthest Neighbors Query with Metric Index . . . . . 437  
*Jianquan Liu, Hanxiong Chen, Kazutaka Furuse, and Hiroyuki Kitagawa*

**Pervasive Data and Sensor Data Management (Short Papers)**

A Scalable and Self-adapting Notification Framework . . . . . 452  
*Anthony Okorodudu, Leonidas Fegaras, and David Levine*

Enrichment of Raw Sensor Data to Enable High-Level Queries . . . . . 462  
*Kenneth Conroy and Mark Roantree*

**Data Mining Algorithms (III)**

Transductive Learning from Textual Data with Relevant Example Selection . . . . . 470  
*Michelangelo Ceci*

A Discretization Algorithm for Uncertain Data . . . . . 485  
*Jiaqi Ge, Yuni Xia, and Yicheng Tu*

**Author Index** . . . . . 501

# Vi-DIFF: Understanding Web Pages Changes

Zeynep Pehlivan, Myriam Ben-Saad, and Stéphane Gançarski

LIP6, University P. and M. Curie Paris, France

{zeynep.pehlivan,myriam.ben-saad,stephane.gancarski}@lip6.fr

**Abstract.** Nowadays, many applications are interested in detecting and discovering changes on the web to help users to understand page updates and more generally, the web dynamics. Web archiving is one of these fields where detecting changes on web pages is important. Archiving institutes are collecting and preserving different web site versions for future generation. A major problem encountered by archiving systems is to understand what happened between two versions of web pages. In this paper, we address this requirement by proposing a new change detection approach that computes the semantic differences between two versions of HTML web pages. Our approach, called Vi-DIFF, detects changes on the visual representation of web pages. It detects two types of changes: content and structural changes. Content changes include modifications on text, hyperlinks and images. In contrast, structural changes alter the visual appearance of the page and the structure of its blocks. Our Vi-DIFF solution can serve for various applications such as crawl optimization, archive maintenance, web changes browsing, etc. Experiments on Vi-DIFF were conducted and the results are promising.

**Keywords:** Web dynamics, Web archiving, Change detection, Web page versions, Visual page segmentation.

## 1 Introduction

The World Wide Web (or web for short) is constantly evolving over time. Web contents like texts, images, etc. are updated frequently. As those updates reflect the evolution of sites, many applications, nowadays, aim at discovering and detecting changes on the web. For instance, there are many services [4] that track web pages to identify what and how a page of interests has been changed and notify concerned users. One of the recent work is proposed by Kukulenz et al. [14]. They propose two strategies: tracking where a number of segments is traced over time and pruning where less relevant segments are removed automatically. A browser plug-in [20], using the page cache, has been also proposed to explicitly highlight changes on the current page since the last visit. The goal of such applications is to help users to understand the meaning of page changes and more generally, the web dynamics. Detecting changes between page versions is also important for web archiving. As web contents provide useful knowledge, archiving the web has become crucial to prevent pages content from disappearing. Thus, many national archiving institutes [25,7,11] around the world are



collecting and preserving different web sites versions. Most of the web archiving initiatives are described at [1]. Most often, web archiving is automatically performed using web crawlers. A crawler revisits periodically web pages and updates the archive with a fresh snapshot (or version). However, it is impossible to maintain a complete archive of the web, or even a part of it, containing all the versions of the pages because of web sites politeness constraints and limited allocated resources (bandwidth, space storage, etc.). Thus, archiving systems must identify accurately how the page has been updated at least for three reasons: (1) avoid archiving several times the same version, (2) model the dynamics of the web site in order to archive “the most important versions” by crawling the site at a “right moment”, (3) ease temporal querying the archive.

Thus, our research problem can be stated as follows: *How to know and to understand what happened (and thus changed) between two versions of web page ?*

To address this issue, we propose, a change detection approach to compute the semantic differences between two versions of web pages. As we know, most of the pages on the web are HTML documents. As HTML is semantically poor, comparing two versions of HTML pages, based on the DOM tree does not give a relevant information to understand the changes. Thus, our idea is to detect changes on the visual representation of web pages. Indeed, the visual aspect gives a good idea of the semantic structure used in the document and the relationship between them (*e.g.* the most important information is in the center of the page). Preserving the visual aspect of web pages while detecting changes gives relevant information to understand the modifications. It simulates how a user understands the changes based on his visual perception. The concept of analyzing the visual aspect of web pages is not new. However, as far as we know, it had never been exploited to detect changes on web pages.

Our proposed approach, called Vi-DIFF, compares two web pages in three steps: (i) segmentation, (ii) change detection and (iii) delta file generation. The segmentation step consists in partitioning web pages into visual semantic blocks. Then, in the change detection step, the two restructured versions of web pages are compared and, finally, a delta file describing the visual changes is produced.

The main contributions of this paper are the following:

- A novel change detection approach (Vi-DIFF) that describes the semantic difference between “visually restructured” web pages.
- An extension of an existing visual segmentation model to build the whole visual structure of the web page.
- A change detection algorithm to compare the two restructured versions of web pages that takes into account the page structure with a reasonable complexity in time.
- An implementation of Vi-DIFF approach and some experiments to demonstrate its feasibility.

The remainder of the paper is structured as follows. Section 2 discusses some related works and presents different contexts in which our proposed approach can be exploited. Section 3 presents the VI-DIFF and the different change operations

we consider. Section 4 explains, in detail, the second step of Vi-DIFF that computes the semantic difference between two restructured versions of web pages. Section 5, presents the implementation of VI-DIFF and discusses experimental results. Section 6 concludes.

## 2 Context and Related Works

The first step of our approach is the segmentation of the web page. Several methods have been proposed to construct the visual representation of web pages. Most approaches discover the logical structure of a page by analyzing the rendered document or analyzing the document code. Gu et al. [12] propose a top-down algorithm which detects the web content structure based on the layout information. Kovacevic et al. [10] define heuristics to recognize common page areas (header, footer, center of the page, etc.) based on visual information. Cai et al. [6] propose the algorithm VIPS which segments the web page into multiple semantic blocks based on visual information retrieved from browser’s rendering. Cosulshi et al. [9] propose an approach that calculates the block correspondence between web pages by using positional information of DOM tree’s elements. Kukulenz et al.’s automatic page segmentation [14] is based on the estimation of the grammatical structure of pages automatically. Among these visual analysis methods, VIPS algorithm [6] seems to be the most appropriate for our approach because it allows an adequate granularity of the page partitioning. It builds a hierarchy of semantic blocks of the page that simulates well how a user understands the web layout structure based on his visual perception. We extended VIPS algorithm to complete the semantic structure of the page by extracting links, images and text for each block. At the end of the segmentation step, an XML document, called Vi-XML, describing the complete visual structure of the web pages, is produced.

In the change detection step, two Vi-XML files corresponding to two versions of a web page are compared. Many existing algorithms have been specially designed to detect changes between two semi structured documents. They find the minimum set of changes (delete, insert, update), described in a delta file, that transform one data tree to another. Cobéna et al. [8] propose the XyDiff algorithm to improve time and memory management. XyDiff supports a move in addition to basic operations. It achieves a time complexity of  $O(n * \log(n))$ . Despite its high performance, it does not always guarantee an optimal result (*i.e.* minimal edit script). Wang et al. [21] propose X-Diff which detects the optimal differences between two unordered XML trees in quadratic time  $O(n^2)$ . DeltaXML [15] can compare, merge and synchronize XML documents for ordered and unordered trees by supporting basic operations. Both X-Diff and DeltaXML do not handle a move operation. There are several other algorithms like Fast XML DIFF [17], DTD-Diff [16], etc. After studying these algorithms, we decided to not use existing methods for our approach because they are generic-purpose. As we have various specific requirements related to the visual layout structure of web pages,

we prefer proposing our *ad hoc* diff algorithm. As it is designed for one given specific type of document, it allows for a better trade-off between complexity and completeness of the detected operations set. Our proposed change detection step in Vi-DIFF algorithm detects structural and content changes. *Structural changes* alter the visual appearance of the page and the structure of its blocks. In contrast, *content changes* modify text, hyperlinks and images inside blocks of the page. Then, it constructs a delta file describing all these changes.

Our solution for detecting changes between two web pages versions can serve for various applications:

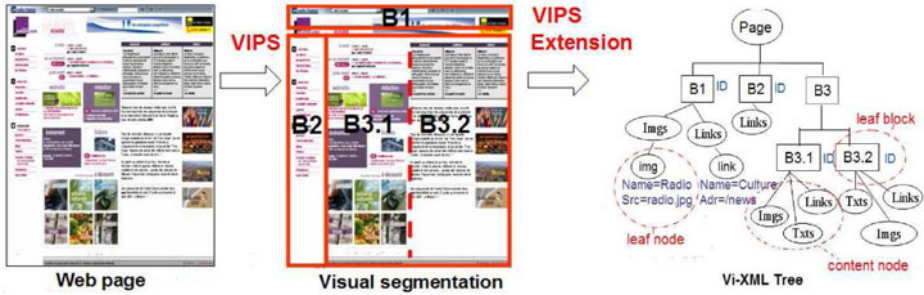
- **Web crawling.** The detected changes between web pages versions can be exploited to optimize web crawling by downloading the most “important” versions [3]. An important version is the version that has important changes since the last archived one. Based on (i) the relative importance of changes operations (insert, delete, etc.) detected in each block and (ii) the weight importance of those blocks [19], the importance of changes between two versions of web pages can be evaluated.
- **Indexing and storage.** Web archive systems can avoid wasting time and space for indexing and/or storing some versions with unimportant changes. An appropriate change importance threshold can be fixed through a supervised machine learning to decide when should pages indexed or stored.
- **Temporal querying.** The proposed change detection can greatly help to provide temporal query capabilities using structural and content changes history. Users can temporally query the archive about the structural changes that affects the visual appearance of web pages. For instance, what type of changes occur in a specific block during the last month...?
- **Visualization and browsing.** Our changes detection algorithm can be used to visualize the changes of web pages over time and navigate between the different archived versions. Several applications [13,18,20] have been designed to visualize changes in the browser by highlighting the content changes on pages such as a text update, delete, etc. However, to the best of our knowledge, no one explicitly view both the structural and content changes occurred on page such as a block insertion, move, update, etc. Our change detection approach can also be exploited to extend web browsers [20] that use the page cache to help users in understanding the changes on the current page since the last visit.
- **Archive maintenance.** Vi-DIFF can also be used after some maintenance operations to verify them in the web archive. The migration from the Internet Archive’s file format ARC to WARC [1] is an example of maintenance operation. After the migration from ARC to WARC, our change detection Vi-DIFF can be exploited to ensure that the page versions are kept “as they are”.

---

<sup>1</sup> The WARC format is a revision of ARC file format that has traditionally been used to store Web crawls url. The WARC better support the harvesting, access, and exchange needs of archiving organizations.

### 3 Vi-DIFF

In this section, we explain in detail the steps of the Vi-DIFF algorithm which detects structural and content changes between two web page versions. It has three main steps:



```
<xml>
  <Page url="www.radiofrance.fr" version="V_01-10-09">
    <Block Ref="B1" ID="001A" Pos="H:10-W:20">
      <Links ID="012C" IDList="042C">
        <link ID="1L23" Name="Culture" Adr="/news">
          </Links>
        <Imgs ID="g15K" IDList="g25K">
          <img ID="25jL" Name="Radio" Src="/radio.jpg"/>
        </Imgs>
      </Block>
    <Block Ref="B2" ID="150K" Pos="H:53-W:10">
      <Links ID="1k2M" IDList="4k2M"> ... </Links>
    </Block>
    <Block Ref="B3" ...>
      <Block Ref="B3.1" ...> ... </Block>
      <Block Ref="B3.2" ...> ... </Block>
    </Block>
  </Page>
</xml>
```

Fig. 1. VIPS Extension

- *Segmentation*: Web pages are visually segmented into semantic blocks by extending the VIPS algorithm.
- *Change Detection*: The structural and content changes between visually segmented web pages are detected.
- *Delta file*: The changes detected are saved in Vi-Delta files. Those three steps are described in detail in the next sections.

#### 3.1 Segmentation

Based on the DOM tree and visual information of the page, VIPS detects horizontal and vertical separators of the page. Then, it constructs the “visual tree” of the page partitioned into multiple blocks. The root is the whole page. Each block

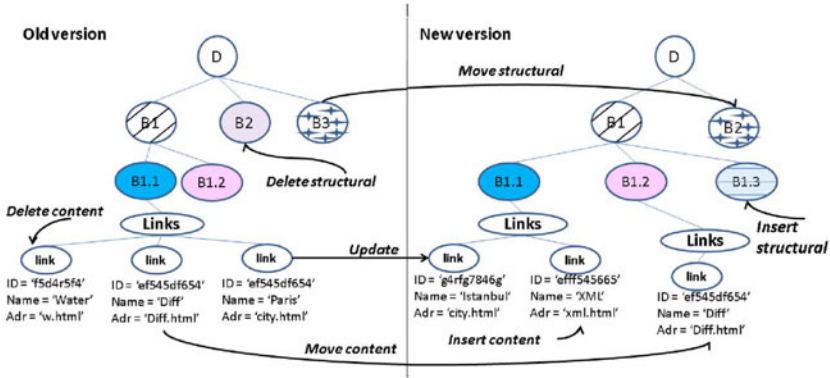


Fig. 2. Operations

is represented as a node in the tree. To complete the visual structure of the page, we extended VIPS to extract links, images and texts for each block as shown in Figure 2.

We define, three types of nodes: a *leaf block* is a block which has zero child block, a *content node* is a child of a *leaf block* like Links, imgs, etc., and a *leaf node* is the node without a child such as img, link. These three types of nodes are uniquely identified by an ID attribute. This ID is a hash value computed using the node’s content and it is children’s node content. Links and Imgs (content nodes) have also an attribute called IDList which has list of all IDs of its leaf nodes. Each block of the page is referenced by the attribute Ref which contains the block’s *Dewey Identifier*. Leaf nodes have other attributes such as the name and the address for links. The complete hierarchical structure of the web page is described in an XML document called Vi-XML. An example of Vi-XML document is shown in Figure 3.

### 3.2 Change Detection

In this section, we give the description of operations which are detected by Vi-DIFF. As mentioned in section 2, we consider two types of changes: content changes and structural changes. Content changes modify the content nodes of blocks. They are detected at leaf nodes. Structural changes are detected at the leaf blocks.

#### • Content change operations

As we mentioned earlier, content changes are realized in the content of blocks for links, images and texts in Vi-XML files through the following operations:

- $Insert(x(name, value), y)$ : Inserts a leaf node  $x$ , with node name and node value, as a child node of content node  $y$ . As shown in Figure 2, in the new version, a new leaf node (Name = ‘XML’) is added to content node Links of leaf block B1.1.

- *Delete(x)*: Deletes a leaf node  $x$ . In Figure 2, in the old version, a leaf node (Name attribute = ‘Water’) is deleted from content node Links of leaf block B1.1.
- *Update(x, attrname, attrvalue)*: Updates the value of *attrname* with *attrvalue* for a given leaf node  $x$ . In Figure 2, a leaf node’s Name attribute ‘Paris’ is updated to ‘Istanbul’ in the new version.
- *Move(x, y, z)*: Moves a leaf node  $x$  from content node  $y$  to content node  $z$ . As shown in Figure 2, a leaf node (Name = ‘Diff’) is moved from content node of the leaf block B1.1 to content node of the leaf block B1.2 in the new version. Most of the existing algorithms treat the move operation as a sequence of delete and insert operations. The use of move operation can have a great impact on the size of the delta script and on its readability. It reduces the file size but increments the complexity of the diff algorithm.

Delete, Update and Insert operations are considered as basic operations and existing algorithms support those operations. Move is only supported by few approaches.

### • Structural changes operations

The structural changes are realized on leaf block nodes in Vi-XML files. Detecting structural changes decreases the size of the delta file and makes it easier to query, to store and to manage. Most of all, it provides a delta much more relevant with respect to what visually happened on the web page.

- *Insert(Bx, y, Bz)*: Inserts a leaf block subtree Bx, which is rooted at  $x$ , to (leaf) block node  $y$ , after (leaf) block node Bz. For example, in Figure 2, a (leaf) block node B1.3 is inserted to block node B1, after leaf block node B1.2.
- *Delete(Bx)*: Deletes a leaf block rooted by  $x$ . In Figure 2, B2 is deleted from the old version.
- *Move(Bx, By)*: Moves sub-tree rooted by  $x$  (Bx) to the node  $y$ . After insert and/or delete operations, the nodes are moved (shifted). As shown in Figure 2, following the structural delete of B2 in old version, B3 is moved/shifted to B2 in new version.

## 3.3 Delta Files

Detected changes are stored in a delta file. We use XML to represent delta files because we need to exchange, store and query deltas by using existing XML tools. The delta files contain all operations (delete, insert, update, move, etc.) that transform one version to another one. The final result after applying delta file to the old version should be exactly the new version. As far as we know there is no standard for delta files.

The content changes are represented by operation tags (delete, insert, update, move) as children of their parent block’s tag. The structural change operations such as delete and insert are added directly to the root of Vi-Delta. For move

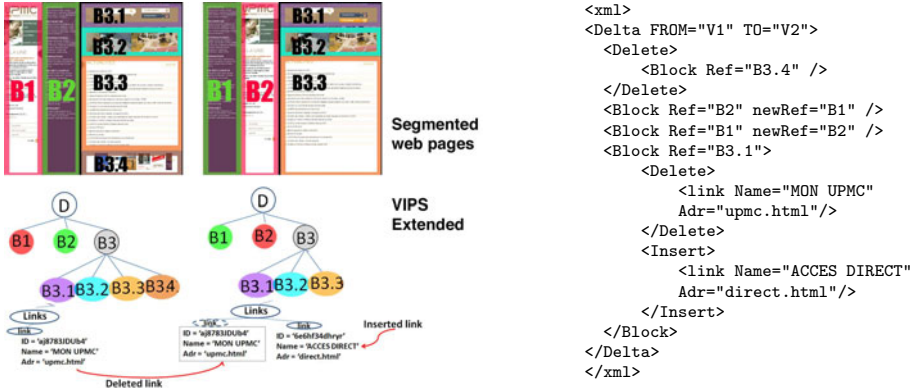


Fig. 3. Vi-Delta

operation in structural changes, a newRef attribute is added to the related block, so that we can keep track of a block even if it moves inside the structure.

Figure 3 shows an example of Vi-Delta. It contains three move and one insert as structural changes operations, one delete and one update as content changes operations. We can see, in this example, how detecting structural changes makes delta file easier to query, to manage and to store. For instance, in our example, instead of deleting and inserting all the content of moved blocks (B1, B2, B3), the structural move operation is used.

## 4 Change Detection

In this section, we give the details of Vi-DIFF’s second step. The algorithm contains a main part and two sub-algorithms: one for detecting structural changes and other one for detecting content changes.

### 4.1 Main Algorithm

The main algorithm is composed of two steps as shown in Figure 4

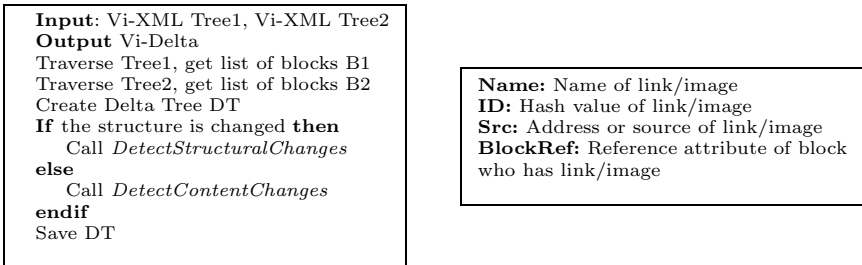


Fig. 4. Main Vi-DIFF - Object Structure

- Step1: Is the structure changed?

We compare the two trees to check if the structure is changed between the two versions. If they have a different number of block nodes, the structure is considered as changed. If they have the same number of blocks we start to compare their Ref in one iteration. As soon as we have a different Ref, the structure is considered as changed.

- Step2: Call change detection algorithms

If the structure is the same, we directly call the content changes detection algorithm. If the structure is changed, the structural changes detection algorithm is called. It calls the content changes algorithm if there is also a content change. Those algorithms are explained in details in the next two sections.

## 4.2 Detecting Content Changes

The content changes are at the level of leaf node, for the children nodes of Links, Imgs and Txts. As Links and Imgs have nearly the same structure, they are treated in the same way. For Txts, we need to find text similarities, thus they are treated separately.

### ◦ *Links and images*

1. Create two arrays (one for each version), create a delta XML file
2. Traverse two trees (XML files) in the same iteration. For each leaf block not matching, check content nodes. If they do not match, create an object (see Figure 4) with each leaf node and add them in its version's array.
3. Sort both arrays by Name attribute (lexical order)
4. Define two counter variables (one for each array)
5. Advance both counters in an endless loop

At each iteration:

- (a) Every time an operation is detected by comparing ID, Name, Src and Ref (see Figure 5 for details), add the operation to delta, remove the corresponding objects from arrays.
- (b) If objects (referenced by counters) in both arrays are different - advance the counter of the first array if it has the smallest Name and its next element has also the smallest Name, else advance the counter of the second array
- (c) Detect operations by comparing objects
- (d) if end of any array: set its counter to 0 which is necessary to compare the rest of the array
- (e) Break when both arrays' sizes are equal to 0

### ◦ *Texts*

We need to find the distance between two texts to decide if it is an update operation or a delete+insert operations. For this, we compute the proportion of distinct words between two versions. If this value is more than 0.5, they are considered as different texts and we have two operations delete then insert. Otherwise, the text is considered as updated. Each text in each block is compared separately.



```

Procedure DetectContentChanges(ASource,AVersion: Array,DT:Delta Tree)
Sort both arrays with merge sort(Name attribute)
while(true)
  if size of ASource and AVersion == 0 break
  if ASource.ID == AVersion.ID then
    if ASource.BlockRef != AVersion.BlockRef then
      MOVE detected
    else if (ASource.Name,ASource.BlockRef) == (AVersion.Name,AVersion.BlockRef)
      && (ASource.Adr/Src != AVersion.Adr/Src) then
        UPDATE detected
    else if (ASource.Adr/Src,ASource.BlockRef) == (AVersion.Adr/Src,AVersion.BlockRef)
      && (ASource.Name != AVersion.Name) then
        UPDATE detected
  else
    If end of any array
      If end of ASource && VSource.size !=0 then
        INSERT detected
      If end of VSource && ASource.size !=0 then
        DELETE detected
    else Increment counter

```

Fig. 5. Detect content change algorithm

### 4.3 Detecting Structural Changes

For detecting structural changes, we use two arrays respectively containing the leaf block nodes of the two compared versions. Two blocks (*e.g.* B1 in version 1 and B2.1 in version 2) are equal if their ID attributes are equal. If IDs are different, we compute the distance between them. To do this, we compare the IDList attribute for Links and Imgs nodes and IDs for Txts.

The distance measure of two blocks is defined on the basis of the distance between their content nodes. The following functions are defined to measure distance. Given two leaf block nodes B1 and B2, we define:

$$Distance(B1, B2) = \frac{Dist('Links', B1, B2) + Dist('Imgs', B1, B2) + DistText(B1, B2)}{3}$$

$$Dist(x, B1, B2) = \frac{\sum x \text{ changed between } B1 \text{ and } B2}{\sum x \text{ in } B1}, \text{ where } x \in \{'Links', 'Imgs'\}$$

$$DistText(B1, B2) = \begin{cases} 0 & \text{if Text ID in } B1 = \text{Text ID in } B2 \\ 1 & \text{otherwise} \end{cases}$$

If Distance(B1,B2) is greater than a fixed value  $\gamma$  (*e.g.*  $\gamma = 0.2$ ), the two blocks are considered as distinct, otherwise, they are considered as similar. If two blocks are similar, we go down in the tree by calling the detecting content change algorithm with the similar block nodes as arguments, here (B1, B2.1). If the two blocks are not similar, the counter of the longest array is incremented if the end of the both arrays is not reached.

Our algorithm has a quadratic complexity. But we should not forget that the asymptotic complexity is only appropriate with large inputs. The web pages are usually rather small and the cases where a page completely changes from one version to another one are very rare. Thus the asymptotic complexity is not relevant here. We decided to measure the actual complexity through experiments as explained in next section.

```

DetectStructuralChanges(Version1, Version2: XML, DT: Delta Tree)
Find and Add leaf blocks(b1,b2) in Version1 and Version2 to arrays A1 and A2
while(true)
  if size of A1 and A2 == 0 break
  If (b1.ID,b1.BlockRef) == (b2.ID,b2.BlockRef) then
    IDEM detected
  else if b1.ID == b2.ID && b1.BlockRef!= b2.BlockRef
    MOVE detected
  else
    if b1 is similar to b2 then
      MOVE detected
      call DetectContentChanges (b1, b2, DT)
    else if end of one of arrays
      if( end of A1) then b2 is inserted
      if( end of A2) then b1 is deleted
    else Increment counter

```

Fig. 6. Detect structural changes algorithm

## 5 Experiments

Experiments are conducted to analyze the performance (execution time) and the quality (correctness) of the Vi-DIFF algorithm.

### 5.1 Segmentation

Visual segmentation experiments have been conducted over HTML web pages by using the extended VIPS method. We measured the time spent by the extended VIPS to segment the page and to generate the Vi-XML document. We present here results obtained over various HTML documents sized from about 20 KB up to 600 KB. These represent only sizes of container objects (CO) of web pages that do not include external objects like images, video, etc.

We measured the performance of the segmentation in terms of execution time and output size. Experiments were conducted on a PC running Microsoft Windows Server 2003 over a 3.19 GHz Intel Pentium 4 processor with 6.0 GB of

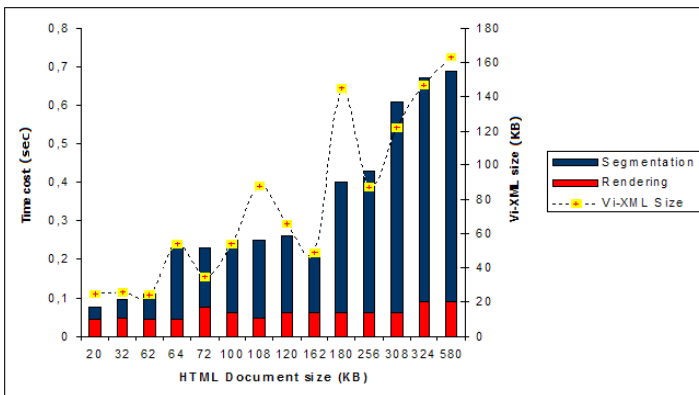


Fig. 7. Segmentation Time

RAM. The execution time for the browser rendering and the visual segmentation is shown in Figure 7. The horizontal axis represents the size of HTML documents in KBytes (KB). The left vertical axis shows the execution time in seconds for each document. The time for rendering is almost constant, about 0.1 seconds. The execution time of the visual segmentation increases according to the size of HTML documents. The time of the segmentation is about 0.2 seconds for documents sized about 150KB which represents the average size of CO in the web. This execution time seems to be a little bit costly but is counterbalanced by the expressiveness of the Vi-XML file that really simulates the visual aspect of web pages. Nevertheless, this time cost must be optimized. The main idea for that purpose is to avoid rebuilding the blocks structure for a page version if no structural change has occurred since the former version. We are currently trying to find a method that detects directly changes inside blocks based on the visual structure of previous versions of the document.

The right vertical axis in Figure 7 shows the the size of the output Vi-XML file with respect to the size of the original HTML document. From this experiment, we can observe that the Vi-XML document size is usually about 30 to 50 percent less than the size of the original HTML document (for those sized more than 100 KB). This is interesting for the comparison of two Vi-XML documents since it can help to reduce the time cost of changes detection algorithm.

## 5.2 Change Detection

We have two types of test in this section: one without structural changes and another with structural changes. For this section, tests are realized on PC running Linux over a 3.33GHz Intel(R) Core(TM)2 Duo CPU E8600 with 8 GB of RAM.

**Without structural changes.** To analyze the performance and the quality of the second step of Vi-DIFF algorithm, we built a simulator that generates synthesized changes on any Vi-XML document. The simulator takes a Vi-XML file and it generates a new one according to the change parameters given as input. It also generates a delta file, called Vi-Sim-Delta. The two Vi-XML files are compared with the Vi-DIFF algorithm and a Vi-Delta is generated. To check the correctness of change detection step, we compared all Vi-Delta et Vi-Sim-Delta files with DeltaXML trial version [15]. Vi-Delta and Vi-Sim-Delta files are always identical which shows the correctness of this step of Vi-DIFF. For this test, different versions are obtained from a crawled web page (<http://www.france24.com/fr/monde>) by using our simulator. For each type of operation (delete, insert, update, move) the change rate is increased by 10% until reaching 100%. It means that the last version with 100% change rate is completely different from the original one. As links and images are treated in the same way, we observe that the execution time is the same on average. To simplify the figure's readability, we only give the results for links. As the change detection part is measured in milliseconds, processor events may cause noise on the test results. Thus, all the process is executed for 10000 times and the results obtained are normalized (average). The results are given in Figure 8.

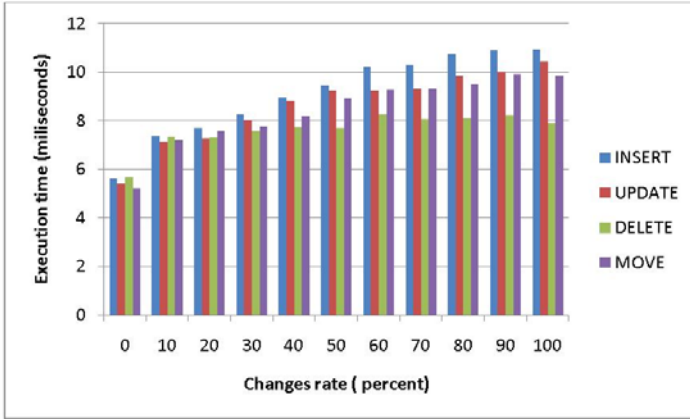


Fig. 8. Change Detection Execution Time

As we can see, move and update operations have nearly the same execution time because they do not change the size of Vi-XML files, but insert and delete operations change the size. The execution time increases along with the insert operation rate (more objects to compare), and decreases along with the delete operation rate (less objects to compare). Also a profiler analyzer is used to see the details of execution time. 70% of the time is used to parse Vi-XMLs and to save Vi-Delta, 10% is used to detect if there is a structural changes, 20% is used to detect changes.

**With structural changes.** In order to test the structural changes, the versions with the structural changes are obtained by modifying the segmentation rate (degree of coherence) in the VIPS algorithm. It can be seen as “merge and split” of blocks which change the Vi-XML’s structure. It also allows us to test blocks’ similarity by using hourly crawled web pages. Average file size of those pages is 80 Kb. To give an idea about initial environment, we first compare two identical Vi-XML files. The execution time is 7,3 ms. Then, two versions hourly crawled (<http://www.cnn.com>) are segmented with different segmentation rate into two Vi-XML files. These files are compared. The execution time is 9.5 ms and the generated Vi-Delta size is 1.8 Kb. Vi-DIFF algorithm is modified to not consider structural changes and the same Vi-XML files are compared. The execution time is 48 ms and the generated Vi-Delta size is 40 Kb. Because with structural change detection, there is no need to go down until leaf nodes for all blocks except similar blocks so the change detection is faster. Also, a smaller delta file is easier to store, to read and to query. As the simulator is not able to make structural changes for the moment, the correctness of delta file for this section is checked manually.

According to our experiments, Vi-DIFF is able to detect basic operations and move (without structural changes) correctly between two web page versions. The total execution is between 0,1 seconds and 0.8 seconds depending on the page

versions' size. The execution time for change detection step is satisfying as it allows to process about one hundred pages per second and per processor. Thus, we are working on reducing the segmentation time.

## 6 Conclusion and Future Works

We propose Vi-DIFF, a change detection method for web pages versions. Vi-DIFF helps to understand what happened and changed between two page versions. It detects semantic differences between two web pages based on their visual representation. Preserving the visual structure of web pages while detecting changes gives relevant information to understand modifications which is useful for many applications dealing with web pages. Vi-DIFF consists of three steps: (i) segmentation, (ii) change detection and (iii) generation of delta file. It detects two types of changes; structural and content changes. Structural changes (insert, delete, move) alter the block structure of the page and its visual appearance. Content changes (insert, delete, update) modify links, images, texts. In addition to basic operations, it supports a move operation, if there is no structural changes. However, it does not support yet the move on content changes when there are structural changes. This should be handled by future versions.

The proposed Vi-DIFF algorithm generates, as output, a Vi-Delta file which describes change operations occurred between the two versions. The structure of the produced Vi-Delta helps to visually analyze the changes between versions. It can serve for various applications: web crawl optimization, archive maintenance, historical changes browsing, etc. A simulator was developed to test the performance of the algorithm. Experiments in terms of execution time were conducted for each step. The execution time for change detection step is very promising, but the execution time cost for segmentation step must be optimized.

Another on-going future work is to detect splitting and merging of blocks as structural changes. The simulator also needs new features like generating new versions with structural changes. Also we want to exploit the produced Vi-Delta to analyze/evaluate the importance of changes between versions. Further work will be done to efficiently query archived Vi-XML versions and the visual/structural changes occurred between them as described in the Vi-Delta.

## References

1. The Web archive bibliography,  
<http://www.ifs.tuwien.ac.at/~aola/links/WebArchiving.html>
2. Abiteboul, S., Cobena, G., Masanes, J., Sedrati, G.: A First Experience in Archiving the French Web. In: Agosti, M., Thanos, C. (eds.) ECDL 2002. LNCS, vol. 2458, p. 1. Springer, Heidelberg (2002)
3. Ben-Saad, M., Gançarski, S., Pehlivan, Z.: A Novel Web Archiving Approach based on Visual Pages Analysis. In: 9th International Web Archiving Workshop (IWA'09), Corfu, Greece (2009)
4. Blakeman, K.: Tracking changes to web page content,  
<http://www.rba.co.uk/sources/monitor.htm>

5. Lampos, D.J.C., Eirinaki, M., Vazirgiannis, M.: Archiving the greek web. In: 4th International Web Archiving Workshop (IWA'04), Bath, UK (2004)
6. Cai, D., Yu, S., Wen, J.-R., Ma, W.-Y.: VIPS: a Vision-based Page Segmentation Algorithm. Technical report, Microsoft Research (2003)
7. Cathro, W.: Development of a digital services architecture at the national library of Australia. EduCause (2003)
8. Cobena, G., Abiteboul, S., Marian, A.: Detecting changes in XML documents. In: ICDE '02: Proceedings of 18th International Conference on Data Engineering (2002)
9. Cosulschi, M., Constantinescu, N., Gabrovanu, M.: Classification and comparison of information structures from a web page. In: The Annals of the University of Craiova (2004)
10. Evi, M.K., Diligenti, M., Gori, M., Maggini, M., Milutinovi, V.: Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification. In: The Proceedings of 2002 IEEE International Conference on Data Mining ICDM'02 (2002)
11. Gomes, D., Santos, A.L., Silva, M.J.: Managing duplicates in a web archive. In: SAC '06: Proceedings of the 2006 ACM Symposium on Applied Computing (2006)
12. Gu, X.-D., Chen, J., Ma, W.-Y., Chen, G.-L.: Visual Based Content Understanding towards Web Adaptation. In: De Bra, P., Brusilovsky, P., Conejo, R. (eds.) AH 2002. LNCS, vol. 2347, p. 164. Springer, Heidelberg (2002)
13. Jatowt, A., Kawai, Y., Nakamura, S., Kidawara, Y., Tanaka, K.: A browser for browsing the past web. In: Proceedings of the 15th International Conference on World Wide Web, WWW '06, pp. 877–878. ACM, New York (2006)
14. Kukulenz, D., Reinke, C., Hoeller, N.: Web contents tracking by learning of page grammars. In: ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services, Washington, DC, USA, pp. 416–425. IEEE Computer Society, Los Alamitos (2008)
15. La-Fontaine, R.: A Delta Format for XML: Identifying Changes in XML Files and Representing the Changes in XML. In: XML Europe (2001)
16. Leonardi, E., Hoai, T.T., Bhowmick, S.S., Madria, S.: DTD-Diff: A change detection algorithm for DTDs. *Data Knowl. Eng.* 61(2) (2007)
17. Lindholm, T., Kangasharju, J., Tarkoma, S.: Fast and simple XML tree differencing by sequence alignment. In: DocEng '06: Proceedings of the 2006 ACM Symposium on Document Engineering (2006)
18. Liu, L., Pu, C., Tang, W.: Webcq - detecting and delivering information changes on the web. In: Proc. Int. Conf. on Information and Knowledge Management (CIKM), pp. 512–519. ACM Press, New York (2000)
19. Song, R., Liu, H., Wen, J.-R., Ma, W.-Y.: Learning block importance models for web pages. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web (2004)
20. Teevan, J., Dumais, S.T., Liebling, D.J., Hughes, R.L.: Changing how people view changes on the web. In: UIST '09: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, pp. 237–246. ACM, New York (2009)
21. Wang, Y., DeWitt, D., Cai, J.-Y.: X-Diff: an effective change detection algorithm for XML documents. In: ICDE '03: Proceedings of 19th International Conference on Data Engineering (March 2003)

# Automatic Fuzzy Semantic Web Ontology Learning from Fuzzy Object-Oriented Database Model

Fu Zhang, Z.M. Ma, Gaofeng Fan, and Xing Wang

College of Information Science & Engineering, Northeastern University, Shenyang,  
110819, China  
mazongmin@ise.neu.edu.cn

**Abstract.** How to construct Web ontologies that meet applications' needs has become a key technology to enable the Semantic Web. Manual development of ontologies remains a cumbersome and time-consuming task. In real-world applications, however, information is often vague or ambiguous. Thus, developing approaches and tools for constructing fuzzy ontologies by extracting domain knowledge from huge amounts of existing fuzzy databases can facilitate fuzzy ontology development. In this paper, we propose a formal approach and an automated tool for constructing fuzzy ontologies from fuzzy Object-Oriented database (FOOD) models. Firstly, we introduce the fuzzy ontology, which consists of the fuzzy ontology structure and instances. Then, the FOOD models are investigated, and we propose a kind of formal definition of FOOD models. On this basis, we develop a formal approach that can translate the FOOD model and its corresponding database instances into the fuzzy ontology structure and the fuzzy ontology instances, respectively. Furthermore, following the proposed approach, we implement an automated learning tool, which can automatically construct fuzzy ontologies from FOOD models. Case studies show that the approach is feasible and the automated learning tool is efficient.

**Keywords:** Fuzzy database; Fuzzy Object-Oriented database (FOOD) model; Fuzzy ontology; Ontology learning; Automated learning tool.

## 1 Introduction

How to construct Web ontologies that meet applications' needs has become a key technology to enable the Semantic Web [2]. Ontology can be generated from various data resources such as textual data, dictionary, knowledge-based, semi-structured schemata, and database models [15]. Compared to the other types of data resources, ontology construction from database models has increasingly attracted the most attention because most formatted data on the Web are still stored in databases and are not published as an open Web of inter-referring resources [2], [15].

On this basis, constructing Web ontologies by extracting domain knowledge from database models has been extensively investigated. The mappings from relational models to ontologies were established in [12], [23], [29]. The approach for translating ER models into ontologies was developed in [27]. The ontology development tools

[15] such as Protégé-2000 and OntoEdit can be used to build or reuse ontologies. For a comprehensive review of constructing Web ontologies, ones can refer to [15].

However, information is often vague or ambiguous. Thus, the problems that emerge are how to represent these uncertain data within the ontology definitions and database models. To fill this gap, the fuzziness has been extensively introduced into ontologies [4], [5], [14], [20], [21], [25], [26]. Also, many researches have already been made to represent and manipulate fuzzy data in various database models such as relational database models [11], [14], Object-Oriented database models [7], [8], [9], [18]. For the comprehensive reviews about fuzzy ontologies and fuzzy database models, please refer to [20], [13], respectively.

In recent years, how to construct fuzzy ontologies from fuzzy database models has increasingly received the attention. In [21], Quan proposed a fuzzy ontology framework (FOGA) that can generate a fuzzy ontology from uncertainty data based on Formal Concept Analysis (FCA) theory. Zhang [30] developed an approach for constructing fuzzy ontologies from fuzzy ER models. Ma [14] presented a fuzzy ontology generation framework from fuzzy relational databases. Zhang [29] investigated how to construct fuzzy ontologies from fuzzy UML models. And the automated learning tools were missed in [14], [29], [30]. Blanco [4] realized the translation from fuzzy relational databases to fuzzy ontologies.

In particular, it has been found that the classical database models (such as ER model, relational model, and UML) and their extensions of fuzziness do not satisfy the need of handling complex objects with imprecision and uncertainty [7], [11]. Fuzzy Object-Oriented database (FOOD) models are hereby developed. The FOOD model, which can model uncertain data and complex-valued attributes as well as complex relationships among objects, has been extensively applied in the data and knowledge intensive applications such as CAD/CAM, office automation systems, and so on [7], [8], [9], [11], [13], [18]. Currently, many data sources are modeled in fuzzy Object-Oriented databases, and abundant domain knowledge is contained in FOOD models [9]. Hence, it is possible and meaningful to use the FOOD models as the base knowledge for domain ontology constructing, which will facilitate the development of fuzzy ontology.

Moreover, as mentioned in [23], to support several Semantic Web applications (e.g., Semantic Web sites), the existing Web data and documents must be “upgraded” to Semantic Web content that is semantically annotated with Web ontologies. It is well known that the Web-accessible databases are the main content sources on the current Web [27]. However, a precondition here is that ones have a Web ontology at hand that can capture the domain knowledge of the database. As lots of fuzzy databases today are modeled in FOOD models (see [7], [9], [18], etc.), it is necessary to develop approaches and tools for constructing fuzzy ontologies from FOOD models, which can also act as a gap-bridge between lots of existing database applications and the Semantic Web.

To our best knowledge, there are no reports on fuzzy ontology construction from FOOD models. In this paper, we develop a formal approach and an automated tool for constructing fuzzy ontologies from FOOD models. The paper includes the details:

- How to represent the constructed fuzzy ontology? Since the paper aims at constructing fuzzy ontologies from FOOD models, it is necessary for us to give the formal definition of target fuzzy ontologies. In **Section 3**, we report the fuzzy ontology definition presented in [30].



- How to formalize the FOOD model? In **Section 4**, FOOD models are studied, and we propose a kind of formal definition of FOOD models.
- How to construct fuzzy ontologies from FOOD models? The answer is (see **Section 5**): (i) translating the FOOD model into the fuzzy ontology structure at conceptual level; (ii) translating the database instances w.r.t. the FOOD model into the fuzzy ontology instances at instance level; (iii) proving that the translations are “semantics-preserving” and giving the translation examples.
- How to develop the automated learning tool? Following the proposed approach, in **Section 6**, we implement an automated learning tool called *FOOD2FOWL*, which can automatically construct fuzzy ontologies from FOOD models. Case studies show that the proposed approach is feasible and the tool is efficient.

The remainder of this paper is organized as follows. Section II introduces related work. Section III introduces the fuzzy ontology. Section IV proposes the formal definition of FOOD models. Section V proposes the learning approach. Section VI implements the automated learning tool. Section VII shows conclusions.

## 2 Related Work

The importance of ontologies to the Semantic Web has prompted the development of kinds of methods and tools to help people construct ontologies. Besides the approaches introduced in Section 1, the following researches are related to our work.

There are several approaches for establishing the relationships between Description Logics and Object-Oriented database models. The approaches presented in [3], [6], [22] investigated how to reason on the Object-Oriented database model by translating it into a Description Logic knowledge base, whereas our work aims at constructing fuzzy ontologies.

Moreover, several works investigated the relationships between Object-Oriented languages and ontologies, but they did not focus on constructing ontologies from Object-Oriented database models. Meditskos [16] used the Object-Oriented language COOL to model and handle ontology concepts and RDF resources. Mota [17] developed an Object-Oriented framework for representing ontologies. Oren [19] presented an Object-Oriented API for managing RDF data. Koide [10] demonstrated the possibility of the integration of OWL and Object-Oriented Programming.

As mentioned in Section 1, there have been several approaches for constructing fuzzy ontologies from fuzzy database models. However, considering that the classical database models (e.g., ER model, relational model, and UML) and their extensions of fuzziness do not satisfy the need of modeling complex objects with imprecision and uncertainty, and many data sources are modeled in FOOD models in real-world applications [7], [8], [9], [11], [13], [18], thus the paper aims at investigating how to construct fuzzy ontologies from FOOD models.

In addition, the fuzzy ontology instances presented in [14], [29] were represented by the fuzzy RDF models, in this paper, as with the representation form of the fuzzy ontology structure, the fuzzy ontology instances were also represented by the form of axioms, which are facilitate access to and evaluation of the fuzzy ontologies [5], [26].

### 3 Fuzzy OWL DL Ontology

A fuzzy ontology formulated in fuzzy OWL DL language is called fuzzy OWL DL ontology [30]. In this section, we first introduce the fuzzy OWL DL language based on the works [24], [25], [26], [30], and then report the fuzzy OWL DL ontology definition presented in [30].

#### 3.1 Fuzzy OWL DL Language

OWL has three increasingly expressive sublanguages OWL Lite, OWL DL, and OWL Full [26]. OWL DL is the language chosen by the major ontology editors because it supports those users who want the maximum expressiveness without losing computational completeness and decidability of reasoning systems [26]. In addition, OWL DL has two types of syntactic form [27]: the exchange syntax, i.e., the RDF/XML syntax, and the frame-like style abstract syntax. The abstract syntax facilitates access to and evaluation of the ontologies [5], [26], [27]. However, OWL DL cannot handle the information represented with a not precise definition.

The fuzzy OWL DL [5], [26] is the fuzzy extension of OWL DL, which can be approximately viewed as the expressive fuzzy Description Logic f-SHOIN(D) [24]. Based on [26], Table 1 gives the *fuzzy OWL DL abstract syntax*, *Description Logic syntax*, and *semantics*. Here, parts of OWL constructors are omitted in Table 1.

The semantics for fuzzy OWL DL is based on the interpretation of f-SHOIN(D) (details refer to [24]). The interpretation is given by a pair  $\langle \Delta^{\text{FI}}, \bullet^{\text{FI}} \rangle$  where  $\Delta^{\text{FI}}$  is the individual domain,  $\bullet^{\text{FI}}$  is a fuzzy interpretation function. In Table 1,  $\Delta_D^{\text{FI}}$  is the data-value domain,  $\forall d^{\text{FI}}, c^{\text{FI}}, o^{\text{FI}} \in \Delta^{\text{FI}}, v^{\text{FI}} \in \Delta_D^{\text{FI}}$ ,  $C$  denotes class description,  $D$  denotes fuzzy data range, fuzzy ObjectProperty and DatatypeProperty identifiers are denoted by  $R$  and  $U$ , respectively,  $\#S$  denotes the cardinality of a set  $S$ ,  $\bowtie \in \{\geq, >, \leq, <\}$ .

#### 3.2 Fuzzy OWL DL Ontology

A fuzzy OWL DL ontology is a couple  $FO = (FO_S, FO_I)$  where  $FO_S$  is the fuzzy ontology structure and  $FO_I$  is the fuzzy ontology instances associated with the fuzzy ontology structure. The following Definition 1 gives the formal definition of fuzzy OWL DL ontology [30], and we further add fuzzy individual axioms to the definition.

**Definition 1 (Fuzzy OWL DL Ontology).** A fuzzy OWL DL ontology is a couple  $FO = (FO_S, FO_I) = (FID_0, FAxiom_0)$ , where:

(1)  $FID_0 = FCID_0 \cup FIID_0 \cup FDRID_0 \cup FOPID_0 \cup FDPID_0$  is a fuzzy OWL DL identifier set (see Table 1) partitioned into:

- a subset  $FCID_0$  of fuzzy class identifiers, include user-defined identifiers plus two predefined fuzzy classes owl: Thing and owl: Nothing.
- a subset  $FIID_0$  of individual identifiers.
- a subset  $FDRID_0$  of fuzzy data range identifiers; each fuzzy data range identifier is a predefined XML Schema fuzzy datatype [13], [25].

- a subset  $FOPID_0$  of fuzzy object property identifiers.
  - a subset  $FDPID_0$  of fuzzy datatype property identifiers.
- (2)  $FAxiom_0$  is a fuzzy OWL DL axiom set (see Table 1) partitioned into:
- a subset of fuzzy class/property axioms.
  - a subset of fuzzy individual axioms.

From a semantics point of view, a fuzzy OWL DL ontology  $FO$  is a set of fuzzy OWL DL axioms in Table 1. An interpretation  $FI$  is a model of  $FO$  iff it satisfies all axioms in  $FO$ . Furthermore, an  $FO$  is satisfiable iff there is a model of it.

**Table 1.** Fuzzy OWL DL Abstract Syntax, Description Logic (DL) Syntax, and Semantics

Fuzzy OWL DL Syntax	DL Syntax	Model-Theoretic Semantics
<b>Fuzzy class description (C)</b>		
A is a <i>URIRef</i> of a fuzzy class owl:Thing owl:Nothing unionOf ( $C_1 \dots C_n$ )	A T $\perp$ $C_1 \cup \dots \cup C_n$	$A^{FI}: \Delta^{FI} \rightarrow [0, 1]$ $T^{FI}(d) = 1$ $\perp^{FI}(d) = 0$ $(C_1 \cup \dots \cup C_n)^{FI}(d) = \max\{C_1^{FI}(d), \dots, C_n^{FI}(d)\}$
restriction (R allValuesFrom(C))	$\forall R.C$	$(\forall R.C)^{FI}(d) = \inf_{d' \in \Delta^{FI}} \{\max\{1 - R^{FI}(d, d'), C^{FI}(d')\}\}$
restriction (R minCardinality(n))	$\geq n R$	$(\geq nR)^{FI}(d) = \sup_{c_1, \dots, c_n \in \Delta^{FI}} \bigwedge_{i=1}^n R^{FI}(d, c_i)$
restriction (R maxCardinality(n))	$\leq n R$	$(\leq nR)^{FI}(d) = \inf_{c_1, \dots, c_{n+1} \in \Delta^{FI}} \bigvee_{i=1}^{n+1} (1 - R^{FI}(d, c_i))$
restriction (R cardinality(n))	$= n R$	$(= nR)^{FI}(d) = (\geq nR \cap \leq nR)^{FI}(d)$
restriction (U allValuesFrom(D))	$\forall U.D$	$(\forall UD)^{FI}(d) = \inf_{v \in \Delta_D^{FI}} \{\max\{1 - U^{FI}(d, v), D^{FI}(v)\}\}$
<b>Fuzzy class axioms</b>		
Class ( A partial $C_1 \dots C_n$ ) SubClassOf ( $C_1 C_2$ ) EquivalentClasses ( $C_1 \dots C_n$ ) DisjointClasses ( $C_1 \dots C_n$ )	$A \subseteq C_1 \cap \dots \cap C_n$ $C_1 \subseteq C_2$ $C_1 \equiv \dots \equiv C_n$ $C_i \neq C_j$	$A^{FI}(d) \leq \min\{C_1^{FI}(d), \dots, C_n^{FI}(d)\}$ $C_1^{FI}(d) \leq C_2^{FI}(d)$ $C_1^{FI}(d) = \dots = C_n^{FI}(d)$ $\min\{C_i^{FI}(d), C_j^{FI}(d)\} = 0 \quad 1 \leq i < j \leq n$
<b>Fuzzy property axioms</b>		
DatatypeProperty ( U domain( $C_1$ )...domain( $C_m$ ) range( $D_1$ )...range( $D_k$ ) [Functional] ) ObjectProperty ( R domain( $C_1$ )...domain( $C_m$ ) range( $C_1$ )... range( $C_k$ ) [inverseOf ( $R_0$ ) ] )	$\geq 1U \subseteq C_i$ $T \subseteq \forall U.D_i$ $T \subseteq \leq 1U$ $\geq 1R \subseteq C_i$ $T \subseteq \forall R.C_i$ $R = (R_0)^{\neg}$	$U^{FI}(d, v) \leq C_i^{FI}(d) \quad i = 1, \dots, m$ $U^{FI}(d, v) \leq D_i^{FI}(v) \quad i = 1, \dots, k$ $\forall d \in \Delta^{FI} \#\{v \in \Delta_D^{FI} : U^{FI}(d, v) \geq 0\} \leq 1$ $R^{FI}(d_1, d_2) \leq C_i^{FI}(d_1) \quad i = 1, \dots, m$ $R^{FI}(d_1, d_2) \leq C_i^{FI}(d_2) \quad i = 1, \dots, k$ $R^{FI}(d_1, d_2) = R_0^{FI}(d_2, d_1)$
<b>Fuzzy individual axioms</b>		
Individual ( $o$ type( $C_i$ ) [ $\bowtie m_i$ ]... value( $R_i, o_i$ ) [ $\bowtie k_i$ ]... value( $U_i, v_i$ ) [ $\bowtie l_i$ ]... ) SameIndividual ( $o_1 \dots o_n$ ) DifferentIndividuals ( $o_1 \dots o_n$ )	$o : C_i \bowtie m_i$ $(o, o_i):R_i \bowtie k_i$ $(o, v_i):U_i \bowtie l_i$ $o_1 = \dots = o_n$ $o_i \neq o_j$	$C_i^{FI}(o) \bowtie m_i, \quad m_i \in [0, 1], \quad 1 \leq i \leq n$ $R_i^{FI}(o, o_i) \bowtie k_i, \quad k_i \in [0, 1], \quad 1 \leq i \leq n$ $U_i^{FI}(o, v_i) \bowtie l_i, \quad l_i \in [0, 1], \quad 1 \leq i \leq n$ $o_1^{FI} = \dots = o_n^{FI}$ $o_i^{FI} \neq o_j^{FI} \quad 1 \leq i < j \leq n$

## 4 The Fuzzy Object-Oriented Database (FOOD) Model

The fuzzy Object-Oriented database (FOOD) model is the fuzzy extension of traditional Object-Oriented database model (OODM) [3], [6], i.e., some major notions in OODMs such as objects, classes, object-class relationships, and inheritance relationship are extended under fuzzy information environment. In addition, similarly for the references [1], [3], [6], [8], [9], [13], we restrict our attention to the structural components of FOOD models.

The FOOD models are based on the notions of fuzzy objects, fuzzy classes, fuzzy attributes, and fuzzy inheritances [9]. In the following, we propose a kind of formal definition of FOOD models, which is the fuzzy extension of OODM definition in [6].

**Definition 2 (FOOD Model).** A FOOD model is a finite set of class declarations, which is a tuple  $FS = (FC_{FS}, FA_{FS}, FD_{FS})$ , where:

- $FC_{FS}$  is a finite set of fuzzy class names, denoted by the letter FC;
- $FA_{FS}$  is a finite set of fuzzy attribute names, denoted by the letter FA;
- $FD_{FS}$  is a finite set of fuzzy class declarations. For each fuzzy class  $FC \in FC_{FS}$ ,  $FD_{FS}$  contains exactly one such declaration:

Class FC *is-a*  $FC_1, \dots, FC_n$  *type-is* FT,

where *is-a* denotes the inheritance relationship between classes, *type-is* specifies the structure of class FC through a *type expressive* FT, which is built according to the following syntax:

FT  $\rightarrow$  FC |  
 Union FT<sub>1</sub>, ..., FT<sub>k</sub> End |  
 Set-of FT |  
 Record FA<sub>1</sub>:FT<sub>1</sub>, ..., FA<sub>k</sub>:FT<sub>k</sub> End.

Fig. 1 shows a simple *FOOD model*  $FS_1$  modeling part of the reality at a company.

```

Class Computer type-is
  Union HP-Computer, Dell-Computer
  End
Class Old-Computer is-a Computer type-is
  Record
    ComID: String
    Brand: String
    UseYear: Integer
  End
Class Young-Employee is-a Employee type-is
  Record
    EmpID: String
    Name: String
    FUZZY Age: Integer
    FUZZY Use: Set-of Old-Computer
  End
  
```

**Fig. 1.** A FOOD model  $FS_1$  modeling part of the reality at a company

The detailed instructions about Fig. 1 are as follows:

- (1) *Computer* is a generalization of *HP-Computer* and *Dell-Computer*.
- (2) Inverse of attributes: in order to distinguish the *subject* and the *object* of attribute *Use*, the roles *Useby* and *Useof* are introduced (see Fig. 2). Therefore, the inverse of attribute *Use* is introduced, which will be discussed in Section 5.1 in detail.

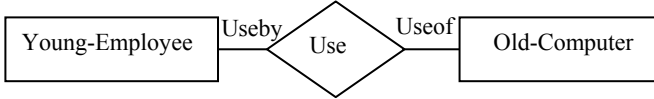


Fig. 2. The graphical representation of the inverse of attribute Use

- (3) The attributes  $FA_i$  (such as *Name*, *Age* in Fig. 1), called *datatype attributes*, which associate with the basic domains such as Integer, Real, etc.
- (4) The attributes  $FA$  (such as *Use* in Fig. 1), called *object attributes*, which denote the relationship between two participant classes.

The semantics of a FOOD model can be given by the fuzzy database states  $FJ$ , i.e., the fuzzy database instances. Fig. 3 shows the *database instances* w.r.t. the FOOD model  $FS_1$  (only part of data):

- (1)  $u = 0.9$  denotes that the possibility which an object  $O_1$  belongs to the class *Young-Employee* is 0.9.
- (2) The value of attribute *Use* is a possibility distribution  $\{0.8/O_2, 0.7/O_3\}$ , where  $0.8/O_2$  denotes that the possibility which object  $O_1$  uses object  $O_2$  is 0.8.
- (3) The values of the other fuzzy attributes (e.g., *Age*) are represented by the possibility distributions [31], [32].

<i>Young-Employee</i>				
<i>EmpID</i>	<i>Name</i>	<i>Fuzzy Age</i>	<i>Fuzzy Use</i>	<i>u</i>
$O_1$	Chris	{ 0.8/27, 0.9/30 }	{ 0.8/ $O_2$ , 0.7/ $O_3$ }	0.9

<i>Old-Computer</i>				
<i>ComID</i>	<i>Brand</i>	<i>UseYear</i>	<i>u</i>	
$O_2$	Dell	4	0.85	

<i>Old-Computer</i>				
<i>ComID</i>	<i>Brand</i>	<i>UseYear</i>	<i>u</i>	
$O_3$	Hp	3	0.8	

Fig. 3. The database instances w.r.t. the FOOD model  $FS_1$  in Fig. 1

## 5 Fuzzy OWL DL Ontology Learning from FOOD Model

This section proposes a formal approach for constructing fuzzy OWL DL ontologies from FOOD models, including: (i) translating the FOOD model into the fuzzy ontology structure (see Definition 3). (ii) translating the database instances w.r.t. the FOOD model into the fuzzy ontology instances (see Definition 4). (iii) proving that the translations are “semantics-preserving” and giving the translation examples.

### 5.1 Translating FOOD Model into Fuzzy OWL DL Ontology Structure

The Definition 3 gives a formal approach for translating the FOOD model into the fuzzy OWL DL ontology structure. Starting with the construction of the *atomic identifiers*  $FID_0$ , the approach induces a set of *fuzzy class/property axioms*  $FAxiom_0$  from the FOOD model.

**Definition 3 (Structure Translation).** Given a FOOD model  $FS = (FC_{FS}, FA_{FS}, FD_{FS})$  in Definition 2. The fuzzy OWL DL ontology structure  $FO_S = \varphi(FS) = (FID_0, FAxiom_0)$  can be derived by function  $\varphi$  as shown in Table 2.

**Table 2.** Translating rules from a FOOD model to a fuzzy OWL DL ontology structure

FOOD model $FS = (FC_{FS}, FA_{FS}, FD_{FS})$	Fuzzy ontology structure $FO_S = \varphi(FS) = (FID_0, FAxiom_0)$
<b>fuzzy class <math>FC_{FS}</math>, fuzzy attribute <math>FA_{FS}</math></b>	<b>identifier set <math>FID_0</math></b>
Each fuzzy class symbol FC	A fuzzy class identifier $\varphi(FC) \in FCID_0$
Each type expression <i>Record FA : Set-of FT End</i>	
Each <i>object attribute</i> symbol FA	A fuzzy class identifier $\varphi(FA) \in FCID_0$
Each <i>object attribute</i> symbol FA	Add two additional roles $FU_1$ and $FU_2$ ; $FU_1$ and $FU_2$ are mapped into two fuzzy object property identifiers $\varphi(FU_1)$ and $\varphi(FU_2)$ : $\varphi(FU_1) \in FOPID_0$ $\varphi(FU_2) \in FOPID_0$ ; In addition, we use symbols $FV_1$ and $FV_2$ denote the inverse properties of $\varphi(FU_1)$ and $\varphi(FU_2)$ , respectively: $FV_1 = \text{invof\_}\varphi(FU_1) \in FOPID_0$ $FV_2 = \text{invof\_}\varphi(FU_2) \in FOPID_0$
Each type expression <i>Record FA<sub>1</sub>:FD<sub>1</sub>, ..., FA<sub>k</sub>:FD<sub>k</sub> End</i>	
Each <i>datatype attribute</i> symbol FA <sub>i</sub>	A fuzzy datatype property identifier $\varphi(FA_i) \in FDPID_0$
Each domain symbol FD <sub>i</sub>	A fuzzy XML Schema datatype identifier $\varphi(FD_i) \in FDRID_0$
<b>fuzzy class declarations <math>FD_{FS}</math></b>	<b>fuzzy class/property axiom set <math>FAxiom_0</math></b>
Each fuzzy class declaration: Class FC <i>is-a</i> FC <sub>1</sub> , ..., FC <sub>n</sub>	Create a fuzzy class axiom: Class ( $\varphi(FC)$ partial $\varphi(FC_1)$ , ..., $\varphi(FC_n)$ ).

**Tabel 2.** (continued)

Each fuzzy class declaration: Class FC <i>type-is</i> Union FT <sub>1</sub> , ..., FT <sub>k</sub> <i>End</i>	Create fuzzy class axioms: Class ( $\varphi(\text{FC})$ partial unionOf( $\varphi(\text{FT}_1), \dots, \varphi(\text{FT}_k)$ ) ); SubClassOf ( $\varphi(\text{FT}_i)$ $\varphi(\text{FC})$ ).
Each fuzzy class declaration: Class FC <sub>1</sub> <i>type-is</i> Record FA: Set-of FC <sub>2</sub> <i>End</i>	Create a fuzzy class axiom: Class ( $\varphi(\text{FA})$ partial restriction ( $\varphi(\text{FU}_1)$ allValuesFrom ( $\varphi(\text{FC}_1)$ cardinality(1) ) restriction ( $\varphi(\text{FU}_2)$ allValuesFrom ( $\varphi(\text{FC}_2)$ cardinality(1) ) ) ); FOR i = 1, 2 DO: The fuzzy property axioms: ObjectProperty ( $\varphi(\text{FU}_i)$ domain ( $\varphi(\text{FA})$ range ( $\varphi(\text{FC}_i)$ ) ); ObjectProperty ( $\text{FV}_i$ domain ( $\varphi(\text{FC}_i)$ range ( $\varphi(\text{FA})$ inverseOf $\varphi(\text{FU}_i)$ ) ); The fuzzy class axioms: Class ( $\varphi(\text{FC}_i)$ partial restriction ( $\text{FV}_i$ allValuesFrom ( $\varphi(\text{FA})$ ) ) ).
Each fuzzy class declaration: Class FC <i>type-is</i> Record FA <sub>1</sub> :FD <sub>1</sub> , ..., FA <sub>k</sub> :FD <sub>k</sub> <i>End</i>	Create fuzzy class/property axioms: Class ( $\varphi(\text{FC})$ partial restriction ( $\varphi(\text{FA}_1)$ allValuesFrom ( $\varphi(\text{FD}_1)$ cardinality(1) ) ... restriction ( $\varphi(\text{FA}_k)$ allValuesFrom ( $\varphi(\text{FD}_k)$ cardinality(1) ) ) ); DatatypeProperty ( $\varphi(\text{FA}_i)$ domain ( $\varphi(\text{FC})$ range ( $\varphi(\text{FD}_i)$ [Functional] ) where i = 1, 2, ..., k .

Below we discuss the effectiveness of the translation, which can be sanctioned by the following Theorem 1.

**Theorem 1.** For every FOOD model  $FS = (\text{FC}_{FS}, \text{FA}_{FS}, \text{FD}_{FS})$ ,  $FJ$  is a fuzzy database state w.r.t.  $FS$ , and  $\varphi(FS)$  is the fuzzy OWL DL ontology structure derived from  $FS$  by Definition 3. There exist mappings:

- $\alpha_{FS}$  from fuzzy database state  $FJ$  to model of  $\varphi(FS)$  such that: For each legal fuzzy database state  $FJ$  for  $FS$ , there is  $\alpha_{FS}(FJ)$  which is a model of  $\varphi(FS)$ ;
- $\beta_{FS}$  from model of  $\varphi(FS)$  to fuzzy database state such that: For each model  $FI$  of  $\varphi(FS)$ , there is  $\beta_{FS}(FI)$  which is a legal fuzzy database state for  $FS$ .

*Proof:* The following briefly gives the proof of the first part of Theorem 1. Given a fuzzy database state  $FJ$ , and  $\bullet^{FJ}$  is a fuzzy interpretation function that can map each element in Fig. 1 to its corresponding value in Fig. 3, we can define a fuzzy interpretation  $\alpha_{FS}(FJ)$  of  $\varphi(FS)$  as follows:

- The domain elements  $\Delta^{\alpha_{FS}(FJ)}$  of interpretation  $\alpha_{FS}(FJ)$  of  $\varphi(FS)$  are constituted by values of  $FJ$  (e.g., Fig. 3). Since each object of  $FS$  is assigned a structured value, in order to explicitly represent in fuzzy ontology the type structure of the object, we denote with  $\Delta_{fid}$ ,  $\Delta_{frec}$ ,  $\Delta_{fset}$  the domain elements of  $\Delta^{\alpha_{FS}(FJ)}$  corresponding to object identifiers, fuzzy record values, and fuzzy set values in  $FV_{FJ}$ , respectively.
- The fuzzy OWL DL identifier set  $FID_0$  of  $\varphi(FS)$  in Definition 3 are defined:  
 $(\varphi(X))^{\alpha_{FS}(FJ)} = X^{FJ}$ , where  $X \in FT$ ;  
 $(\varphi(FD_i))^{\alpha_{FS}(FJ)} = FD_i^{FJ}$ ;  
 For each class declaration *Class FC type-is Record FA<sub>1</sub>:FD<sub>1</sub>,..., FA<sub>k</sub>:FD<sub>k</sub> End*, we have  $(\varphi(FA_i))^{\alpha_{FS}(FJ)} = \{ \langle c, d_i \rangle \in \Delta^{\alpha_{FS}(FJ)} \times \Delta^{\alpha_{FS}(FJ)} \mid c \in FC^{FJ} \wedge d_i \in FD_i^{FJ} \}$ , where  $i = 1, 2, \dots, k$ ;  
 For class declaration *Class FC<sub>1</sub> type-is Record FA: Set-of FC<sub>2</sub> End*, we have  $(\varphi(FU_j))^{\alpha_{FS}(FJ)} = \{ \langle r, c_j \rangle \in \Delta^{\alpha_{FS}(FJ)} \times \Delta^{\alpha_{FS}(FJ)} \mid r \in FA^{FJ} \wedge c_j \in FC_j^{FJ} \}$ ,  $j = 1, 2$ .

$FO_S = (FID_0, FAxiom_0)$ . For reasons of space,  $FID_0$  and parts of axioms are omitted.  
 $FAxiom_0 = \{$   
 SubClassOf ( Old-Computer Computer );  
 SubClassOf ( Young-Employee Employee );  
 Class ( Computer partial unionOf ( Hp-Computer, Dell-Computer ) );  
 SubClassOf ( Hp-Computer Computer );  
 SubClassOf ( Dell-Computer Computer );  
 Class ( Old-Computer partial restriction ( ComID allValuesFrom (xsd:String) cardinality (1) ) restriction ( Brand allValuesFrom (xsd:String) cardinality (1) ) restriction ( UseYear allValuesFrom (xsd:Integer) cardinality (1) ) );  
 Class ( Young-Employee partial restriction ( EmpID allValuesFrom (xsd:String) cardinality (1) ) restriction ( Name allValuesFrom (xsd:String) cardinality (1) ) restriction ( Age allValuesFrom (xsd:fuzzy Integer) cardinality (1) ) );  
 Class ( Use partial restriction ( Useby allValuesFrom (Young-Employee) cardinality(1) ) restriction ( Useof allValuesFrom (Old-Computer) cardinality(1) ) );  
 Class ( Young-Employee partial restriction ( invof\_ Useby allValuesFrom (Use) ) );  
 Class ( Old-Computer partial restriction ( invof\_ Useof allValuesFrom (Use) ) );  
 ObjectProperty ( invof\_ Useby domain (Young-Employee) range (Use) inverseOf Useby );  
 ObjectProperty ( invof\_ Useof domain (Old-Computer) range (Use) inverseOf Useof );  
 ObjectProperty ( Useby domain (Use) range (Young-Employee) );  
 ObjectProperty ( Useof domain (Use) range (Old-Computer) );  
 DatatypeProperty ( EmpID domain (Young-Employee) range (xsd:String) [Functional] );  
 DatatypeProperty ( Name domain (Young-Employee) range (xsd:String) [Functional] );  
 DatatypeProperty ( Age domain (Young-Employee) range (xsd:Integer) [Functional] );  
 DatatypeProperty ( ComID domain (Old-Computer) range (xsd:Integer) [Functional] );  
 DatatypeProperty ( Brand domain (Old-Computer) range (xsd:String) [Functional] );  
 ... }

**Fig. 4.** The fuzzy OWL DL ontology structure  $\varphi(FS1)$  derived from the FOOD model  $FS1$



Here, we omit the complete proof of Theorem 1 for reasons of space, and the proof of the second part can be done similarly, they are a mutually inverse process.

**Example 1.** Given a *FOOD model*  $FS_1$  in Fig. 1, by Definition 3, we can obtain the corresponding *fuzzy OWL DL ontology structure*  $FO_S = \phi(FS_1)$  in Fig. 4.

## 5.2 Mapping Database Instances to Fuzzy OWL DL Ontology Instances

In the previous section, we realize the translation from the FOOD model to the fuzzy OWL DL ontology structure. In order to build fuzzy ontologies of high completeness, in the following, we propose an approach for mapping *the database instances* w.r.t. a FOOD model (e.g., Fig. 3) to *the fuzzy OWL DL ontology instances* (i.e., a set of fuzzy individual axioms in Table 1).

Firstly, let us briefly sketch the assertional formalisms used in the fuzzy Object-Oriented databases and fuzzy OWL DL ontologies.

The assertional formalisms of fuzzy OWL DL ontologies are represented by the fuzzy individual axioms, which contain the axioms (see Table 1): Individual ( $o$  type( $C_l$ ) [ $\bowtie$   $m_l$ ] ... value( $R_l$ ,  $o_l$ ) [ $\bowtie$   $k_l$ ] ... value( $U_l$ ,  $v_l$ ) [ $\bowtie$   $l_l$ ]...); DifferentIndividuals ( $o_1 \dots o_n$ ); SameIndividual ( $o_1 \dots o_n$ ).

From Section 4, a fuzzy Object-Oriented database, which describes the real world by means of objects, values, and their mutual relationships, can be considered as a finite set of assertions [1], [9], [18]. Based on [1], the assertional formalisms of the database instances with respect to a FOOD model specify that: A fuzzy object  $fo$  is an instance of fuzzy class  $FC$  with membership degree of  $n$  by means of fuzzy assertion  $fo: FC: n$ , where  $n \in [0, 1]$ ; The fuzzy structured value associated with  $fo$  by means of fuzzy assertion  $fo: [FA_1:FV_1:n_1, \dots, FA_k:FV_k:n_k]$ , where  $FV_i$  is the value of attribute  $FA_i$ ,  $n_i \in [0, 1]$  denotes the membership degree,  $FA_i$  may be a fuzzy datatype attribute or a fuzzy object attribute, and  $i \in \{1 \dots k\}$ . Here, since the value of an attribute  $FA_i$  may be a possibility distribution, for simplicity,  $FV_i: n_i$  only denotes one element of the possibility distribution; In addition, since the fuzzy subclass/superclass relationships in FOOD models can be assessed by utilizing the inclusion membership degree of objects to the class [8], [13], the assertional formalism of fuzzy subclass/superclass relationships can be re-expressed by the above assertional formalism of objects to the class.

Based on the discussion above, the mappings from the *database instances* w.r.t. a FOOD model to the *fuzzy OWL DL ontology instances* can be established by Definition 4.

**Definition 4 (Instance Translation).** Given the database instances with respect to a FOOD model (i.e., a finite set of assertions), the corresponding fuzzy OWL DL ontology instances (i.e., a set of fuzzy individual axioms) can be derived as the mapping rules shown in Table 3.

In Table 3, notice that:

- (i) In order to represent that the membership degree of an object to a class is equal to  $n$  (i.e.,  $fo: FC: n$ ), the symbol  $\bowtie'$  denotes  $\geq$  and  $\leq$ .

For example, the axiom *Individual* ( $fo$  type( $FC$ ) [ $\bowtie'$   $n$ ]) denotes two axioms *Individual* ( $fo$  type( $FC$ ) [ $\geq n$ ]) and *Individual* ( $fo$  type( $FC$ ) [ $\leq n$ ]).

- (ii) When the membership degree  $n = 1.0$ , the [ $\bowtie'$   $n$ ] part is omitted in a fuzzy individual axiom.

**Table 3.** Mapping rules from database instances to fuzzy OWL DL ontology instances

Database instances	Fuzzy OWL DL ontology instances $FO_1$	
Each fuzzy object symbol $fo$	A fuzzy individual identifier $\varphi(fo) \in FIID_0$	
Each fuzzy class symbol $FC$	A fuzzy class identifier $\varphi(FC) \in FCID_0$	See Table 2
Each fuzzy datatype attribute $FA_i$	A fuzzy datatype property identifier $\varphi(FA_i) \in FDPID_0$ , denoted by $U_i$	
Each fuzzy object attribute $FA$	A fuzzy class identifier $\varphi(FA) \in FCID_0$ ; In addition, adding the additional fuzzy object property identifiers, denoted by $R_i$	
The fuzzy assertion $fo: FC: n$	The fuzzy individual axiom: Individual ( $\varphi(fo)$ type( $\varphi(FC)$ ) [ $\bowtie$ ' $n$ ] )	See Fig. 5
The fuzzy assertion: $fo:[FA_1:FV_1:n_1, \dots, FA_k:FV_k:n_k]$	The fuzzy individual axiom: Individual ( $\varphi(fo)$ value( $R_i, o_i$ ) [ $\bowtie$ ' $n_i$ ]... value( $U_i, v_i$ ) [ $\bowtie$ ' $n_i$ ]... ) where $o_i, v_i \in FJ, i \in \{1 \dots k\}$	

**Example 2.** Given the *database instances* in Fig. 3 (w.r.t. the FOOD model  $FS_1$  in Fig. 1), by Definition 4, we can obtain the corresponding *fuzzy OWL DL ontology instances*  $FO_1$  in Fig. 5 (w.r.t. the fuzzy OWL DL ontology structure  $FO_S$  in Fig. 4).

<p>Given the database instances in Fig. 3, the fuzzy OWL DL ontology instances <math>FO_1</math> can be derived as follows:</p> <p>Notice that, since each object attribute <math>FA</math> is mapped to a fuzzy class identifier <math>\varphi(FA) \in FCID_0</math>, we need to add an additional object symbol <math>o'</math> such that <math>o'</math> belongs to class <math>\varphi(FA)</math>. <math>FIID_0 = \{ o_1, o_2, o_3, o' \}</math> ;</p> <p><math>FAxiom_0 = \{</math></p> <p>DifferentIndividuals (<math>o_1, o_2, o_3, o'</math>) ;</p> <p>Individual ( <math>o_1</math> type(Young-Employee) [<math>\bowtie</math>' 0.9] ) ;</p> <p>Individual ( <math>o_2</math> type(Old-Computer) [<math>\bowtie</math>' 0.85] ) ;</p> <p>Individual ( <math>o_3</math> type(Old-Computer) [<math>\bowtie</math>' 0.8] ) ;</p> <p>Individual ( <math>o'</math> type(Use) ) ;</p> <p>Individual ( <math>o'</math> value(Useby, <math>o_1</math>) value(Useof, <math>o_2</math>) [<math>\bowtie</math>' 0.8] value(Useof, <math>o_3</math>) [<math>\bowtie</math>' 0.7] ) ;</p> <p>Individual ( <math>o_1</math> value(EmpID, <math>o_1</math>) value(Name, Chris) value(Age, 27) [<math>\bowtie</math>' 0.8] value(Age, 30) [<math>\bowtie</math>' 0.9] value(invof_Useby, <math>o'</math>) ) ;</p> <p>Individual ( <math>o_2</math> value(ComID, <math>o_2</math>) value(Brand, Dell) value(UseYear, 4) value(invof_Useof, <math>o'</math>) [<math>\bowtie</math>' 0.8] ) ;</p> <p>Individual ( <math>o_3</math> value(ComID, <math>o_3</math>) value(Brand, Hp) value(UseYear, 3) value(invof_Useof, <math>o'</math>) [<math>\bowtie</math>' 0.7] ) .</p> <p>}</p>
---

**Fig. 5.** The fuzzy OWL DL ontology instances derived from database instances in Fig. 3

## 6 Automated Learning Tool and Case Study

Based on the proposed approach in Section 5, we developed an automated learning tool called *FOOD2FOWL*, which can automatically construct fuzzy OWL DL ontologies from FOOD models and the corresponding database instances. For reasons of space, the following briefly introduces the implementation of *FOOD2FOWL*.

### 6.1 Design and Implementation of *FOOD2FOWL*

The tool *FOOD2FOWL* includes three main modules: *parsing module* that parses the FOOD model file, *translation module* that translates the parsed file into the fuzzy OWL DL ontology, and *output module* that produces the constructed fuzzy OWL DL ontology as a text file.

The implementation of *FOOD2FOWL* is based on Java 2 JDK 1.5 platform. The parsing module uses the regular expression to parse the FOOD model file and stores the parsed results as Java ArrayList classes; translation module uses Java class methods to translate the FOOD model and database instances into the fuzzy OWL DL ontology structure and instances based on the proposed approach in Section 5; output module produces the resulting fuzzy OWL DL ontology which is saved as a text file and displayed on the tool screen (see Fig. 6).

### 6.2 Case Study

We have carried out lots of case studies using *FOOD2FOWL*. For saving space, here we give an example which can show that the proposed approach is feasible and the implemented tool is efficient. Fig. 6 shows the screen snapshot of *FOOD2FOWL*, which displays the translations from the FOOD model (Fig. 1) and the corresponding database instances (Fig. 3) to the fuzzy OWL DL ontology structure (Fig. 4) and the fuzzy OWL DL ontology instances (Fig. 5).

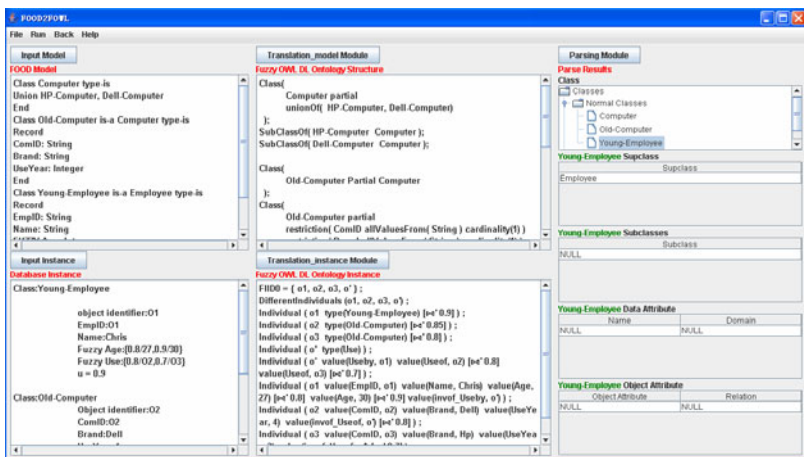


Fig. 6. Screen snapshot of *FOOD2FOWL*

## 7 Conclusions and Future Work

The real power of the Semantic Web will be realized only when people create much machine-readable content, and ontologies play a key role in this effort. In this paper, we developed a formal approach and an automated tool for constructing fuzzy Semantic Web ontologies from fuzzy Object-Oriented database (FOOD) models. The fuzzy ontology was introduced. A kind of formal definition of FOOD models were proposed. Furthermore, we realized the translations from the FOOD model and its corresponding database instances to the fuzzy ontology structure and fuzzy ontology instances. Following the proposed approach, an automated learning tool called *FOOD2FOWL* was implemented, which can automatically construct fuzzy ontologies from FOOD models. All of these will facilitate the development of Web ontologies and the realization of semantic interoperations between lots of existing database applications and the Semantic Web.

In the future, we aim at developing the other approaches of constructing (fuzzy) ontology. Moreover, it should be noted that many databases have not been designed following the disciplined methodologies, thus we will experiment with other FOOD models to enrich our construction approach.

**Acknowledgments.** We thank the anonymous reviewers for their valuable suggestions. The work is supported by the *Fundamental Research Funds for the Central Universities* (N090604012 and N090504005) and the *National Natural Science Foundation of China* (60873010), and in part by the *Program for New Century Excellent Talents in University* (NCET-05-0288).

## References

1. Artale, A., Cesarini, F., Soda, G.: Describing database objects in a concept language environment. *IEEE Trans. Knowledge and Data Engineering* 8(2), pp. 345–351 (1996)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284(5), 34–43 (2001)
3. Baader, F., McGuinness, D., Nardi, D. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge (2003)
4. Blanco, I.J., Vila, M.A., Martinez-Cruz, C.: The use of ontologies for representing database schemas of fuzzy information. *J. Intelligent Syst.* 23(4), 419–445 (2008)
5. Calegari, S., Ciucci, D.: Fuzzy ontology, fuzzy description logics and fuzzy-owl. In: Masulli, F., Mitra, S., Pasi, G. (eds.) *WILF 2007*. LNCS (LNAI), vol. 4578, pp. 118–126. Springer, Heidelberg (2007)
6. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. *J. Artificial Intelligence Research* 11(2), 199–240 (1999)
7. Cross, V., Caluwe, R., Vangyseghe, N.: A perspective from the fuzzy object data management group (FODMG). In: *Proc. Fuzzy Systems*, vol. 2, pp. 721–728 (1997)
8. George, R., Srikanth, R., Petry, F.E., Buckles, B.P.: Uncertainty management issues in the object-oriented data model. *IEEE Trans. Fuzzy Systems* 4(2), 179–192 (1996)
9. Koyuncu, K., Yazici, A.: IFOOD: an intelligent fuzzy object-oriented database architecture. *IEEE Trans. on Knowledge and Data Engineering (TKDE)* 15(5), 1137–1154 (2003)
10. Koide, S., Aasman, J., Haflich, S.: OWL vs. object oriented programming. In: *Workshop on Semantic Web Enabled Software Engineering (SWESE) at ISWC 2005* (2005)
11. Ling, T.C., Yaacob, M.H., Phang, K.K.: Fuzzy database framework-Relational versus Object-oriented model. In: *Proc. of Intelligent Information Systems*, pp. 246–250 (1997)

12. Lubyte, L., Tessaris, S.: Automatic extraction of ontologies wrapping relational data sources. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 128–142. Springer, Heidelberg (2009)
13. Ma, Z.M., Zhang, W.J., Ma, W.Y.: Extending object-oriented databases for fuzzy information modeling. *Information Systems* 29(5), 421–435 (2004)
14. Ma, Z.M., Yanhui, L.Y., Yan, L.: A Fuzzy Ontology Generation Framework from Fuzzy Relational Databases. *Int. J. Semantic Web Information Systems* 4(3), 1–15 (2008)
15. Maedche, A., Staab, S.: Ontology Learning for the Semantic Web. *IEEE Intelligent Systems* 16(2), 72–79 (2001)
16. Meditskos, G., Bassiliades, N.: A Rule-Based Object-Oriented OWL Reasoner. *IEEE Trans. Knowledge & Data Engineering* 20(3), 397–410 (2008)
17. Mota, L., Botelho, L.M., Mendes, H., Lopes, A.: O3F: an object oriented ontology framework. In: Proc. of the Second International Joint Conference on AAMAS'03 (2003)
18. Ma, Z.M.: *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications*. Idea Group Publishing (2004)
19. Oren, E., Heitmann, B., Decker, S.: ActiveRDF: Embedding Semantic Web data into object-oriented languages. *Journal of Web Semantics* 6(3), 191–202 (2008)
20. Ortega, F.B.: *Managing Vagueness in Ontologies*. E.T.S. de Ingenierías Informática y de Telecomunicación. Departamento Ciencias de la Computación e I. A. (2008)
21. Quan, T.T., Hui, S.C., Fong, A.C.M.: Automatic fuzzy ontology generation for Semantic Web. *IEEE Trans. on Knowledge & Data Engineering* 18(6), 842–856 (2006)
22. Roger, M., Simonet, A., Simonet, M.: Bringing together description logics and database in an object oriented model. In: Hameurlain, A., Cicchetti, R., Traummüller, R. (eds.) DEXA 2002. LNCS, vol. 2453, pp. 504–513. Springer, Heidelberg (2002)
23. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive web sites into the semantic Web. In: Proc. 17th ACM Symp. on Applied Computing, pp. 1100–1107 (2002)
24. Straccia, U.: Towards a fuzzy description logic for the semantic Web. In: Gómez-Pérez, A., Euzenat, J. (eds.) *ESWC 2005*. LNCS, vol. 3532, pp. 167–181. Springer, Heidelberg (2005)
25. Sanchez, E., Yamanoi, T.: Fuzzy ontologies for the semantic web. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreassen, T., Christiansen, H. (eds.) *FQAS 2006*. LNCS (LNAI), vol. 4027, pp. 691–699. Springer, Heidelberg (2006)
26. Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J.Z., Horrocks, I.: Fuzzy OWL: Uncertainty and the Semantic Web. In: *International Workshop of OWL: Experiences and Directions (OWL-ED 2005)*, Galway, Ireland (2005)
27. Xu, Z., Cao, X., Dong, Y.: Formal approach and automated tool for translating ER schemata into OWL ontologies. In: Dai, H., Srikant, R., Zhang, C. (eds.) *PAKDD 2004*. LNCS (LNAI), vol. 3056, pp. 464–475. Springer, Heidelberg (2004)
28. Xu, Z., Zhang, S., Dong, Y.: Mapping between Relational Database Schema and OWL Ontology for Deep Annotation. In: *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, December 18–22, pp. 548–552 (2006)
29. Zhang, F., Ma, Z.M., Cheng, J., Meng, X.: Fuzzy Semantic Web Ontology Learning from Fuzzy UML Model. In: Proc. of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009), pp. 1007–1016 (2009)
30. Zhang, F., Ma, Z.M., Lv, Y., et al.: Formal Semantics-Preserving Translation from Fuzzy ER Model to Fuzzy OWL DL Ontology. In: Proc. of 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2008), pp. 503–509 (2008)
31. Zadeh, L.A.: Fuzzy sets. *Information and Control* 8(3), 338–353 (1965)
32. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Systems* 1(1), 3–28 (1978)

# Combining Fuzzy Logic and Semantic Web to Enable Situation-Awareness in Service Recommendation

Alessandro Ciaramella<sup>1</sup>, Mario G.C.A. Cimino<sup>2</sup>,  
Francesco Marcelloni<sup>2</sup>, and Umberto Straccia<sup>3</sup>

- <sup>1</sup> IMT Lucca Institute for Advanced Studies, Piazza San Ponziano 6, 55100 Lucca, Italy  
a.ciaramella@imtlucca.it
- <sup>2</sup> Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni,  
University of Pisa, Via Diotisalvi 2, 56122 Pisa, Italy  
{m.cimino, f.marcelloni}@iet.unipi.it
- <sup>3</sup> Istituto di Scienza e Tecnologie dell'Informazione (ISTI - CNR), Pisa, Italy  
straccia@isti.cnr.it

**Abstract.** Mobile Internet is rapidly growing and an enormous quantity of resources are currently available. Thus, the common mechanisms used up to now to locate resources, such as browsing and searching, do not look anymore to be effective in helping users in mobility. Indeed, the user's personal information space can be very large, with respect to the limited interaction capabilities of mobile devices. This paper proposes a situation-aware framework for providing personalized resources in a proactive manner. Current situations of the user are inferred by exploiting domain knowledge expressed in terms of ontologies and semantic rules, which are represented in the well-known Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL), respectively. Uncertainty in some contextual rule conditions is handled by defining appropriate linguistic variables through the Fuzzy Control Language (FCL), a standard representation of fuzzy systems for data exchange among different implementations, and adopting a purposely-adapted coding of ontologies and rules. Uncertain conditions bring to infer more than one situation with different certainty degrees: these degrees are used to assign a rank to concurrent situations. Finally, situations are connected to a set of related resources to be recommended to the user.

**Keywords:** Context-awareness; Fuzzy Inference System; Mobile Service Recommender; Web Ontology; Semantic Rules.

## 1 Introduction

Nowadays, the number of mobile users is rapidly growing and more and more services and documents are available on marketplaces. Categorizing such resources is a very difficult task: there exists no standardized taxonomy, and the user preferences can vary from an individual to another [9]. Hence, finding the best-fitting service by browsing and searching in resource repositories is very time expensive for the common user.

Moreover, mobile devices are becoming more and more capable to accommodate many types of resources such as services, documents and links to external sources, enlarging incredibly the personal information space of a mobile user. On the contrary,

the interaction features of the devices are very poor, due mainly to user interfaces less comfortable and manageable than in a personal computer. Furthermore, often services have to be configured with a set of proper parameters in order to be executed effectively. These parameters can vary in dependence of specific circumstances, and users have often to provide textual inputs on small keypads [6].

Hence, a significant cognitive effort is required to users in mobility to find and configure the most appropriate resources among the many available [8][5]. A tool able to automatically searching for the most suitable resources for the specific situation and configuring them would be strongly desirable. To this aim, in this paper we propose a situation-aware framework for providing personalized resources in a proactive manner. Situation-awareness is enabled by a specific engine based on semantic web technologies and fuzzy logic.

More specifically, contextual information is maintained in the system by domain ontology [23] and is enriched with a truth degree depending on a level of certainty. Situations are inferred by means of semantic rules [25], which take the fuzziness of the contextual antecedents into account, and are ranked depending on their fuzzy values. Unlike in our previous work [1], here fuzziness is directly managed within the semantic rules and the semantic inference engine rather than by a specific fuzzy inference engine. These situations allow the identification of specific tasks, on the basis of domain knowledge expressed in terms of task ontology, which represents common sense knowledge about user usual activities. Finally, the specific current task together with contextual information is used to recommend a set of resources, in a task-navigation paradigm [7], where the user is supported to find appropriate services and documents by relying on the task ontology.

## 2 Background

A large amount of research in recommendation systems has focused on providing personalized services [10]. In our approach, the current user situation is exploited to narrow down the set of recommendable resources, thus showing to the user only those resources that are relevant in the specific situation. In [26], the term *situation* refers to a business level concept that allows targeting precisely and at different levels of granularity the demand of the user at a certain time. More precisely, a situation can be defined as a collection of context information that is invariant as long as the situation occurs [26]. According to Dey and Abowd [4], context refers to any relevant information that can be used to characterize a user. Hence, a situation can be inferred as a consequence of rules that verify whether a set of contextual conditions hold in the system. For instance, the situation *meeting* can be inferred when the contextual conditions *user is stationary*, *user is located in the scheduled place at the scheduled time*, and *user is close to the meeting organizer* are verified in the system.

Several projects consider the use of ontology as a promising means to develop context-aware systems. In the framework of semantic web, an ontology is a knowledge model that describes a domain of interest using semantic aspects and structure. An ontology consists of: (i) facts representing explicit knowledge, consisting of concepts, their properties, and instances that represent entities described by concepts; (ii) axioms

and predicates representing implicit knowledge, by means of rules used to add semantics and to derive knowledge from facts [26]. In [21], a comparative analysis shows that the most promising assets for context modeling in ubiquitous computing environments can be found in the use of ontology.

To reflect the varying nature of context and to ensure a universal applicability of context-aware systems, context is typically represented at different levels of abstraction [15]. At the first level of raw context sources there are context data coming from sensor devices, or user applications. At this level, logic embodied in semantic languages does not allow a treatment of uncertainty and imprecision existent in real world [17]. For instance, a typical smartphone GPS receiver provides a device position with dynamic accuracy ranging from some meters to hundreds of meters, depending on many environmental variables. Thus, the situation recognition from the environment should rely on a vague characterization. Furthermore, these situations are often connected to specific user requirements, and then the system should offer a specification mechanism that is intuitive, for instance in terms of standardized natural language, as guaranteed by employing linguistic terms [27].

### 3 Overall Architecture

In our implementation, the situation-aware resource recommender is running on the mobile device as an advanced menu, whose elements are dynamically updated, according to the different situations in which the user is involved. The overall system architecture is shown in Fig. 1.

In the server side, the main module is represented by the *situation engine*, which is in charge of interpreting contextual conditions and assessing the user situations. Contextual conditions that are inherently vague, such as mobility and proximity state of users, are evaluated by means of fuzzy logic, i.e., enriched with a truth degree maintained in the ontology. Such degrees represent the extent to which the conditions hold in the system. For instance *the user is close to a place* is a contextual condition that can be characterized with a truth degree representing the level of closeness of the user to the place. Semantic rules enhanced with the ability of managing the uncertainty allow inferring multiple situations with an appropriate ranking. This allows the system to recommend the related resources with different priorities.

The control flow of the application is steered by the *application controller* module, which manages the activities of each module, granting access to different functions and data sources. The *contextual data sources* package comprises a set of interfacing modules for different data sources, such as geographical maps, users' personal calendars and positions. In particular, numerical data concerning users positions are fed by the *location detector* module. This module provides outdoor/indoor location estimation, also on the basis of several possible technologies, such as GPS, GSM, WiFi [22]. Regardless of the available technologies, the *location detector* provides a generalized interface in terms of position and accuracy.

The *Rule Translator* is an off-line module that translates the rules, expressed in a high-level language, into a well-established standard for semantic rules, the Semantic Web Rule Language (SWRL, [25]). Thus, designers can express how the system should



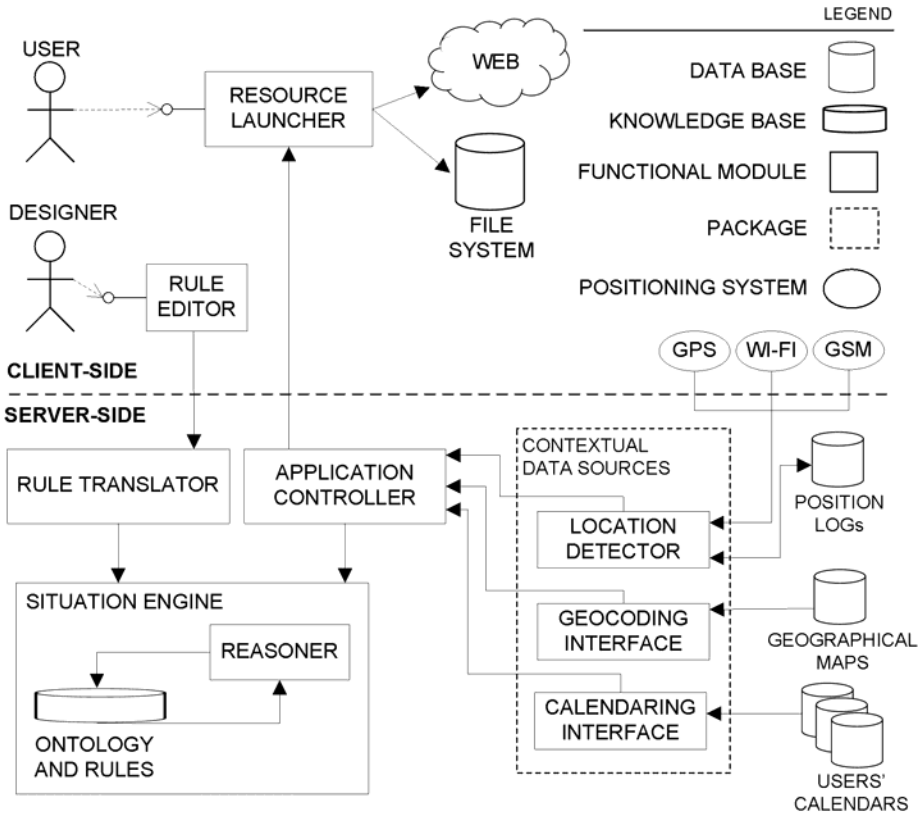


Fig. 1. Overall architecture of the situation-aware resource recommender

interpret contextual conditions in order to assess the most appropriate situations in a natural language close to their language. Further, the *Rule Translator* module allows the representation of the fuzzy logic within the SWRL, mapping directly the fuzzy information into the crisp ontology.

On the client side, the *Rule Editor* module allows a designer to configure and express the semantic rules for situation assessment. Finally, the *Resource Launcher* module shows the recommended resources to the user and allows the launch of these resources. In the following, the paper is focused on the design of the *situation engine* module.

#### 4 Semantic Domain Knowledge

In the system, domain and general knowledge is represented by the situation ontology and related semantic rules. The ontology has been developed by using the Web Ontology Language (OWL [23]), a W3C standard well-supported in most semantic engines. In the upper situation ontology, general context information is represented by basic concepts such as *User*, *Calendar*, *Device*, *Time* and *Place*.

In order to manage fuzzy information in an OWL compliant ontology, we established a representation pattern. The pattern is applicable to properties that are related to the same base variable and to the same pair of concepts. For instance, let us consider the base variable *distance*, and the concepts *User* and *Place*. Depending on the actual value of the distance, and considering a prefixed set of distance intervals, we can establish properties like *User is-close-to a Place* or *User is-far-from a Place*. The presence of each property depends on the membership of the distance value to a prefixed interval. For example, considering the first interval as *LowDistance = 0-10 meters*, it can be said that *User is-close-to* depends on *LowDistance*, more formally *is – close – to*<sub>|*LowDistance*</sub>. Fig. 2-a shows an abstract representation of this mechanism, for a series of *n* properties and related *n* intervals. Here, concepts have been enclosed in oval shapes, whereas properties are represented by arrows. In order to capture vagueness in this representation, we propose the extension shown in Fig. 2-b. Here, an OWL group of properties is transformed into a concept, which includes a specification of the degree for each property. In other words, we assert that there is a property with a certain degree. Each degree is the membership level of the base variable to a specific fuzzy set.

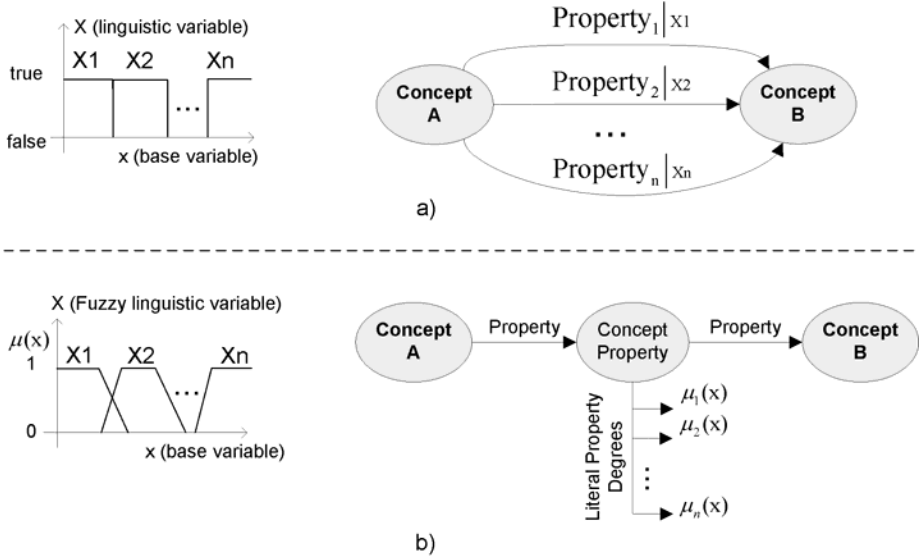


Fig. 2. An OWL-compliant fuzzy extension of a property

It is worth noting that this scheme can be used also in case of a property related to a single concept. In such case, the concept property corresponds to the concept itself.

In Fig. 3, the complete upper situation ontology is presented. This ontology is made of 10 general concepts and 25 properties, together with 5 concepts and 14 properties for the fuzzy representation. In particular, general concepts such as *Time* and *Place* are inherited from publicly available ontologies [24,5], according to the best practices of reusing domain ontologies. In the figure, such external ontologies are enclosed in

dashed rectangular shapes. Concepts are connected by properties, represented with directed black edges in the figure. Edges with white arrowhead show classical inheritance (i.e., an is-a relation).

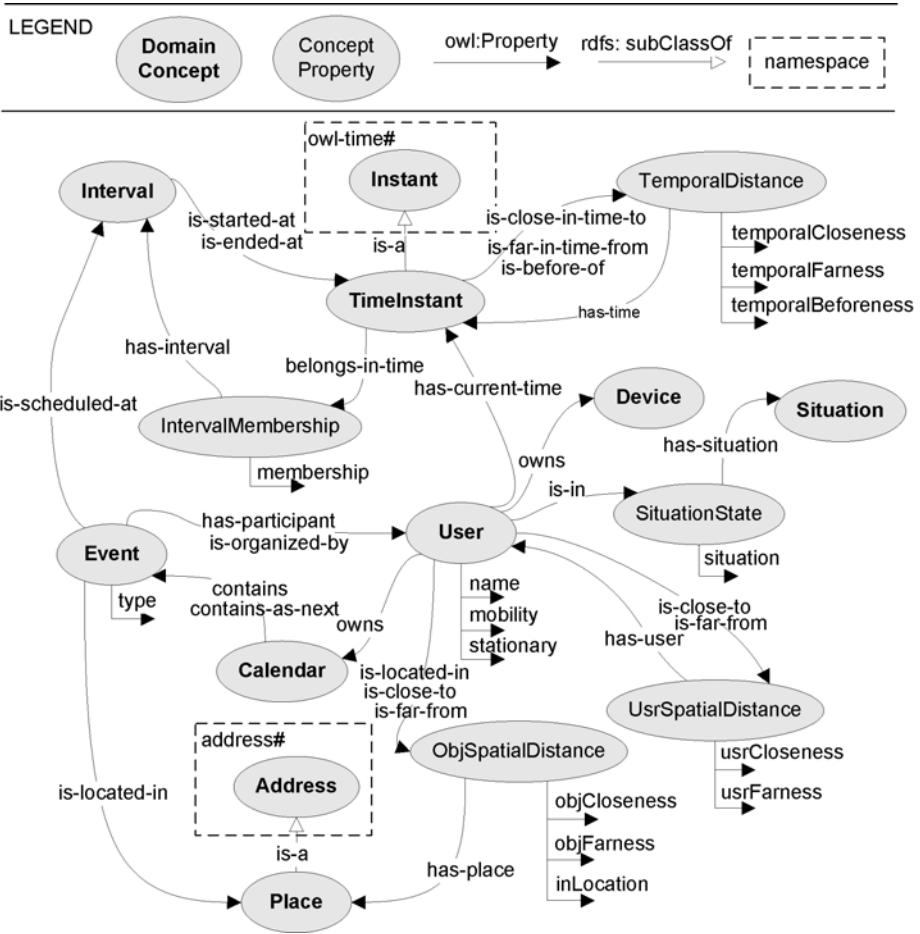


Fig. 3. The upper situation ontology

The model comprises a set of rules to infer the current situations on the basis of the situation ontology. Rules are expressed in the Semantic Web Rule Language SWRL, an emerging standard that extends OWL with additional rule-based knowledge representation. In terms of expressiveness, this reasoning standard corresponds to description logics, a particular decidable fragment of first order logic, and is named OWL DL [5]. Fig. 4 shows an example of rule in human readable syntax (a), commonly used in the literature, and in natural language (b). We point out that there are two types of antecedent

conditions, i.e., crisp (binary) and fuzzy, represented in Fig. 4 in bold and italic bold, respectively. The conditions *is a participant* and *has type* are derived from the user's calendar, and are inherently crisp, whereas the other conditions can be assessed only with vagueness. This implies that also the conclusion inferred from the rule is characterized by vagueness. This vagueness can be represented directly in SWRL (see next sections), which implements some mechanisms to express truth degrees and related membership functions.

<pre>owns(?user1, ?aCalendar) ^ contains-as-next(?aCalendar, ?nextEvent) ^ is-scheduled-at(?nextEvent, ?anInterval) ^ is-started-at(?anInterval, ?aTime) ^ mobility(?user1, ?user1mobility) ^ has-current-time(?user1, ?user1Time) ^ is-before-of(?user1Time, ?temporalDistance) ^ has-time(?temporalDistance, ?aTime) ^ type(?nextEvent, "business") ^ Pre-Meeting-on-Movement(?aSituation) ⇒ is-in-a-situation(?user1,?aSituation)</pre> <p style="text-align: center;">(a)</p>
<pre>IF <i>user1 IS A PARTICIPANT</i> to the scheduled event AND <i>user1 IS moving</i> AND <i>user1Time IS BEFORE</i> the scheduled event start-time AND event HAS TYPE <i>business</i> THEN <i>user1 IS IN A SITUATION OF pre-meeting-on-movement</i></pre> <p style="text-align: center;">(b)</p>

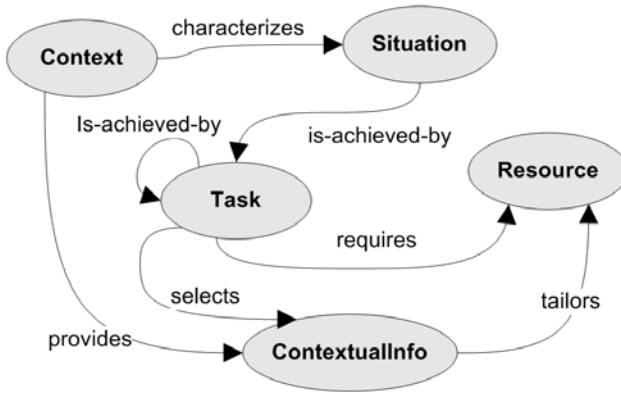
Fig. 4. A rule example

Once some situations have been inferred, with a certainty degree, a task ontology allows connecting a situation to specific tasks, and then specific tasks to specific resources to be recommended. Furthermore, such resources are tailored by proper contextual information, selected according to the identified user task. In Fig. 5 the upper task ontology is represented.

## 5 Managing the Uncertainty

There is some uncertainty in many contextual conditions related to real-world events. For instance, the condition *user1 is before the scheduled event start-time*, in Fig. 4b, can be assessed only with a certainty degree. This uncertainty can arise, for instance, from lack of precision in the information stored in the users calendar. Furthermore, it is possible that noise affects sensed data. For instance, the condition *user1 is moving* requires an estimation of the user's speed, often known only with a limited accuracy.

Fuzzy set theory and fuzzy logic have proved to be a promising approach to manage the natural uncertainty that affects such contextual data [16]. In order to evaluate



**Fig. 5.** The upper task ontology

the certainty degree of the contextual conditions, a number of linguistic variables have been defined. The universe of definition of such variables is partitioned with trapezoidal membership functions. An appropriate tuning of these functions has been carried out by means of experimental data.

Linguistic variables have been described using the Fuzzy Control Language (FCL, [2]), a standard representation of fuzzy systems for data exchange among different implementations. An example of the linguistic variable *speed*, used to decide about the user mobility, is shown in Fig. 6.

## 6 A Simple Integration of Fuzzy Logic into SWRL

In our implementation, we expressed fuzzy rules, such as the one described in Fig. 4-b, within SWRL, which however does not directly support fuzzy rules. While we refer the interested reader on fuzzy extensions of the logics behind Semantic Web Languages to [14,19,20], here we show that there is a simple way to encode the fuzzy rules into a crisp rule language supporting arithmetic built-in functions and, thus, in SWRL, making them directly available in current reasoners and in the Protégé editor<sup>1</sup>. In fact, we followed the below mentioned method to correctly deal with our fuzzy rule base.

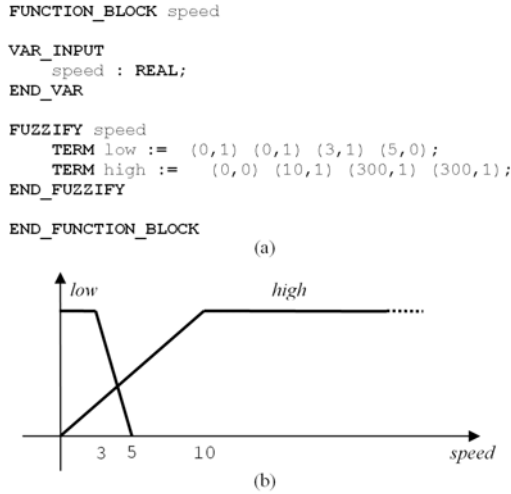
In our setting, a *fuzzy rule* is of the form (which closely resembles [13])

$$R(\mathbf{x})[s] \leftarrow \exists \mathbf{y}. R_1(\mathbf{z}_1)[s_1], \dots, R_l(\mathbf{z}_l)[s_l], s = f(s_1, \dots, s_l)$$

where

1.  $R$  is an  $n$ -ary relation, every  $R_i$  is an  $n_i$ -ary relation;
2.  $\mathbf{x}$  are the *distinguished variables*.
3.  $\mathbf{y}$  are existentially quantified variables called the *non-distinguished variables*. We omit to write  $\exists \mathbf{y}$  when  $\mathbf{y}$  is clear from the context;

<sup>1</sup> <http://protege.stanford.edu/>



**Fig. 6.** Definition of linguistic labels in FCL

4.  $\mathbf{z}_i, \mathbf{z}'_i$  are tuples of constants or variables in  $\mathbf{x}$  or  $\mathbf{y}$ ;
5.  $s, s_1, \dots, s_l$  are distinct variables and different from those in  $\mathbf{x}$  and  $\mathbf{y}$ , called *scores* or *truth degrees*;
6.  $f$  is a *scoring* total function  $f: [0, 1]^l \rightarrow [0, 1]$ , which combines the scores of the  $l$  relations  $R_i(c'_i)$  into an overall *score* to be assigned to the rule head  $R(c)$ . We assume that  $f$  can be computed in finite time.

We call  $R(\mathbf{x})[s]$  the *head*,  $\exists \mathbf{y}. R_1(\mathbf{z}_1)[s_1], \dots, R_l(\mathbf{z}_l)[s_l]$  the *body* and  $s = f(s_1, \dots, s_l)$  the *scoring atom*. We also allow the scores  $[s], [s_1], \dots, [s_l]$  and the scoring atom to be omitted. In this case we assume the value 1 for  $s_i$  and  $s$  instead. The informal meaning of such a rule is: if  $\mathbf{z}_i$  is an instance of  $R_i$  to degree at least or equal to  $s_i$ , then  $\mathbf{x}$  is an instance of  $R$  to degree at least or equal to  $s$ , where  $s$  has been determined by the scoring atom, *i.e.*  $s = f(s_1, \dots, s_l)$ .

As an example, in the following we show the high-level encoding of the fuzzy rule in Fig. 4b:

```

is-in-a-situation(?user1, ?aSituation)[s] ← owns(?user1, ?aCalendar),
    contains-as-next(?aCalendar, ?nextEvent),
    is-located-in(?nextEvent, ?aPlace),
    is-scheduled-at(?nextEvent, ?anInterval),
    is-started-at(?anInterval, ?aTime),
    mobility(?user1, 'moving')[s1],
    has-current-time(?user1, ?userTime),
    is-before-of(?userTime, ?aTime)[s2],
    type(?nextEvent, 'business'),
    Pre-Meeting-on-Movement(?aSituation),
    s = min(s1, s2)

```

(1)

Note that the final degree  $s$  of being in a “pre-meeting-on-movement” situation is determined by the minimum of the users’ degree of being moving ( $s_1$ ) and being before a meeting ( $s_2$ ) (here, “moving” and “before” are considered fuzzy concepts). So, here the scoring combination function  $f$  is the minimum, which is also the function used in all the rules we have developed in our specific application. Of course, other functions can be used as well such as any so-called *t-norm* (used to combine conjunctive information) [12].

A rule base  $\mathcal{R}$  is a finite set of fuzzy rules, which we assume to be *acyclic*. This latter notion is defined as follows: we say that a relation  $R$  *directly uses* a relation  $R'$  if there is a rule in  $\mathcal{R}$  having  $R$  as head and  $R'$  occurring in its body. Let *uses* be the transitive closure of the relation “directly uses”. Then we say that  $\mathcal{R}$  is *acyclic* iff for any relation  $R$  it is not the case that  $R$  uses  $R$ . Please note that acyclicity is required to guarantee decidability. Note that cyclic rules bases can be allowed if specific conditions are met on the score combination functions (see *e.g.*, [18], for more on this issue), but we do not address them here.

We point out that we may represent a fuzzy rule in a succinct way as

$$R(\mathbf{x})[s] \leftarrow \exists \mathbf{y}. \phi(\mathbf{x}, \mathbf{y})[s],$$

where  $\phi(\mathbf{x}, \mathbf{y})[s]$  is

$$R_1(\mathbf{z}_1)[s_1], \dots, R_l(\mathbf{z}_l)[s_l], s = f(s_1, \dots, s_l).$$

We also impose that a rule base  $\mathcal{R}$  is such that there are no two rules in it with the same head. Note that this restriction is harmless. Indeed, in case we would like to have  $n$  rules with same head [2], *i.e.*

$$\begin{aligned} R(\mathbf{x})[s] &\leftarrow \exists \mathbf{y}_1. \phi_1(\mathbf{x}, \mathbf{y}_1)[s_1] \\ R(\mathbf{x})[s] &\leftarrow \exists \mathbf{y}_2. \phi_2(\mathbf{x}, \mathbf{y}_2)[s_2] \\ &\vdots \\ R(\mathbf{x})[s] &\leftarrow \exists \mathbf{y}_n. \phi_n(\mathbf{x}, \mathbf{y}_n)[s_n] \end{aligned}$$

then we may replace them with the  $n + 1$  rules

$$\begin{aligned} R_1(\mathbf{x})[s] &\leftarrow \exists \mathbf{y}_1. \phi_1(\mathbf{x}, \mathbf{y}_1)[s_1] \\ R_2(\mathbf{x})[s] &\leftarrow \exists \mathbf{y}_2. \phi_2(\mathbf{x}, \mathbf{y}_2)[s_2] \\ &\vdots \\ R_n(\mathbf{x})[s] &\leftarrow \exists \mathbf{y}_n. \phi_n(\mathbf{x}, \mathbf{y}_n)[s_n] \\ R(\mathbf{x})[s] &\leftarrow R_1(\mathbf{x})[s_1], \dots, R_n(\mathbf{x})[s_n], s = g(s_1, \dots, s_n) \end{aligned}$$

where  $R_1, \dots, R_n$  are new relation symbols, and  $g$  specifies how to combine the scores of the individual rules into one overall score to be assigned to  $R$ . Usually,  $g(s_1, \dots, s_n) = \max(s_1, \dots, s_n)$ , but in general, any so-called *s-norm* [12] (used to combine

---

<sup>2</sup> In our specific fuzzy rule base, we do not have this scenario, though we present how to deal with it as it works generally.

disjunctive information) may be appropriate as well. This transformation guarantees then that  $\mathcal{R}$  remains acyclic and that there are no two rules in it with same head.

It remains to show how to represent fuzzy rules in a crisp rule language, which however supports arithmetic built-in predicates to perform arithmetic operations. To this end we proceed as follows.

1. Any  $n$ -ary relation  $R$  becomes an  $n + 1$ -ary relation. The additional slot is used to store the score  $s$ . So, in any rule, an expression  $R(\mathbf{z})[s]$  is replaced with the predicate  $R(\mathbf{z}, s)$ . For instance,

$$\text{is-before-of}(\text{?userTime}, \text{?aTime})[s2]$$

becomes

$$\text{is-before-of}(\text{?userTime}, \text{?aTime}, s2) .$$

2. As our crisp rule language supports arithmetic built-in predicates, there is a way to express a rule

$$P_f(s_1, \dots, s_l, s) \leftarrow \text{built-in}(s = f(s_1, \dots, s_l))$$

which defines a predicate  $P_f(s_1, \dots, s_l, s)$  such that  $s = f(s_1, \dots, s_l)$ , using the built-in arithmetic operations of the rule language.

3. Now, we replace each rule

$$R(\mathbf{x})[s] \leftarrow \exists \mathbf{y}. R_1(\mathbf{z}_1)[s_1], \dots, R_l(\mathbf{z}_l)[s_l], s = f(s_1, \dots, s_l)$$

with the crisp rule

$$R(\mathbf{x}, s) \leftarrow \exists \mathbf{y}. R_1(\mathbf{z}_1, s_1), \dots, R_l(\mathbf{z}_l, s_l), P_f(s_1, \dots, s_l, s)$$

which concludes the case in which the rule language supports  $n$ -ary predicates. For instance, fuzzy rule (II) becomes

$$\text{min}(s1, s2, s3)$$

$$\leftarrow \text{built-in}(s3 = \text{min}(s1, s2))$$

$$\begin{aligned} \text{is-in-a-situation}(\text{?user1}, \text{?aSituation}, s) \leftarrow & \text{owns}(\text{?user1}, \text{?aCalendar}), \\ & \text{contains-as-next}(\text{?aCalendar}, \text{?nextEvent}), \\ & \text{is-located-in}(\text{?nextEvent}, \text{?aPlace}), \\ & \text{is-scheduled-at}(\text{?nextEvent}, \text{?anInterval}), \\ & \text{is-started-at}(\text{?anInterval}, \text{?aTime}), \\ & \text{mobility}(\text{?user1}, \text{'moving'}, s1), \\ & \text{has-current-time}(\text{?user1}, \text{?userTime}), \\ & \text{is-before-of}(\text{?userTime}, \text{?aTime}, s2), \\ & \text{type}(\text{?nextEvent}, \text{'business'}), \\ & \text{Pre-Meeting-on-Movement}(\text{?aSituation}), \\ & \text{min}(s1, s2, s) \end{aligned}$$



However, SWRL is a rule language supporting unary and binary predicates only. This is not a particular problem, as to this end, we may rely on a well-known procedure, called *reification*<sup>3</sup> (see also [3]), which allows to represent an  $n$ -ary relation via unary and binary relations. So, for instance, for the relation

$$\text{is-before-of}(\text{?userTime}, \text{?aTime}, s2)$$

we create a new class

$$\text{is-before-ofRelation}(\text{?aTimeReification})$$

with two additional properties

$$\begin{aligned} &\text{is-before-ofValue}(\text{?aTimeReification}, \text{?aTime}) \\ &\text{is-before-ofDegree}(\text{?aTimeReification}, s2) \end{aligned}$$

and, thus,  $\text{is-before-of}(\text{?userTime}, \text{?aTime}, s2)$  will be replaced with

$$\begin{aligned} &\text{is-before-of}(\text{?userTime}, \text{?aTimeReification}), \\ &\text{is-before-ofValue}(\text{?aTimeReification}, \text{?aTime}), \\ &\text{is-before-ofDegree}(\text{?aTimeReification}, s2) . \end{aligned}$$

This allows removing  $n$ -ary ( $n \geq 3$ ) relations from the rules bodies.

Concerning a  $n$ -ary ( $n \geq 3$ ) relation in the rule head, such as

$$\text{is-in-a-situation}(\text{?user1}, \text{?aSituation}, s)$$

as before, we create a new class

$$\text{is-in-a-situationRelation}(\text{?aSituationReification})$$

with two additional properties

$$\begin{aligned} &\text{is-in-a-situationValue}(\text{?aSituationReification}, \text{?aSituation}) \\ &\text{is-in-a-situationDegree}(\text{?aSituationReification}, s) \end{aligned}$$

then add

$$\begin{aligned} &\text{is-in-a-situation}(\text{?user1}, \text{?aSituationReification}), \\ &\text{is-in-a-situationValue}(\text{?aSituationReification}, \text{?aSituation}) \end{aligned}$$

to the rule body and replace the head with

$$\text{is-in-a-situationDegree}(\text{?aSituationReification}, s)$$

For instance, our fuzzy rule about pre-meeting becomes in SWRL (here, the minimum is implemented as  $\min(a, b) = (a + b - |a - b|)/2$ ), see [11]):

<sup>3</sup> <http://www.w3.org/TR/swbp-n-aryRelations/>

```

min(s1, s2, s3)
← sum(sm, s1, s2)
  subtract(diff, s1, s2),
  abs(absdiff, diff),
  subtract(sd, sm, absdiff),
  divide(s3, sd, 2),

is-in-a-situationDegree(?aSituationReification, s) ← owns(?user1, ?aCalendar),
  contains-as-next(?aCalendar, ?nextEvent),
  is-located-in(?nextEvent, ?aPlace),
  is-scheduled-at(?nextEvent, ?anInterval),
  is-started-at(?anInterval, ?aTime),
  mobility(?user1, ?mobilityReification),
  mobilityValue(?mobilityReification, 'moving'),
  mobilityDegree(?mobilityReification, s1),
  has-current-time(?user1, ?userTime),
  is-before-of(?userTime, ?aTimeReification),
  is-before-ofValue(?aTimeReification, ?aTime),
  is-before-ofDegree(?aTimeReification, s2),
  type(?nextEvent, 'business'),
  Pre-Meeting-on-Movement(?aSituation),
  is-in-a-situation(?user1, ?aSituationReification),
  is-in-a-situationValue(?aSituationReification, ?aSituation),
  min(s1, s2, s)

```

which concludes. We don't go further into the reification procedure as it is pretty common and well-known in the Semantic Web literature.

## 7 Conclusions and Future Work

In this paper, a framework for providing personalized resources based on situation awareness has been proposed, showing situation awareness as a key asset to enable proactive behavior in the system. In particular, the study focuses on combining semantic web standards with fuzzy logic. Domain knowledge is maintained by means of proper ontologies and exploited to infer the current user situations. Inference is carried out by semantic rules which embody fuzzy logic to take the assessment of real-world inaccurate information into account.

We applied the system in two real business cases, in order to assess the effectiveness of our approach. The first evaluation case study concerns a pharmaceutical consultant in typical business situations. In particular, the situations of interest are: (i) *Meeting-Planning*, (ii) *Pre-Meeting*, (iii) *Ongoing-Meeting*, (iv) *Post-Meeting*, (v) *Hospital-Conference*, (vi) *Call-for-Tenders*, and (vii) *Meal*. The second evaluation case study concerns an off-site student, who performs a daily travel to go to university and return. In this case, the identified situations are the following: (i) *Pre-University-Day*, (ii) *Preparing-for-Transportations*, (iii) *Traveling*, (iv) *Studying*, (v) *Attending-Courses*, and (vi) *Meal*. For each case study, by means of a series of interviews, a domain-specific ontology has been added to the upper ontology and the fuzzy linguistic variables have been tuned properly. Hence, a set of 13 rules has been designed for the above business cases. Finally, the system has been tested by considering the events concerning one overall working week of two different consultants and three different students, with 53 and 82 different test events, respectively. The system has been able to recognize the right situations related to all the test events under the different conditions. Further, the differences between the time at which actually the situation occurred and the time at which the system recognized the situation was in the range of few seconds to few minutes. This proves the usefulness of our service recommender which exploits data

collected by different sensors to determine the situations of interest with respect to a recommender based only on the scheduled events stored in the users calendar.

A weakness of the system concerns the design of the linguistic variables, which is domain-specific and does not take into account actual differences among users. We are currently working on improving the possibility of adaptation of the system to the specific user. We are focusing on the exploitation of the user's profile, expressed in terms of user's preferences. Further, we are considering the application of machine learning paradigms for the tuning of the linguistic variables.

## References

1. Ciarabella, A., Cimino, M.G.C.A., Lazzarini, B., Marcelloni, F.: Situation-aware mobile service recommendation with fuzzy logic and semantic web. In: IEEE International Conference on Intelligent Systems Design and Applications (ISDA'09), pp. 1037–1042 (2009)
2. Cingolani, P.: Jfuzzylogic, a java package that implement fuzzy control language (fcl) specification (iec 1131p7), <http://jfuzzylogic.sourceforge.net>
3. Dahchour, M., Pirotte, A.: The semantics of reifying n-ary relationships as classes. In: 5th International Conference on Enterprise Information Systems (ICEIS-02), pp. 580–586 (2002)
4. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: Workshop on the What, Who, Where, When, and How of Context-Awareness, pp. 304–307. ACM Press, New York (2000)
5. Ding, L., Chen, H., Kagal, L., Finin, T.: Public address ontology, <http://daml.umbc.edu/ontologies/ittalks/>
6. Figge, S.: Situation-dependent services a challenge for mobile network operators. *Journal of Business Research* 57(12), 1416–1422 (2004)
7. Fukazawa, Y., Naganuma, T., Fujii, K., Kurakake, S.: A framework for task retrieval in task-oriented service navigation system. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 876–885. Springer, Heidelberg (2005)
8. Goix, L.W., Valla, M., Cerami, L., Falcarin, P.: Situation inference for mobile users: a rule based approach. In: The 2007 International Conference on Mobile Data Management (MDM'07), pp. 299–303. IEEE Computer Society, Los Alamitos (2007)
9. Heinonen, K., Pura, M.: Classifying mobile services. Helsinki Mobility Roundtable. Working Papers on Information Systems, p. 6 (2006)
10. Hong, J., Suh, E.H., Kim, J., Kim, S.Y.: Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications* 36(4), 7448–7457 (2009)
11. Kalman, D.: The maximum and minimum of two numbers using the quadratic formula. *The College Mathematical Journal* 15(4), 329–330 (1984)
12. Klement, E.P., Mesiar, R., Pap, E.: *Triangular Norms*. Trends in Logic - Studia Logica Library. Kluwer Academic Publishers, Dordrecht (2000)
13. Lukasiewicz, T., Straccia, U.: Top-k retrieval in description logic programs under vagueness for the semantic web. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 16–30. Springer, Heidelberg (2007)
14. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics* 6, 291–308 (2008)
15. Luther, M., Fukazawa, Y., Wagner, M., Kurakake, S.: Situational reasoning for task-oriented mobile service recommendation. *The Knowledge Engineering Review* 23(01), 7–19 (2008)
16. Mantyjarvi, J., Seppanen, T.: Adapting applications in handheld devices using fuzzy context information. *Interacting with Computers* 15(4), 521–538 (2003)

17. Sanchez, E., Yamanoi, T.: Fuzzy ontologies for the semantic web. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 691–699. Springer, Heidelberg (2006)
18. Straccia, U.: Uncertainty management in logic programming: Simple and effective top-down query answering. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES 2005. LNCS (LNAI), vol. 3682, pp. 753–760. Springer, Heidelberg (2005)
19. Straccia, U.: Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In: Baroglio, C., Bonatti, P.A., Matuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 54–103. Springer, Heidelberg (2008)
20. Straccia, U.: A minimal deductive system for general fuzzy RDF. In: Polleres, A. (ed.) RR 2009. LNCS, vol. 5837, pp. 166–181. Springer, Heidelberg (2009)
21. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management (UbiComp'04), Citeseer (2004)
22. Sun, G., Chen, J., Guo, W., Liu, K.J.R.: Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs. IEEE Signal Processing Magazine 22(4), 12–23 (2005)
23. W3C.Owl web ontology language reference (February 10, 2004), <http://www.w3.org/TR/owl-ref/>
24. W3C.Public time ontology, <http://www.w3.org/TR/owl-time>
25. W3C.SWRL: a semantic web rule language combining OWL and RuleML, W3C member, <http://www.w3.org/Submission/SWRL/>(submission May 21, 2004)
26. Weißenberg, N., Gartmann, R., Voisard, A.: An ontology-based approach to personalized situation-aware mobile service supply. Geoinformatica 10(1), 55–90 (2006)
27. Zadeh, L.A.: Is there a need for fuzzy logic? Information Sciences 178(13), 2751–2779 (2008)

# Source Selection in Large Scale Data Contexts: An Optimization Approach\*

Alexandra Pomares<sup>1,2,3</sup>, Claudia Roncancio<sup>1</sup>, Van-Dat Cung<sup>1</sup>,  
José Abásolo<sup>2</sup>, and María-del-Pilar Villamil<sup>2</sup>

<sup>1</sup> Grenoble INP, Grenoble, France

<sup>2</sup> Universidad de los Andes, Bogotá, Colombia

<sup>3</sup> Pontificia Universidad Javeriana, Bogotá, Colombia

**Abstract.** This paper presents OptiSource, a novel approach of source selection that reduces the number of data sources accessed during query evaluation in large scale distributed data contexts. These contexts are typical of large scale Virtual Organizations (VO) where autonomous organizations share data about a group of domain concepts (e.g. patient, gene). The instances of such concepts are constructed from non-disjointed fragments provided by several local data sources. Such sources overlap in a non mastered way making data location uncertain. This fact, in addition to the absence of reliable statistics on source contents and the large number of sources, make current proposals unsuitable in terms of response quality and/or response time. OptiSource optimizes source selection by taking advantage of organizational aspects of VOs to predict the benefit of using a source. It uses an optimization model to distinguish the sets of sources that maximize benefits and minimize the number of sources to contact to while satisfying resource constraints. The precision and recall of source selection is highly improved as demonstrated by the tests performed with the OptiSource prototype.

**Keywords:** Large Scale Data Mediation, Source Selection, Combinatorial Optimization.

## 1 Introduction

A Virtual Organization (VO) is a set of autonomous collaborating organizations, called VO units, working towards a common goal. It enables disparate groups to share competencies and resources such as data and computing resources [1]. VOs have evolved to a national and world-wide magnitude [2,3], introducing complex business processes and data sharing contexts.

This work deals with large scale VOs following co-alliance and value-alliance models, where participants provide data related to a group of agreed domain concepts (e.g. patient, client, gene). The VO data context involves a large number of autonomous and distributed sources (tens to thousands), provided by VO units. Sources contain structured data, typically fragmented horizontally

---

\* This research was supported by the project Ecos-Colciencias C06M02.

and vertically w.r.t. the domain concepts of the VO. Therefore sources may be incomplete intentionally (schema) and extensionally (contents). Moreover, participant interactions produce logical relationships between sources generating uncontrolled situations of data overlapping and data replication. In most cases, query processing becomes difficult and heavily resource dependent because the mediation level does not have precise knowledge of the contents of each source. Considering the large scale context, a tradeoff between the number of answers to obtain and the number of sources (potential contributors) to access becomes crucial, especially while not compromising the accuracy of query processing.

Current source selection approaches are unsuited for this kind of contexts because they do not significantly reduce the number of sources to access when their schemas are similar [4,5,6,7,8] like is the case in VOs. Others do not take into account source overlapping leading to redundant answers when used in VOs [9,10,11,12]. Proposals such as [13,8] do consider overlapping, but assume the existence of some metadata that is difficult to obtain and maintain in large VOs.

**Contribution.** This paper presents OptiSource, a new approach that addresses the source selection problem<sup>1</sup> as a decision making problem within a complex environment. OptiSource predicts the benefit of using a source during query evaluation, and optimizes the assignment of query predicate evaluation to sources. It uses a combinatorial optimization model to distinguish the sources that maximize the benefit and minimize the number of sources to contact to while satisfying processing resource constraints. OptiSource is well suited when sources are numerous, when the instances that match the query are fragmented and distributed in several sources, and when not exhaustive<sup>2</sup> answers are acceptable. A prototype of OptiSource is operational. This paper reports test results, which demonstrate the improvement of the precision and recall of source selection.

**Outline.** In the following, Section 2 presents the context of the problem and related works. Section 3 presents OptiSource. Section 4 presents the combinatorial optimization model used in OptiSource. Section 5 presents the OptiSource prototype and its evaluation. Finally, Section 6 concludes this paper.

## 2 Data Source Selection: Context and Related Works

The selection of the right set of data sources to evaluate a query is one of the crucial steps to assure an efficient query evaluation on large scale distributed systems. Considering all the available sources is of course unsuitable because the number of possible query plans grows rapidly as the complexity of the query and the number of sources increase. This section first points out the main characteristics of VO's data context and then discusses related works.

### 2.1 Data Context in Large Scale Virtual Organizations

In large scale VOs, VO units (participants) are distributed in a wide area network. They share preexisting heterogeneous sources following a federated approach and

<sup>1</sup> Henceforth, source and data source are going to be used interchangeably.

<sup>2</sup> In these contexts a sub set of the available answers is enough.



Fig. 1. VDO Patient

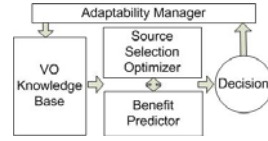


Fig. 2. OptiSource Components

according predefined cooperation agreements [14]. Shared data are related to these concepts (e.g. patient, client, gen), henceforth to be referred to as Virtual Data Objects (VDO). An instance of a VDO may not exist completely in one single source, but may be distributed across several sources. The creation of such an instance requires the integration of data issued by several sources.

A VDO is defined by a set of classes and related properties. Figure 1 presents the definition of the VDO Patient. It inherits from the class *Person*, and is linked to classes *Medical Act* and *Affiliation*. *Person* is linked to the class *Demographic Data*. VDO Queries specify the set of required conditions of the instances.

**Data characteristics.** Business processes involving the VO units have a significant impact on VDO instances distribution. They introduce data source fragmentation and overlapping without providing a global description of this situation to the system. The following are the most important of these aspects, which affect source selection:

**-Non disjoint vertical fragmentation.** VDO properties are fragmented between different sources. However, fragments are not disjointed which means that a VDO’s property may be provided by several sources.

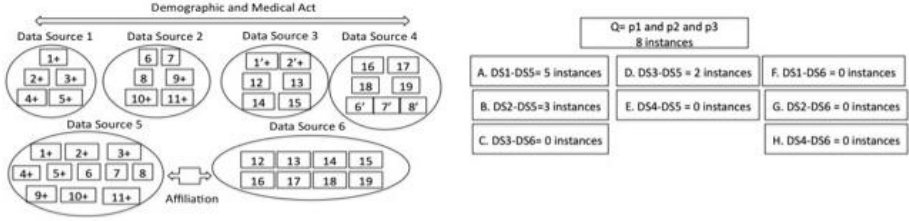
**-Non disjoint horizontal fragmentation.** Several sources may provide the same fragment or instance of a VDO, but may also provide different groups of properties and values related to the same VDO.

**-Uncertainty on data location.** Instances of VDOs follow uncertain pattern distribution. Indeed, two instances may be fragmented and distributed in completely different ways. Such fragmentation is not intentionally designed, but results from preexisting data sources and various business processes.

**-Fuzzy copies.** A phenomenon of inaccurate copies<sup>3</sup> may even appear, particularly if there are integration efforts of VO unit’s subgroups. In a health VO, instances of *Patient* may be replicated due to the mobility of people whereas batch replication may be introduced by reporting governmental politics.

**Data distribution scenario.** Figure 3 illustrates a typical scenario of data distribution in a health VO. Six sources (represented by ovals) provide instances of VDO Patient (represented by rectangles). None of the sources includes all the properties of the class *Patient* (vertical fragmentation) and no source contains all the instances of *Patient* (horizontal fragmentation). Additionally, vertical

<sup>3</sup> Fuzzy copies are copies that do not have an explicit protocol of consistency.



**Fig. 3.** Health VO Data Context

fragments are not disjoint because multiple sources (DS1 to DS4) provide the properties of the class *Demographic Data* (e.g. gender, age) and of the class *Medical Act* (e.g. diagnosis), and multiple sources (DS5 and DS6) provide the properties of the *Affiliation* class. Horizontal fragments are not disjoint neither. Instances 1 and 2 and instances 6 to 8 are contained in two sources that provide the same properties. Although instances 1 and 2 provided by DS1 and DS3 have the same *id*, it is not possible to establish in advance whether or not they are replicas due to the autonomy of sources. This occurs in a similar way for instances 6,7 and 8 in DS2 and DS4.

Equation (II) presents the logical query plan that provides a complete answer for a query  $Q$  asking for instances of *Patient* that match a predicate  $p$  with three conditions  $p1$ ,  $p2$  and  $p3$ . These conditions are related to the properties of classes *Demographic Data*, *Medical Act*, and *Affiliation* respectively (e.g.  $\text{hasGender}=\text{F}$  and  $\text{hasDiagnosis}=\text{Cancer}$  and  $\text{isAffiliatedtoSS}=\text{true}$ ). This plan does not discard any data source because the query planner does not know how the distribution of instances that match the query is.

$$\text{Plan}(VDO_{\text{patient}}, Q(p1, p2, p3)) = DS1 \bowtie DS5 \sqcup DS2 \bowtie DS5 \sqcup DS3 \bowtie DS5 \sqcup DS4 \bowtie DS5 \sqcup DS1 \bowtie DS6 \sqcup DS2 \bowtie DS6 \sqcup DS3 \bowtie DS6 \sqcup DS4 \bowtie DS6 \quad (1)$$

However, assuming instances that match these conditions are 1,2,3,4,5,9,10 and 11, it can be noticed that  $DS1 \bowtie DS5$  provides more than 60% of the answers (instances 1 to 5). If a complete answer is required, adding  $DS2 \bowtie DS5$  will be enough. The problem is how to identify during query planning the sources that best answer the query without knowing exactly which instances are contained by each source. This is the main issue addressed by this paper.

## 2.2 Related Works

Even though multi-sources querying has been intensively studied for more than 15 years, none of the available strategies for query planning are well suited for large scale VO data contexts. In the first generation of integration systems for structured sources like Information Manifold [4], TSIMMIS [5], DISCO [6] and Minicon [7], source selection is performed according to the capabilities of sources (e.g. the number of required conditions, the attributes that can be restricted). Although such proposals are efficient in several contexts, they have little effect on VOs where the processing capabilities of sources are alike.



Another group of algorithms like iDrips and Streamer [8] could be used in VO contexts. They propose to put in order query plans according to an utility function. The coverage plan is an example of this function that measures the number of instances provided by each query plan. Even though the coverage considers the extensional overlapping between sources, the algorithms do not specify how this information is obtained before the query execution; hence, the utility function directly affects the behavior of the algorithms (see Section 5).

Other proposals like QPIAD [15] and Quality Oriented [16] use detailed statistics on data source quality to reduce the number of possible query plans and therefore sources. Unfortunately, such statistics are difficult to obtain in VOs due to the large number of sources and the high volume of contents. Similarly, the proposal to improve the selection of navigational paths between sources presented in [13] assumes a previous knowledge of the instances in each source and how they are linked together. These detailed metadata is not available in VOs. The proposal of Balancing Providers [17] is well suited for environments where finding one possible query plan is enough to obtain the required set of answers. Nevertheless, it is not appropriate for VO where data sources are incomplete, and several query plans are therefore necessary to obtain a complete answer.

Proposals using a P2P approach (PIER [9], PIAZZA [10], EDUTEL-LA [11], SomeWhere [12]) manage numerous sources. However, as their rewriting and source selection strategies do not take into account source overlapping, they would lead to high redundancy in the answers when used in large scale VOs.

### 3 OptiSource: A Decision System for Source Selection

This section presents OptiSource, our proposal to handle source selection in large scale VOs. OptiSource reduces the queried sources to the most profitable w.r.t. the query. It takes advantage of the organizational notion of VOs to reduce the uncertainty on data location, and to differentiate sources, even if they overlap.

#### 3.1 Queries and Joining Sets

The source selection process must decide which sources will evaluate each predicate condition (or simply condition) and which source answers are going to be integrated for creating the required instances. Knowing that sources can have common instances, the process must decide for each condition the minimal set of sources required to obtain the maximum number of instances that match the condition, while avoiding the contact of unnecessary sources. VO query processors require an appropriate source selection strategy able to identify joining sets (see Definition 1) that will provide higher benefits to the query.

**Definition 1. *Joining Set.*** *Given a set of data sources and a query, a joining set is a group of data sources able to evaluate all the conditions of the query predicate and whose join operation between the data sources it comprises will not produce an empty set of instances.*

The input of the source selection process are  $n$  available data sources  $DS_i$ , their mapping (or view)  $M_i$  to the VDO (e.g. *Person*, *MedicalAct*) and the conditions  $p_j$  involved in the query predicate:

$$Input = [(DS_1, M_1), (DS_2, M_2), \dots, (DS_n, M_n)], [p_1, p_2, \dots, p_m].$$

The output is the collection of joining sets:  $Output = (JS_1, JS_2, \dots, JS_p)$ ,

where  $JS_i = (DS_k, p_1), \dots, (DS_l, p_m), 1 \leq k, l \leq n$ . A couple  $(DS_k, p_l)$  means that the data source  $DS_k$  will evaluate the condition  $p_l$ . A source can evaluate several conditions  $p_j$  and a condition  $p_j$  can be evaluated by several sources.

OptiSource will produce the appropriate joining sets and optimize query processing by using a dynamically updated knowledge base to predict the benefit of using particular sources. This knowledge will act as another input for the source selection process.

### 3.2 Components of OptiSource

The main components of OptiSource (Figure 2) are the VO knowledge base, the benefit predictor, the source selection optimizer, and the adaptability manager. The VO knowledge base maintains knowledge facts and events related to the data context of the VO. This includes intentional (schema) and extensional information (contents) of sources. The knowledge is represented as an ontology in OWL [18] with three initial classes: **VOUnit**, **VOResource** and **VODomainConcept**. These classes are specialized and related to better reflect the evolution of the VO data context. **VOUnit** class represents the participants of the VO. An instance of **VOUnit** can be atomic or composite. In the latter case, it represents a temporal or a permanent group of atomic VO units working together around a specific collaboration process. **VOResource** class represents the physical or logical resources (like data sources) provided by VO units. Finally, **VODomainConcept** includes the subclasses that describe the domain of the VO. For a health VO for instance, initial classes are *Patient*, *Medical Act*, and so on.

The Benefit Predictor uses VO knowledge base to predict the benefit of using a source to evaluate the conditions of a query predicate. The Source Selection Optimizer determines the best assignment of data sources for each condition and produces the joining set that maximizes the benefit, in terms of the number of resulting instances. The Adaptability Manager is in charge of updating the VO knowledge base according to the findings after the execution of a query.

### 3.3 Benefit Prediction Using Data Source Roles

OptiSource estimates the benefit (see Definition 2) of using a source in the evaluation of a query to identify the level of relevance it will have in the query.  $DSa$  is more relevant than  $DSb$  if it provides more instances that match the query predicate, even if both of them have the same schema. In order to predict the benefit we use the available knowledge about the extension (contents) of sources. To obtain this knowledge we work under the assumption that the organizational

dimension of VOs allows to relate the role that VO units play in the VO with the contents of the sources they provide. The more complete and reliable the extensional knowledge, the more accurate the measurement of the benefit.

**Definition 2. Data Source Benefit.** *Given a Query  $Q$  with a query predicate  $P$  with conditions  $[p_1, p_2, \dots, p_m]$  the benefit of using a data source  $DS_i$  to evaluate  $Q$  is a variable that measures the degree of contribution of  $DS_i$  in terms of instances that match one or more  $p_j \in P, 1 \leq j \leq m$ .*

**Data source's roles.** In a health VO, for instance, two data sources  $DSa$  and  $DSb$  provide information about procedures performed on patients. Let's assume that  $DSa$  is provided by a VO unit specialized on cancer whereas  $DSb$  is provided by a hospital specialized on pediatric procedures. In this case, the  $DSa$  may be considered specialist of patients with cancer (vo:Patient, hasDiagnosis vo:Cancer) whereas the  $DSb$  is children specialist (vo:Patient, hasAge  $\leq 15$ ). A source can therefore play different **roles** as a contributor of instances of a VDO (e.g. vo:Patient) verifying some predicate conditions (e.g. hasAge  $\leq 15$ ). Roles reflect the ability of sources to resolve conditions. Given the analysis of the roles played by VO units in the business processes of the VO, we propose the following three roles for their data sources: *authority*, *specialist* and *container*.

The definition of each source role is described in Definition 3, 4 and 5. In these definitions all the instances of  $VDO_j$  stored in data source  $DS_i$  are noted  $ext(DS_i, VDO_j)$ .  $U$  designates the data sources participating in the VO. All the instances of  $VDO_j$  available in the VO are noted  $ext(U, VDO_j)$ . The subset of  $ext(DS_k, VDO_j)$  corresponding to the instance that verifies a condition  $p$  is denoted  $ext(DS_k, VDO_j)^p$  and  $card()$  is the cardinality function.

**Definition 3. Authority Role.** *A data source  $DS_i$  plays an authority role w.r.t. a condition  $p$  in a query on  $VDO_j$  iff it stores all the instances of  $VDO_j$  available in  $U$  that match  $p$ .  $IsAuthority(DS_i, VDO_j, p) \implies ext(U, VDO_j)^p \subset ext(DS_i, VDO_j)$*

**Definition 4. Specialist Role.** *A data source  $DS_i$  plays a specialist role w.r.t. a condition  $p$  in a query on  $VDO_j$  iff most instances of  $VDO_j$  stored in  $DS_i$  match  $p$ .  $IsSpecialist(DS_i, VDO_j, p) \implies card(ext(DS_i, VDO_j)^p) \geq card(ext(DS_i, VDO_j)^{\neg p})$*

**Definition 5. Container Role.** *A data source  $DS_i$  plays a container role w.r.t. a condition  $p$  in a query on  $VDO_j$  iff  $DS_i$  contains at least one instance of  $VDO_j$  that matches  $p$ .  $IsContainer(DS_i, VDO_j, p) \implies ext(U, VDO_j)^p \cap ext(DS_i, VDO_j)^p \neq \emptyset$*

Data source roles are registered as extensional facts in the VO knowledge base and can be acquired using three approaches: (a) Manually (e.g. expert's or DBA's definition of knowledge), (b) Interpreting the execution of processes, (c) Automatically extracting it from sources of knowledge. In [19] we present strategies for the last two approaches.

<sup>4</sup> This extension contains the object identifiers.

**Benefit model.** In order to predict the benefit of a source  $DS_i$  w.r.t. a predicate condition  $p$  the model uses Formula 2. This formula relates the role of the data source and the relative cardinality of the data source. The intention is to take into account the expected relevance of the source, given by the role factor, with the maximum number of instances that a data source may provide (in the best case). This combination of knowledge was necessary because sources that play a specialist role could be less important than sources that play container role if the number of instances that the first group may provide is considerably lesser.

The  $RoleFactor()$  (Formula 3), returns a value between  $[0,1]$  indicating the most important role that a source may play in the query.  $ContainerFactor$ ,  $SpecialistFactor$  and  $AuthorityFactor$  are constants reflecting the importance of the role.

$$Benefit(DS_i, VDO_j, p) = \frac{RoleFactor(DS_i, VDO_j, p) * card(ext(DS_i, VDO_j))}{\max\{card(ext(Dk, VDO_j)), \forall Dk \text{ in } U \text{ where } RoleFactor(DSk, VDO_j, p) > 0\}} \quad (2)$$

$$RoleFactor(DS_i, VDO_j, p) = \max((IsContainer(DS_i, VDO_j, p) * ContainerFactor), (IsSpecialist(DS_i, VDO_j, p) * SpecialistFactor), (IsAuthority(DS_i, VDO_j, p) * AuthorityFactor)) \quad (3)$$

Although Formula 2 would be more accurate if it uses the cardinality taking into account the query predicate  $-card(ext(DS_i, VDO_j)^p)$ -, the exact number of instances that satisfy a predicate is not available in VOs.

**Creating joining sets.** To predict the set of sources that will not produce empty joins, the VO knowledge base is queried for obtaining the individuals of the composite VO units. This decision was made under the assumption that atomic VO units belonging to the same composite VO unit have more probability of containing the same group of instances. However, the creation of joining sets can use other type of rules to identify when two or more sources may share instances. For example, if two VO units are located in the same region, the probability that their data sources share instances may increase. Similarly, the fact that two VO units participate in the same VO process increases this probability.

Algorithm 1 uses the rule of composite units to create joining sets. The objective is to group together sources whose join will not produce an empty set and that are able to evaluate all the conditions of the query. It first [1] obtains the data sources of the composite units and creates one set for each of them. If there are redundant sets, the algorithm removes the set with fewer atomic units. Then, in [2], it validates the completeness of a set. A set is complete if the data sources it contains are able to evaluate all the conditions of the query. In [3] it extracts data sources from complete sets and [4] removes those incomplete sets whose data sources exist in the complete sets. If it is not the case, the algorithm gets the query conditions that are not already evaluated on each incomplete set [5] and completes these sets finding [6] the data source with higher role able to evaluate the missing conditions, from the complete sets.

**Algorithm 1.** Joining Sets Creation

---

```

Input: Q, CompositeUnits{ID1(DSi,...,DSj),...,IDm(DSk,...,DSm)},
      QRoles(DS1:role,...,DSn:role)
Output: JoiningSets{JS1(DSp,...,DSq),...,JSn(DSt,...,DSv)}
Begin
[1] initalSets{} = CreateInitialSets(CompositeUnits,QRoles)
    incompleteSets{} = {}    completeSets{} = {}
    ForAll (set in initialSets)
[2]   If (isComplete(set,Q)){ add(completSets,set)}
      Else{ add(incompleteSets,set)}
    If (size(incompleteSets) > 0){
[3]   com{} = getDataSources(completeSets)//The data sources of complete sets
    ForAll (set in incompleteSets){
      inc{} = getDataSources(set)//The data sources of one incomplete set
      If ( contains(com, inc) // com contains inc
[4]     remove(set, incompleteSets)
[5]   Else{ conditions{} = getMissingEvaluation(set, Q)
      ForAll (cond in Conditions){
[6]     dataSource = getHighestRole(com,cond)
      addDataSource(set,dataSource)}
      add(completeSets,set)}
    }}
  Return(completeSets)
End

```

---

The output of the benefit predictor is the set of joining sets. According to Definition [1](#) a joining set must contain at least one data source to evaluate each condition of the query predicate. However, if more than one data source may evaluate the same condition, should this condition be evaluated at all data sources or it would be possible to choose only one of them. The optimizer component, described in the next section, helps to make the right decision.

## 4 Optimizing Source Selection

A joining set may have several sources able to evaluate a query condition with different values of benefit w.r.t. the condition. Querying all of them is neither necessary nor convenient, considering the cost of querying a source. The proposal is to see the problem of deciding which conditions are evaluated by which sources as an assignment problem [\[20\]](#) subject to resource constraints that assigns a condition to a source to maximize the general benefit. We propose a mathematical model that receives the results of the benefit predictor component and the processing capacities of sources, if they are known, and determines the best assignment. Although this model has been created for source selection in VOs, it can be used in any distributed query processing system.

### 4.1 Source Selection as an Assignment Problem

In the source selection problem, there are a number of conditions to assign to a number of sources. Each assignment brings some benefits, in terms of response

quality, and consumes some resources of the sources. The objective is to maximize the benefits using the minimum number of sources while satisfying the resource constraints. From the query point of view, one condition is assigned to one main source as *main assignment*, but this condition could also be evaluated as *secondary assignment* in parallel on other sources which have been selected by other conditions. The reason is that once a source is queried, it is better to evaluate there a maximum possible number of conditions, expecting to reduce the number of sources and the cost of transmitting instances that are not completely evaluated. Indeed, we have to deal with a bi-objective combinatorial optimization problem subject to semi-assignment and resource constraints. In practice, we choose to control the objective of minimizing the number of selected sources by converting it into a constraint. By default, the number of selected sources is limited to the number of conditions.

## 4.2 Mathematical Model

Given the set of sources  $I = \{1, \dots, m\}$  and the set of predicate conditions  $P = \{1, \dots, n\}$  of a query over a VDO, the input data are as follows:

- $Ben_{i,p}$ : benefit of assigning condition  $p$  to source  $i$ ,  $\forall i \in I, \forall p \in P$ , as explained in formula 2 in Section 3.3,  $Ben_{i,p} = Benefit(DSi, VDOj, p)$  for a given  $VDOj$ ;
- $MaxRes_i$ : processing resource capacity of source  $i$ ,  $\forall i \in I$ ;
- $Res_{i,p}$ : processing resources consumed in assigning condition  $p$  to source  $i$ ,  $\forall i \in I, \forall p \in P$ ;
- $MaxAssig_i$ : maximal number of condition assignments for source  $i$ ,  $\forall i \in I$ .

The decision variables are:

- $x_{i,p}$  are 0-1 variables that determine whether source  $i$  has (=1) or not (=0) been selected as a main source to evaluate the condition  $p$ ,  $\forall i \in I, \forall p \in P$ .
- $y_i$  are 0-1 variables that turn to 1 when the source  $i$  is selected,  $\forall i \in I$ .
- $assig_{i,p}$  are 0-1 variables that determine whether a condition  $p$  is assigned to source  $i$  (=1) or not (=0),  $\forall i \in I$ . These variables represent the final assignment of conditions to sources. The  $x_{i,p}$  variables indicate the main assignments while the  $assig_{i,p}$  variables indicate all the main and secondary assignments.

The mathematical program can be formulated as follows:

$$\max \sum_{p=1}^n \sum_{i=1}^m Ben_{i,p} * (x_{i,p} + assig_{i,p}), \quad (4)$$

subject to:

$$\sum_{i=1}^m x_{i,p} = 1, \forall p \in P; \quad (5) \quad \sum_{i=1}^m y_i \leq k; \quad (6) \quad \sum_{p=1}^n x_{i,p} \geq y_i, \forall i \in I; \quad (7)$$

$$\sum_{p=1}^n Res_{i,p} * assig_{i,p} \leq MaxRes_i, \forall i \in I; \quad (8) \quad \sum_{p=1}^n assig_{i,p} \leq MaxAssig_i, \forall i \in I; \quad (9)$$

$$x_{i,p} \leq assig_{i,p}, \forall i \in I, \forall p \in P; \quad (10) \quad assig_{i,p} \leq y_i, \forall i \in I, \forall p \in P. \quad (11)$$

Constraint (5) ensures that any condition  $p$  is assigned to one main source. Constraint (6) limits the total amount of queried sources to  $k$ , we take  $k=n$  by default to start the optimization process. This constraint is somehow redundant with constraint (7) which prevents to select a source if no condition is assigned to it, but in practice one could reduce  $k$  in constraint (6) to control the minimization of the number of selected sources. Constraints (8) and (9) ensure that all the main and secondary assignments of conditions do not exceed neither the processing resource capacities nor the maximum number of possible assignments per source. Finally, coupling constraints (10) and (11) indicate respectively that the main assignments should be in all main and secondary assignments as well, and that a source  $i$  is selected when at least one condition  $p$  is assigned to it.

The resolution of this model provides the selected sources in the variables  $y_i$  and all the main and secondary condition assignments in the variables  $assign_{i,p}$ . The joining of the results provided by each source are the instances required by the user. If the number of instances obtained are not enough to satisfy user requirements, the query planner will use the model to generate a new assignment with the remaining sources (i.e. those that were not in  $y_i$ ).

## 5 Implementation and Validation

The objective is to measure the recall and precision provided by OptiSource. Section 5.1 presents the evaluation using different data contexts and levels of knowledge about sources. Section 5.2 compares OptiSource to related works.

### 5.1 OptiSource Experiments

Our prototype of OptiSource is written in Java. The knowledge base was implemented in OWL [18]. The optimization model was written in GNU MathProg modeling language and is processed by the GNU Linear Programming Kit (GLPK) [21]. We also validated our optimization model performance in CPLEX 10.2. Queries of VDOs are accepted in SPARQL [22].

We define the knowledge base of a VO in the health sector. Metadata are generated at different levels of the knowledge base using a developed generator of metadata. This generator describes the intentional knowledge of sources, the known roles of each source and possible replication of instances.

Experiment scenarios vary in five dimensions: the number of sources, the level of knowledge of sources, the number of composite VO units, the relationships between sources and the query. The experiments are focused on measuring three metrics: Extensional Precision, Extensional Recall, and Extensional Fall-Out. To obtain these metrics, we measure the following variables:

**S**: set of sources available on the VO

**-R**: set of not relevant sources for Q

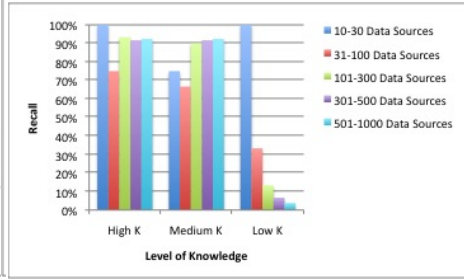
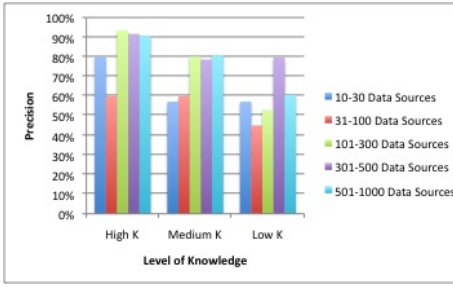
**Extensional Precision**:  $|A \cap R|/A$

**Extensional Fall-Out**:  $|A \cap -R|/-R$

**R**: set of relevant sources for Q

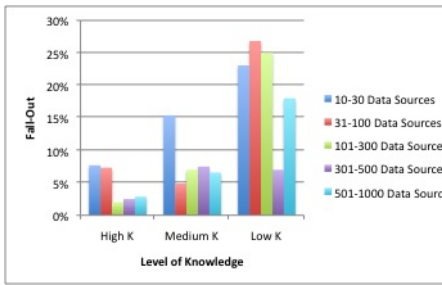
**A**: set of sources selected by OptiSource.

**Extensional Recall**:  $|A \cap R|/R$

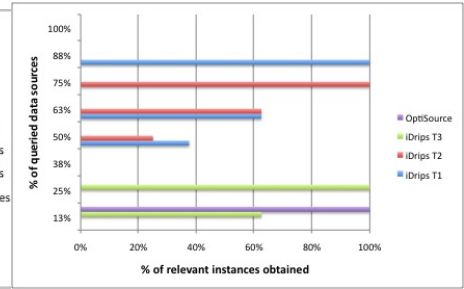


**Fig. 4.** Precision evaluation changing the data context and the level of knowledge

**Fig. 5.** Recall evaluation changing the data context and the level of knowledge



**Fig. 6.** Fall-Out evaluation changing the data context and the level of knowledge



**Fig. 7.** OptiSource vs. iDrips

**Experiment results.** Figures 4, 5, 6 present respectively the precision, recall and fall-out, varying the level of knowledge (K) and the number of sources. Low level means the knowledge base only contains the intentional knowledge of sources (e.g. can evaluate the diagnosis). The medium level means it also knows a group of roles of sources, but they are related to higher classes of the knowledge base. For instance, sources have related roles to *cancer* and queries ask for *kidney cancer*. High level indicates that roles are related to more specific classes (e.g. roles related to kidney cancer).

The experiments show that even if a high level of knowledge is desired, OptiSource precision and recall have a good behavior with a medium level. This is probably due to the fact that a medium level is enough to discard a large group of sources and to direct the queries to sources with higher probabilities of having matching instances. In contrast, the fall-out metric improves (i.e. is lower) when the level of knowledge increases. However, as it can be noticed in Figure 6 even with a lower level of knowledge the fall-out can be low due to the existence of sources that have a considerably higher benefit in comparison with the majority of sources, as is the case in the group of 301 and 500 sources. Tests also show that for contexts with fewer sources, the improvement in precision according to



the level of knowledge is not significant. This validates our assumption that in small data contexts great efforts in query planning for reducing queried sources are not necessary. On the other hand, the improvement observed when there is a large number of sources led us to conclude that OptiSource is especially useful under these circumstances. Finally, the experiments show that in the worst case OptiSource selects the same set of sources that traditional strategies of query rewriting select because it only considers the intentional views.

## 5.2 Comparison with Related Works

Although OptiSource can not be directly compared with any of the proposals in the literature because they worked under different assumptions, we evaluated the implications of using three available strategies (QPIAD [15] and Navigational Paths [13], iDrips [8]) in VOs contexts.

The strategy of rewriting of QPIAD [15] allows the query processor to return certain and “possible” certain answers from autonomous databases. The first type are answers that match all the query predicate; the second type are the answers that probably match the query, but have one condition that can not be evaluated because the required value is *null*. QPIAD generates and uses detailed statistics on source’s quality to reformulate a query according to the characteristics of each source that is able to evaluate the predicate. Even though the principle of QPIAD will be very attractive for use in VOs whose sources have quality problems, the strategy could not scale up when the number of sources is large. To prove this we will consider a VO with 20 sources, which are able to evaluate a query  $Q$  in terms of the schema. We supposed that in average each source provides 500 certain answers. For each source 50 query reformulations are required to obtain the possible answers. This means that 1020 queries are executed including the original  $Q$  to obtain certain answers from each source.

Another related strategy is the proposal to improve the selection of navigational paths between biological sources [13]. It assumes the availability of a graph describing the objects contained in the sources and their relationships. The objective is to find the path with best benefit and lower cost given an initial and a final object class. Applying this proposal to VOs we found that it is possible to organize sources in a graph. Each node could contain sources with equivalent intentional fragments of VDO, and the relationships between sources that contain complementary fragments could be the links between nodes. Using this graph it will be possible to select the most “profitable” sources for a query; however, it is not clear how the benefit of each path is obtained and how the overlapping between each path can be calculated. This lack could be combined with our strategy using the roles of sources to compute the potential benefit of a path. Thus, the prediction model of OptiSource can be used to compute the ratios between paths used in this proposal.

Finally, we compared OptiSource with the iDrips algorithm. iDrips works under the assumption of source similarity that allows the query processor to retrieve the most relevant plans efficiently. It proceeds in two phases. The first phase groups similar sources and creates abstract plans that are able to solve

all the predicates. Each abstract plan represents a set of concrete plans with an utility. This utility is an interval that contains the utility of the concrete plans of the set. The second phase obtains query plans in decreasing order of utility selecting only the dominant plan in each iteration. A dominant abstract plan has at least one concrete plan whose utility is not less than the utility of all concrete plans in other abstract plans.

The comparison with iDrips is focused on measuring the recall in terms of relevant instances obtained vs. the number of sources queried. We did not measure precision because iDrips and OptiSource always return correct instances. In order to make comparable the measurements of recall in OptiSource and iDrips, we evaluate in each iteration of iDrips the number of relevant instances until it finds the complete set of relevant instances. We also measure the number of queried sources in each iteration. In the case of OptiSource, we measure the relevant instances obtained after the execution of subqueries of the joining sets.

We used the plan coverage as the utility measure for iDrips. Given a query  $Q\{p(p_1, \dots, p_m)\}$  with  $m$  conditions, and a plan  $P\{DS_1, \dots, DS_n\}$  where  $DS_i$  are the sources that will be queried on the plan, the formulas used to compute plan coverage are the following:

$$Cov(P) = \min(ext(DSi, VDOj)) \text{ where } DS_i \in P, \quad (12)$$

$$Cov(P) = \min(overlap(DSi, DS_k)) \text{ where } DS_i \text{ and } DS_k \in P, \quad (13)$$

$$Cov(P) = \frac{ext(DSi, VDOj)^p - (ext(DSk, VDOj)^p \dots ext(DSl, VDOj)^p)}{ext(U, VDOj)^p} \quad (14)$$

where  $DS_i, DS_k$  and  $DS_l \in P$  and  $DS_k \dots DSl$  have been previously queried.

The first group of experiments are run in a context where sources are extensionally and intentionally overlapped, and a few number of sources contain a large number of relevant instances; even though instances are distributed largely along the VO sources. In other words, there are sources specialized in one or more of the conditions. Consequently, querying them is enough to obtain almost all the relevant instances. Using the aforementioned utility measures for iDrips. Figure 7 illustrates the differences on recall between iDrips and OptiSource. The y axis shows the % of queried sources from the total number of relevant data sources. On the other hand, the x axis illustrates the % of instances obtained.

The results demonstrate the effectiveness of OptiSource in this type of contexts where it only queried 13% of relevant sources to obtain 100% of relevant instances. On the contrary, iDrips required to query 25% of relevant sources to obtain the 100% of relevant instances in the best case (using (14)). Figure 7 shows that the number of sources queried by OptiSource is considerable lower than those queried using (12) and (13) of the plan coverage. Although this could be inferred considering the difference on the level of knowledge, it allows us to conclude that solutions for ranking queries cannot be applied to VOs without using an utility function able to represent source relationships and source probability of having query relevant instances. Nonetheless, even with strong utility

functions, the absence of replication knowledge prevents discarding sources that do not contribute with new instances. This lack affects iDrips when we use (14).

## 6 Conclusions and Future Work

Source selection in large scale VO data contexts is a complex decision problem. It requires finding the minimal sets of sources for each condition and performing a final intersection operation to avoid empty joins. Current strategies of query planning are not focused on producing query plans under these circumstances, generating redundant and unnecessary plans when they are applied to VOs. Our proposal to solve this problem is a decision system called OptiSource. It uses extensional characteristics of sources to predict the benefit of using them during query evaluation. OptiSource produces joining sets using a combinatorial optimization model that establishes which sources will evaluate each condition of the query. Although OptiSource was created for VOs, it can be used in any distributed query processing system during the planning phase. A prototype of OptiSource has been developed to evaluate its precision. Experiments have demonstrated that OptiSource is effective to reduce the contact of unnecessary sources when data contexts involve extensional and intentional overlapping. They have also showed that OptiSource is an evolving system whose precision grows according to the degree of knowledge of sources, but does not require a high level of knowledge to have good levels of precision and recall.

Future work will be focused on performance evaluation of the approach. We are also interested on applying data mining techniques on logs of query executions to identify the role of sources. Additionally, we are going to improve the prediction of the benefit using live statistics of sources that have a higher role.

## References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.* 15, 200–222 (2001)
2. NEESGrid: Nees consortium (2008), <http://neesgrid.ncsa.uiuc.edu/>
3. BIRN: Bioinformatics research network (2008), <http://www.loni.ucla.edu/birn/>
4. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: *VLDB 1996*, Bombay, India, pp. 251–262 (1996)
5. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The tsimmis approach to mediation: Data models and languages. *Journal of Intelligent Information Systems* 8, 117–132 (1997)
6. Tomasic, A., Raschid, L., Valduriez, P.: Scaling access to heterogeneous data sources with DISCO. *Knowledge and Data Engineering* 10, 808–823 (1998)
7. Pottinger, R., Halevy, A.Y.: Minicon: A scalable algorithm for answering queries using views. *VLDB J.* 10, 182–198 (2001)
8. Doan, A., Halevy, A.Y.: Efficiently ordering query plans for data integration. In: *ICDE '02*, Washington, DC, USA, p. 393. IEEE Computer Society, Los Alamitos (2002)

9. Huebsch, R., Hellerstein, J.M., Lanham, N., Loo, B.T., Shenker, S., Stoica, I.: Querying the internet with pier. In: VLDB 2003, pp. 321–332 (2003)
10. Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X.L., Kadiyska, Y., Miklau, G., Mork, P.: The piazza peer data management project. SIGMOD Rec. 32, 47–52 (2003)
11. Nejdil, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: Edutella: a p2p networking infrastructure based on rdf. In: WWW '02, pp. 604–615. ACM, New York (2002)
12. Adjiman, P., Goasdoué, F., Rousset, M.C.: Somerdfs in the semantic web. J. Data Semantics 8, 158–181 (2007)
13. Bleiholder, J., Khuller, S., Naumann, F., Raschid, L., Wu, Y.: Query planning in the presence of overlapping sources. 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. 811–828. Springer, Heidelberg (2006)
14. Venugopal, S., Buyya, R., Ramamohanarao, K.: A taxonomy of data grids for distributed data sharing, management, and processing. ACM Comput. Surv. 38, 3 (2006)
15. Wolf, G., Khatri, H., Chokshi, B., Fan, J., Chen, Y., Kambhampati, S.: Query processing over incomplete autonomous databases. In: VLDB, pp. 651–662 (2007)
16. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Inf. Syst. 29, 583–615 (2004)
17. Quiané-Ruiz, J.A., Lamarre, P., Valduriez, P.: Sqlb: A query allocation framework for autonomous consumers and providers. In: VLDB, pp. 974–985 (2007)
18. Horrocks, I.: Owl: A description logic based ontology language. In: CP, pp. 5–8 (2005)
19. Pomares, A., Roncancio, C., Abasolo, J., del Pilar Villamil, M.: Knowledge based query processing. In: Filipe, J., Cordeiro, J. (eds.) ICEIS. LNBIP, vol. 24, pp. 208–219. Springer, Heidelberg (2009)
20. Hillier, F.S., Lieberman, G.J.: Introduction to Operations Research, 8th edn. McGraw-Hill, New York (2005)
21. Makhorin, A.: Gnu project, gnu linear programming kit (2009), <http://www.gnu.org/software/glpk/>
22. Eric Prud, A.S.: Sparql query language for rdf (2007), <http://www.w3.org/tr/rdf-sparql-query/>

# Semi-automatic Discovery of Web Services Driven by User Requirements

María Pérez, Ismael Sanz, Rafael Berlanga, and María José Aramburu

Universitat Jaume I, Spain  
{matalan,isanz,berlanga,aramburu}@uji.es

**Abstract.** Current research in domains such as the Life Sciences depends heavily on the integration of information coming from diverse sources, which are typically highly complex and heterogeneous, and usually require exploratory access. Web services are increasingly used as the preferred method for accessing and processing these sources. Due to the large number of available web services, the sheer complexity of the data and the frequent lack of documentation, discovering the most appropriate web service for a given task is a challenge for the user.

In this paper we propose a semi-automatic approach to assist the user in the discovery of which web services are the most appropriate to achieve her requirements. We describe the overall framework of our approach and we provide a detailed description of the techniques used in each phase of our approach. Finally, the usefulness of our approach is demonstrated through a Bioinformatics case study.

## 1 Introduction

In recent years, emergent fields such as the Life Sciences have experimented an exponential growth in the production of data and, consequently, a surge in the need for the creation of new techniques and technologies to analyze these data. This has caused an intense research effort in information integration techniques [9], which have to deal with a number of technical challenges. The first major challenge is the large number of available data sources (the latest reference on publicly available data sets in the Life Sciences [5] lists more than 1200 databases). A second problem is that these data sources have been developed by different institutions and, since there are few standards for data representation, there is a high level of data heterogeneity which causes serious impedance mismatch problems. Finally, the data sources are distributed, and typically available by using Web Services, which provide a limited API for data retrieval and exploration when compared with a full-fledged query language.

This situation creates challenging issues, given the typical process of research in Bioinformatics, which involves integrating data obtained by in-house experiments (for instance, DNA sequencing) with reference databases available on the Internet and queryable using a web page or a web service, and then performing analysis tasks using specific algorithms which may also be already available as web services [2]. To help researchers to find out publicly available services, some

repositories such as BioCatalogue [1] have been developed; however, searches are hampered by poor documentation and the lack for a standard way of creating annotations. As a consequence, finding data collections and web services which are appropriate for a given research task usually becomes a costly trial-and-error process [1].

Currently, to the best of our knowledge, there is no guide to assist users in the discovery process. There are approaches that focus on the development of interfaces to assist in the location of web services; [10] presents a client engine for the automatic and dynamic development of service interfaces built on top of BioMOBY standard. Other approaches focus on the the discovery of web services that are annotated with a specific vocabulary. This is the case of the *myGrid* project [4], whose aim is to provide a controlled vocabulary to make annotations. They have implemented BioCatalogue [1], a Life Sciences web service registry with 1180 registered web services that are meant to be annotated using their Life Sciences ontology, the *myGrid* ontology, which should provide in principle a high precision in search. However, most available annotations are just free text, and many web services are not annotated at all. Another issue to be taken into account is that, in many cases, multiple services provide very similar functionality (a particularly insidious example is the multitude of services providing variants of alignments of genes and proteins). In this case, the user has to decide which one is the most appropriate based on diverse quality criteria (availability, coverage of the domain of interest, and so on). To address this problem, assessment techniques must be applied to provide the user with some information about the quality and the functionality of each service [3].

In the literature, web service discovery has been deeply studied, but mainly for traditional business-oriented applications [13]. In scientific domains, such as the Life Sciences, there are very significant differences in skills and expectations with respect to those in business domains. The user in the Life Sciences is a scientist that is assumed to be expert in a highly complex and specific domain, and can be assumed to know exactly what she wants, but may not be an expert in information processing technology – even though she is capable (and usually willing) to deal directly with the integration of the necessary web services. This explains the emergence of specific applications aimed at allowing scientists to design in-silico experiments by combining discovered web services into workflows [4]. Moreover, as [13] remarked, web services in traditional business-oriented applications are usually annotated with a signature (inputs, outputs and exceptions), web service states (preconditions and postconditions) and non-functional values (attributes used in the evaluation of the service), information that facilitates the discovery of web services and their composition. In contrast to these domains, as we have mentioned above, Life Science web services are poorly documented and therefore, it is not possible to apply traditional discovery and composition techniques.

In this paper, we contribute a semi-automatic approach to assist the user in web service discovery, looking for web services that are appropriate to fulfill the information requirements of researchers in the Life Sciences domain. Our aim

---

<sup>1</sup> <http://www.mygrid.org.uk>

is to make the whole process driven by well-captured requirements, in order to avoid the high costs associated with non-disciplined, non-reusable, ad-hoc development of integration applications. It is important to note that the approach is semi-automatic by design; as we have said before, the user is a scientific expert on the domain which knows the analysis objectives and the steps to attain them.

The remainder of this paper is structured as follows. First, Section 2 presents an overview of the proposed approach. Next, Section 3 focuses on the requirements elicitation and specification phase. Section 4 describes the normalization phase and in Section 5 the web service discovery phase is explained. Then, in Section 6 we present a Bioinformatics case study that showcases the usefulness of our approach, and finally in Section 7 some conclusions and future research lines are summarized.

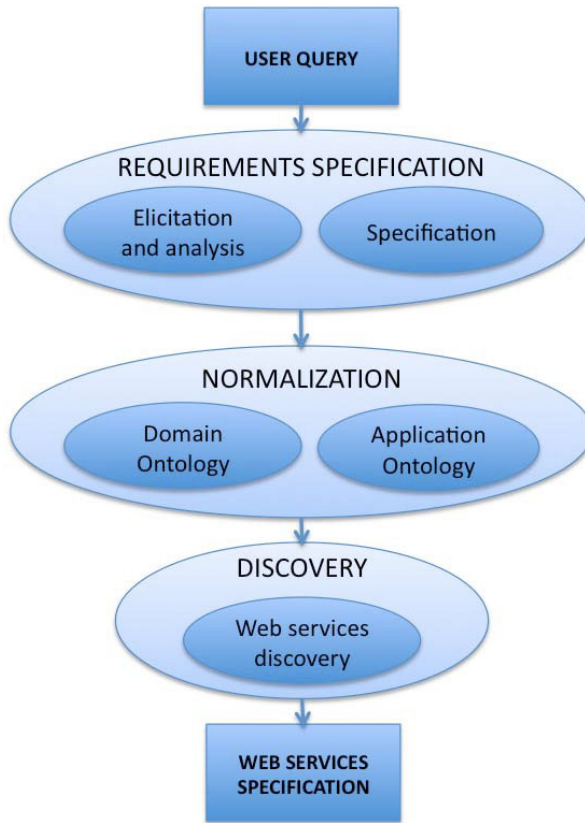
## 2 Approach Overview

The overall approach we propose to assist the discovery of web-services based on the users requirements is shown in Figure 1. It consists of three main phases:

1. **Requirements elicitation and specification.** Scientists have information requirements that must be fulfilled by information that is stored in any of the multiple available data sources or even information that is the result of processing data stored in those data sources.

The main purpose of this phase is to gather the user information requirements and to specify them in a formal way. The user determines her information needs and moreover, gives extra information about her experience and some knowledge describing the steps necessary to do it herself manually. The description of these steps is also included in the formal specification. Section 3 describes the techniques used in this phase.

2. **Normalization.** In the Requirements model, task descriptions are expressed in natural language, and therefore they must be normalized in order to be automatically processed. The normalization consists of a semantic annotation process in which the information of the requirements model, concretely the description of the tasks, is processed and annotated with semantic data. The normalization is carried out in two phases: (i) domain specific annotations, and (ii) application specific annotations. Section 4 provides a more detailed description of the normalization process.
3. **Web services discovery.** The aim of this step is to discover suitable web services that provide the functionality specified by the user-defined tasks in order to fulfill the user requirements. The discovery is based on the annotations made in the previous phase, so it depends largely on the quality of these annotations. In this paper, we focus only on the discovery of web services, but in the future we aim to discover other types of resources such as “naked” databases. Frequently, there is more than one web service providing the same functionality, and therefore the result of this phase is a set of web services per each user-defined task. Section 5 explains the discovery process.



**Fig. 1.** The phases of the proposed guide

At the end of the process, the user receives as output some sets of web services, one per each specified task, that provide the functionality required by the tasks. The user will then have to select the ones that are the most appropriate for her requirements. In order to be able to decide which are the best ones, which is a complex process outside the scope of this paper, the user may need some assessment techniques. In case the results are not those expected by the user, she can refine the process at the three phases of the guide: (i) Requirements phase: the requirements model is refined by modifying the goals or the tasks, (ii) Normalization phase: another category can be selected to annotate a specific task, or (iii) Web Services discovery phase: a different web service can be selected in the list of discovered services. This process is iterative and it can be executed as many times as required to refine the selection.

### 3 Requirements Specification

The requirements elicitation and specification process plays a crucial role in the discovery of web services, since the selection is basically driven by the user's



information requirements. These requirements must be gathered and formally specified to be the input of the discovery process. The requirements elicitation task is made by the system designer by having personal interviews with the end user. The user determines which information she needs and, moreover, gives extra information about her experience and knowledge describing the steps necessary to do it herself manually. All the provided information is completely independent of the characteristics of web services or other resources. The elicited requirements are analyzed by the system designer in order to detect inconsistencies or missing information. This analysis is made by querying domain ontologies and by interacting with the user.

Once the requirements have been elicited and analyzed, the system designer creates a formal specification of them, called the *Requirements model*, in order to be used in an automatic way in the subsequent phases. This specification is made using the  $i^*$  formalism [16,17], which is both a goal-oriented and an agent-oriented language. We use this framework because it provides the functionality required to obtain a formal specification of the user's requirements without taking into account the characteristics of the system. The goals and the tasks of the *Strategic Rationale* (SR) model of the  $i^*$  framework capture the user's information requirements and the steps, specified by the user, to achieve those requirements. Here, we generalize the techniques used in [12] to specify the user's requirements in the context of finding appropriate similarity measures for complex information.

## 4 Normalization

At this stage, the system designer has already gathered the user's information needs and she has specified them in the Requirements model. In this model the information about the tasks is described in natural language, which is hard to process in an automatic way. Therefore, the purpose of this phase is to annotate the tasks descriptions with domain and application ontologies to allow the reconciliation of the requirements of the user with the sources of information (web services) in a process that can be considered as a *normalization* of knowledge. In our case, it is only necessary to normalize the task descriptions, since they will be used to guide the discovery of web services. Among the different techniques to normalize natural language sentences we have chosen to use *semantic annotations*. Our aim is to select the most relevant terms in the task descriptions and use these terms to search for web services. We have divided the semantic annotation process in two steps:

1. **Domain specific annotations.** This step consists of identifying the terms of the task descriptions that are related to the domain in which we are working. For each task, the most relevant terms are retrieved by querying domain ontologies. In our domain, the Life Sciences domain, there are several domain ontologies that can be used to annotate semantically. One example of domain ontology is the UMLS, a metathesaurus that comprises the main biomedical

linguistic resources widely used in health care applications. Another example of Life Sciences ontology is *myGrid* ontology [15], which describes the Life Sciences research domain and data related to web services. *myGrid* ontology has been implemented in *myGrid* project and is aimed to provide a formal vocabulary to describe the web services. In some cases, the user can also decide that she prefers not to use any domain ontology, that is, to use the whole task descriptions expressed in free-text in the next step for tasks classification.

- 2. Application specific annotations.** Once the terms related to the domain have been detected, the next step is to use them to query the application ontology in order to determine the type of each task. As we are searching for Life Sciences web services, we apply the taxonomy of categories used by BioCatalogue in order to classify the user-defined tasks. BioCatalogue is a registry of curated Life Science Web Services, whose aim is to provide an easy way for scientists to discover web services of interest. BioCatalogue has a shallow taxonomy of web services categories, and most of the registered web services have at least one category. So, we use this taxonomy to classify the user-defined tasks with the aim of using these categories in the discovery of the web services that are suitable to fulfill the user's requirements. Figure 2 illustrates a fragment of the categories taxonomy of BioCatalogue.

The final result of this process is a ranked list of annotations for each task. By default, each task is annotated with the category that has the highest similarity score. These annotations have to be validated by the user.

## 5 Web Service Discovery

Web services in BioCatalogue have four main different types of annotations: descriptions, operations, tags and categories. As a first approximation, we have decided to use categories as the criterion for the discovery process, since they are formally described by a taxonomy and, moreover, they express very well the functionality of the services. For the future, we are planning to generalize this approach by applying techniques for the combination of several types of annotations as criterion for the discovery process, which would result in a more flexible approach [6].

In the Normalization phase, the tasks defined by the user have been annotated with the categories used in BioCatalogue in order to determine the functionality described by the tasks. The discovery process consists in querying BioCatalogue, using its recent launched API, as many times as tasks in the requirements model are. Each query searches for a specific category and retrieves a set of web services that are annotated with this category, that is, web services that are supposed to execute the functionality required by the task. In most cases, the search retrieves more than one service, since there are many services annotated with the same category. In this set of services, maybe some of them do not provide exactly the required functionality and this may be because some categories are too general to describe a specific functionality. So, it is the responsibility of the user to know

```

Service Categories
  Sequence Analysis
    Protein Sequence Analysis
      Protein Sequence Alignment
        Protein Multiple Alignment
        Protein Pairwise Alignment
      Protein Sequence Similarity
      Structure Prediction
        Secondary Structure
        Tertiary Structure
      Function Prediction
        Domains
        Motifs
        Repeats
    Nucleotide Sequence Analysis
      Nucleotide Sequence Alignment
        Nucleotide Multiple Alignment
        Nucleotide Pairwise Alignment
      Nucleotide Sequence Similarity
      Gene Prediction
      Promoter Prediction
      Transcription Factors
  Text Mining
    Document Discovery
    Document Similarity
    Document Clustering
    Named Entity Recognition
  Ontology
    Ontology Lookup
    Ontology Annotation
  Phylogeny
    Evolutionary Distance Measurements
    Tree Inference
    Tree Display
    Statistical Robustness

```

**Fig. 2.** Fragment of the BioCatalogue Taxonomy

which services are suitable for the tasks and which are not. We are currently working on assessment techniques that could assist the user in the web services selection.

## 6 Case Study

To prove the usefulness of our approach, a prototype web services discovery guide has been implemented by using the Eclipse EMF modelling framework<sup>2</sup>. The necessary transformations between models have been defined in the ATL<sup>7</sup> language. In this section, we use this prototype to develop a bioinformatics case study extracted from <sup>8</sup>. In this way, we can illustrate how to use our approach to guide the user in a real web service discovery task.

The case study concerns biological research that analyzes the presence of specific genes involved in the genesis of Parkinson's Disease, called LRRK2 genes, in different organisms, in order to know more about the biochemical and cellular functions of these genes. The author studies the presence of the LRRK2 genes in

<sup>2</sup> <http://www.eclipse.org/modeling/emf/>

the organism *N. vectensis*, since previous studies have shown that this organism is a key organism to trace the origin of these genes. The author describes the process step-by-step. We have selected this case study because it describes with detail the techniques used in every step, and it could be useful to validate our guide. However, our intention is not to model a concrete case study, but to offer a guide for more general cases.

Next, there is a short description of each one of the steps of our approach in order to discover the web services that provide the functionality required by the scientist.

- 1. Requirements elicitation and specification.** In this first phase the purpose is to gather the user's requirements through a personal interview. The system designer obtains as much information as possible, gathering the user's information requirements and the steps the user would make if she had to make the search herself manually. In this case study the user information requirement is to obtain a comparison of the LRRK2 genes in different organisms. The steps the user would make herself manually are: (i) to retrieve the protein sequences of the different domains of the gene in different databases; (ii) to predict the gene structure automatically combining the sequences retrieved in (i); (iii) to align protein sequences to build phylogenetic trees; (iv) to build the phylogenetic trees; and (v) to analyze the structure of the proteins.

Figure 3 shows the requirements model obtained as a result of this step and includes all the information elicited by the system designer.

- 2. Normalization.** Once the system designer has elaborated the requirements model, the next step is to normalize the description of the user-defined tasks. Next we describe each one of the two steps of the normalization.

- 2.1. Domain specific annotations.** This step consists in selecting the terms relevant to the domain. In this case study, we present both the results of annotating with the *my*Grid ontology (OWL ontology) and the results of not using any domain ontology.

The experiments we have made until now suggest that extracting the most relevant terms is only required when the task descriptions are too verbose, and not when the descriptions are short and simple sentences. Table 1 shows a ranked list of concepts of *my*Grid ontology similar to the task description terms, in this case *Retrieve protein sequences*. The matching has been made with ISub [14], a string metric for the comparison of names used on the process of ontology alignment and on other matching problems. Each matching has a score that indicates the similarity between the description and the concepts of the ontology.

In this example the task description is clear and simple, but the matching with *my*Grid ontology does not obtain good results. For instance, the top ranked concept has a very low score, 0.114, and it corresponds to the ontology concept *protein sequence id*, which is not very similar to

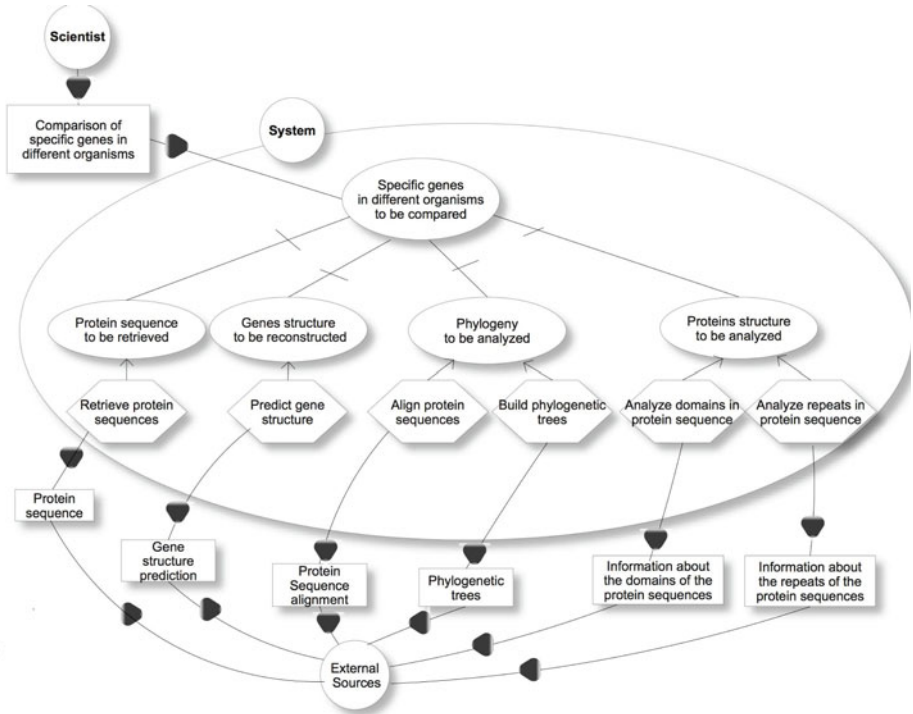


Fig. 3. Requirements model of the case study

*Retrieve protein sequences.* The scores are low because the sentences of this case do not contain meaningless words and, due to them, the user has decided to classify the tasks without using any domain ontology, that is, using all the words in the task description.

Table 1. Matching between the *Retrieve Protein Sequences* and <sup>my</sup>Grid ontology

Concepts	Score
protein_sequence_id	0.11428571428571432
protein_sequence_record	0.08706390861376972
protein_sequence_database	0.07459572248304647

2.2. **Application specific annotations.** In this step the tasks are classified based on their functionality. This classification is based on the BioCatalogue taxonomy of categories. The matching is not exact, so a ranked list of categories per each task is provided to the system designer. Figure 2 shows a fragment of the web services categories taxonomy of BioCatalogue used to annotate the task descriptions. Table 2 shows the ranked list of the task *Retrieve protein sequences*; this matching has been made also using the ISub metric and, all the terms of the task description have participated in the matching. The category *Sequence retrieval*, which is

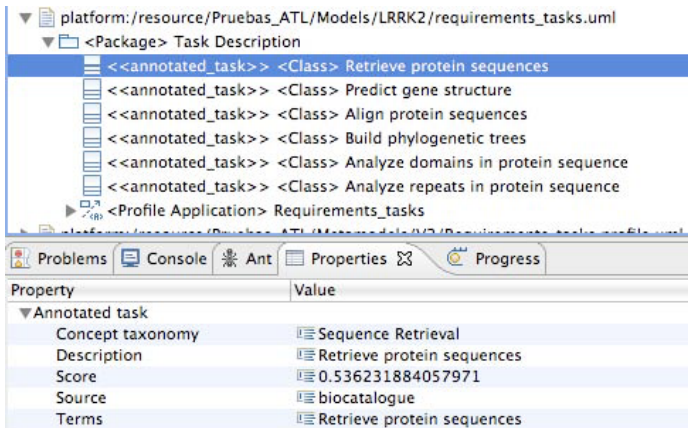
the one with the highest similarity score, is automatically selected. In case the user does not agree with the selected category, she could select any other category in the ranked list.

Figure 4 shows a screenshot of a fragment of the model produced by our prototype that contains all the tasks annotated with BioCatalogue categories. This fragment shows the details of the annotated task *Retrieve protein sequences*. The task *Retrieve protein sequences* has been annotated with the taxonomy concept *Sequence retrieval* extracted from the source *BioCatalogue* with a score of 0.53652 and the terms used to query the application taxonomy have been *Retrieve protein sequences*.

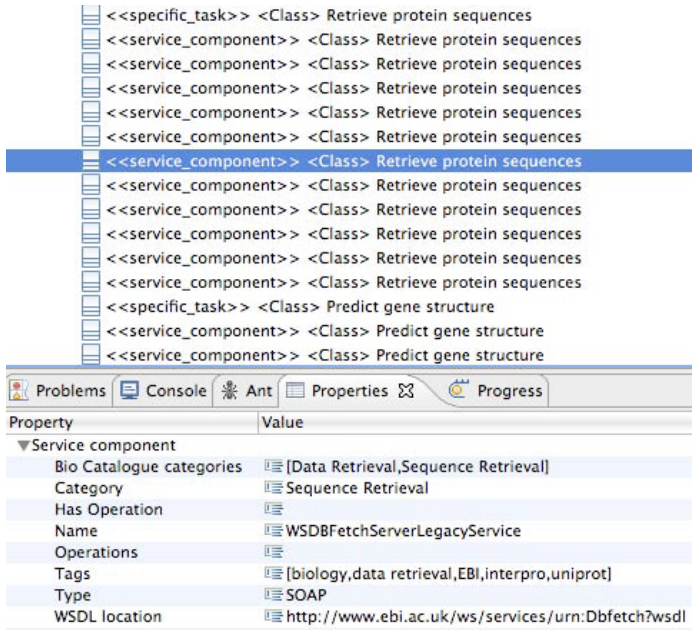
- Web Service discovery.** The web service discovery process is carried out by searching in the BioCatalogue registry services that are annotated with the same category as the user-defined tasks. This search retrieves a set of web services per each user-defined task, and there is no way to know in advance which one is the most appropriate for the user-defined task. So, at this step, the information of each retrieved web service is stored in a model, called *Service model*, which contains a component for each retrieved web service. Each component stores information about the categories, tags, type and WSDL location of the web service. This component is extensible to other required attributes of Web services.

**Table 2.** Ranked list of categories for the task *Retrieve Protein Sequences*

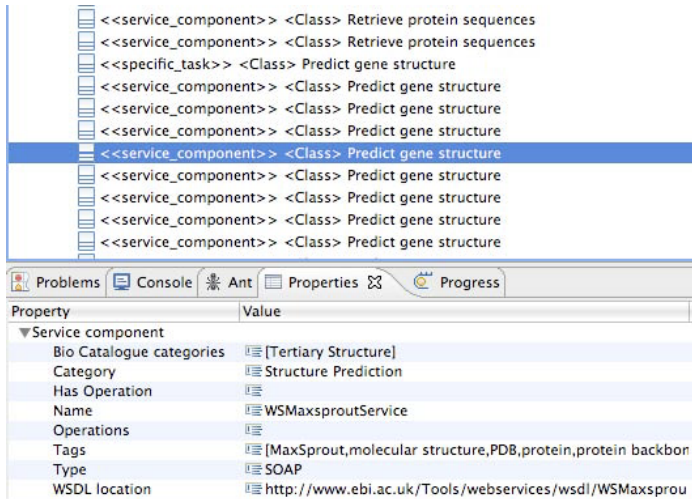
Categories	Score
Sequence Retrieval	0.536231884057971
Sequence Similarity	0.4285714285714286
Sequence Analysis	0.39468690702087295
Sequence Alignment	0.3758921490880254



**Fig. 4.** Normalization of the task *Retrieve protein sequences*



**Fig. 5.** *WSDBFetchServerLegacyService* service component. Candidate for the task Retrieve protein sequence.



**Fig. 6.** *WSMaxsproutService* service component. Candidate for the task Predict gene structure.

Figure 5 shows a fragment of the Service model produced by our prototype. In this fragment it can be seen that each user-defined task has a set of discovered web services. In the figure it is shown the information of the web service *WSDBFetchServerLegacyServer* that is a SOAP service whose location is <http://www.ebi.ac.uk/ws/services/urn:Dbfetch?wsdl> and it is annotated with the tags: *biology*, *data retrieval*, *EBI*, *interpro*, *uniprot*, and its categories in BioCatalogue are *Data Retrieval* and *Sequence Retrieval*.

Figure 6 shows the information of the web service *WSMacsproutService* whose category, *Tertiary Structure*, is not the category of the user-defined task, but it is a specialization of it.

This model is provided to the user who has to decide which web service is the most appropriate for her information requirements.

## 7 Conclusions and Future Work

The approach presented in this paper is focused on domains where applications have to deal with distributed and highly heterogeneous data sources, in which there are few standards for data representation and usually they require an exploratory access. In these domains many web services have been implemented, but due to their distribution and the frequent lack of documentation, they are not easily discovered by potential users. Life Sciences is an example of this type of domain, since many biological data have been released in recent years and, in consequence, many techniques and applications have been implemented to manage that data.

In this paper we have presented a semi-automatic approach to guide the user in the discovery of the appropriate techniques required to fulfill their information needs. We have mainly focused on web services, due to their wide acceptance and popularity within the Life Sciences domain. So, the aim of our approach is to assist the user in the discovery of the web services that provide the necessary functionality to fulfill her information requirements.

Considering the type of end-user the approach is addressed to, an expert on the domain with knowledge and experience to recognise which web services are the most appropriate, one of the main benefits of the approach is that it is a semi-automatic process. In this way, the user is able to change parameters, annotations or automatic selections in each one of the phases in the way she thinks is better, based on her own knowledge and experience or previous results. Thus, we aim to design a process of exploratory search, advising the user in each step, and taking advantage of her previous knowledge.

Currently, we are working on web service assessment techniques. The purpose of this assessment is to assist the user in the web service selection providing her with a set of measures to select the most appropriate among all the discovered services. We are working not only on general QoWS measures, but also on measures that analyze the quality of web services with respect to the user's requirements, for example, measures that analyze the type and the semantics of the results retrieved by a web service, the relevance of the web service with respect to the other user-defined tasks, etc.



Another important issue is the data sources discovery. Nowadays, we are only focusing on web services but in a near future, we also aim to search for data sources that may contain the required information. Their discovery will be based on techniques similar to the ones used for web services discovery and we expect that the main differences will appear in the types of measures necessary to assess the data sources.

Finally, we aim to develop an end-user tool to assist the user during all the process, since her requirements specification to the validation of the results, in order to avoid the presence of a system designer expert.

## Acknowledgements

This research has been supported by the Spanish Ministry of Education and Science (grant TIN2008-01825/TIN) and by Universitat Jaume I – Fundació Bancaixa (grant P11B2008-43). María Pérez has been supported by Universitat Jaume I predoctoral grant PREDOC/2007/41.

## References

1. Belhajjame, K., Goble, C., Tanoh, F., Bhagat, J., Wolstencroft, K., Stevens, R., Nzuobontane, E., McWilliam, H., Laurent, T., Lopez, R.: BioCatalogue: A Curated Web Service Registry for the Life Science Community. In: Microsoft eScience Conference (2008)
2. Burgun, A., Bodenreider, O.: Accessing and integrating data and knowledge for biomedical research. *Med. Inform. Yearb.* 2008, 91–101 (2008)
3. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. *Web Sem.* 1(3), 281–308 (2004)
4. Stevens, R., Goble, C., Pocock, M., Li, P., Hull, D., Wolstencroft, K., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34(Web Server issue), 729–732 (2006)
5. Cochrane, G.R., Galperin, M.Y.: The 2010 Nucleic Acids Research Database Issue and online Database Collection: a community of data resources. *Nucleic Acids Research* 38, 1–4 (2010)
6. Prez, M., Sanz, I., Berlanga, R.: Measure selection in multi-similarity xml applications. In: 3rd International Workshop on Flexible Database and Information System Technology, FlexDBIST-08 (2008)
7. Jouault, F., Kurtev, I.: Transforming models with atl. In: Bruel, J.-M. (ed.) MoD-ELS 2005. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
8. Marín, I.: Ancient origin of the Parkinson disease gene LRRK2. *Journal of Molecular Evolution* 64, 41–50 (2008)
9. Mesiti, M., Jiménez-Ruiz, E., Sanz, I., Berlanga, R., Valentini, G., Perlasca, P., Manset, D.: Data integration issues and opportunities in biological XML data management. In: Open and Novel Issues in XML Database Applications: Future Directions and Advanced Technologies. IGI Global (2009)
10. Navas-Delgado, I., Rojano-Muñoz, M., Ramírez, S., Pérez, A.J., León, E.A., Aldana-Montes, J.F., Trelles, O.: Intelligent client for integrating bioinformatics services. *Bioinformatics* 22(1), 106–111 (2006)

11. Anderson, N.R., Lee, E.S., Brockenbrough, J.S., Minie, M.E., Fuller, S., Brinkley, J., Tarczy-Hornoch, P.: Issues in biomedical research data management and analysis: needs and barriers. *J. Am. Med. Inform. Assoc.* 14(4), 478–488 (2007)
12. Pérez, M., Casteleyn, S., Sanz, I., Aramburu, M.J.: Requirements gathering in a model-based approach for the design of multi-similarity systems. In: *MoSE+DQS '09: Proceeding of the First International Workshop on Model Driven Service Engineering and Data Quality and Security*, pp. 45–52. ACM, New York (2009)
13. Rao, J., Su, X.: A survey of automated web service composition methods. In: Cardoso, J., Sheth, A.P. (eds.) *SWSWPC 2004*. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)
14. Stoilos, G., Stamou, G. B., Kollias, S.D.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) *ISWC 2005*. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
15. Wolstencroft, K., Alper, P., Hull, D., Wroe, C., Lord, P.W., Stevens, R.D., Goble, C.A.: The mygrid ontology: bioinformatics service discovery. *Int. J. Bioinformatics Res. Appl.* 3(3), 303–325 (2007)
16. Yu, E.: *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Canada (1995)
17. Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. In: *RE 1997*, vol. 85, pp. 2444–2448 (1997)

# An Open Platform for Business Process Modeling and Verification\*

Antonio De Nicola<sup>1</sup>, Michele Missikoff<sup>1</sup>, Maurizio Proietti<sup>1</sup>, and Fabrizio Smith<sup>1,2</sup>

<sup>1</sup> IASI-CNR, Viale Manzoni 30, 00185, Rome, Italy

<sup>2</sup> DIEI, Università degli Studi de L'Aquila, Italy

{antonio.denicola,michele.missikoff,maurizio.proietti,  
fabrizio.smith}@iasi.cnr.it

**Abstract.** In this paper we present the BPAL platform that includes a logic-based language for business process (BP) modeling and a reasoning mechanism providing support for several tasks. Firstly, the definition of a BP meta-model (MM) consisting of a set of rules that guide the BP designers in their work. Secondly, given a BP, the BPAL platform allows for the automatic verification of the compliance (well-formedness) of a given BP w.r.t. the defined MM. Finally, the execution semantics of a BP is given in term of its instances (referred to as traces) to provide services for *i*) checking if the actual execution of a BP has been carried out in accordance with the corresponding definition, *ii*) simulating executions by trace generation. The proposed platform is open since it can easily be enhanced by adding other logic-based modeling, reasoning, and querying functionalities.

**Keywords:** business process, modeling language, Horn logic, BPAL.

## 1 Introduction

Business Process (BP) management is constantly gaining popularity in various industrial sectors, especially in medium to large enterprises, and in the public administration. BP modeling is a complex human activity, requiring a special competence and, typically, the use of a BP design tool. Several tools<sup>1</sup> are today available on the market, open source or free of charge. Many of these tools are able to provide, besides a graphical editor, additional services, such as some forms of verification, simulation of the designed processes, execution or (semi) automatic generation of executable code (e.g., in the form of BPEL<sup>2</sup> code). The availability of the mentioned tools has further pushed the diffusion of several languages (e.g.,

---

\* This work is partially supported by the Tocai Project (<http://www.dis.uniroma1.it/~tocai/>), funded by the FIRB Programme of the Italian Ministry of University and Research (MIUR).

<sup>1</sup> See for instance: Intalio BPMS Designer, Tibco Business Studio, ProcessMaker, YAWL, JBPM, Enhydra Shark.

<sup>2</sup> Business Process Execution Language. See: <http://www.oasis-open.org/specs/#wsbpelv2.0>

BPMN [7], EPC [15]) used both in the academic and in the industrial realities. But, despite the growing academic interest and the penetration in the business domain, heterogeneous and ad-hoc solutions that often lack a formal semantics have been so far proposed to deal with the different perspectives that are of interest for an effective BP management: workflow modeling, business rules representation, integration of organizational and data models, data flow, query and retrieval of BP fragments, enactment, reengineering, log analysis, process mining.

This paper mainly intends to lay the formal foundations of a platform for BP modeling and verification. The proposed platform is centered around BPAL (Business Process Abstract Language) [8], a logic-based language for modeling the dynamic behavior of a business process from a workflow perspective. BPAL relies on a formalism, Horn clause logic, that is particularly well-suited for its use within a wider knowledge representation framework (for instance in conjunction with rule based ontology languages [19,20]) with an uniform semantics. The use of a logic-based uniform approach makes the BPAL platform open to further extensions and to the easy integration of more advanced functionalities, such as reasoning services for verifying consistency properties and methods for querying BP repositories, which can be supported by tools already developed in the area of logic programming.

BPAL is a rule-based formalism that provides an integrated support to the following three levels: (i) the BP meta-model, where we define the meta-model, establishing the rules for building well-formed BPs; (ii) the schema level, where we define the BP schemas, in accordance with the given meta-model; (iii) the ground level, where we represent the BP instances, i.e., the traces that are produced by the execution of a BP. The reasoning support provided by the BPAL platform allows the BP designer to: (i) verify business process schema well-formedness with respect to the given meta-model; (ii) verify if a given process trace, i.e., the actual execution of a BP, is compliant with a well-formed BP schema; (iii) simulate a BP execution by generating all possible traces (which are finitely many, whenever the BP schema is well-formed).

The BPAL platform is characterized by both a solid formal foundation and a high level of practical usability. The formal foundation is rooted in the logic-based approach of the BPAL language. The practical usability is guaranteed by the fact that BPAL platform has not been conceived as an alternative to existing BP tools but, conversely, it intends to be associated to the existing BP modeling tools enhancing their functionalities.

The rest of this paper is organized as follows. In Section 2 some relevant related works are presented. The BPAL language for business process modeling and verification is described in Section 3. Section 4 presents the BPAL meta-model and in Section 5 the execution semantics (in term of *execution traces*) of a BPAL BP schema is described. In Section 6 an overview of the BPAL platform, consisting of the well-formedness verification service, the trace analysis, and traces generation service, is presented. Finally, conclusions in Section 7 end the paper.

## 2 Related Works

In the literature, much attention is given to BP modeling, as its application to the management of complex processes and systems [10] is an important issue in business organizations. It appears increasingly evident that a good support to BP management requires reliable BP modeling methods and tools. Such reliability can be achieved only if the adopted method is based on formal foundations. In this perspective, our work is related to the formal BP languages for the specification, the verification, and analysis of business processes. The BPAL framework is positioned among the logic-based languages but, with respect to existing proposals, it is characterized by enhanced adaptability, since we propose a progressive approach where a business expert can start with the (commercial) tool and notation of his/her choice and then enrich its functionalities with BPAL.

Formal semantics of process modeling languages (e.g., the BPMN case is discussed in [1]) is usually defined in terms of a mapping to Petri nets [2]. Petri nets represent a powerful formal paradigm to support automatic analysis and verification of BPs within a procedural approach. A different approach is represented by the logic-based formalisms. A logical approach appears more suited to manipulate, query, retrieve, compose BP diagrams. Furthermore, by using Petri Nets, it is difficult to provide a “meta-level” that can be used to guide and constrain business process modeling, verifying properties at the intensional level.

As already mentioned, a different approach to formal BP specification is represented by a logic-based declarative approach [3,4]. Here a process is modeled by a set of *constraints* (business rules) that must be satisfied during execution: these proposals provide a partial representation of a BP that overlooks the procedural view, i.e., the control flow among activities. [3] proposes ConDec, a declarative flow language to define process models that can be represented as conjunction of Linear Temporal Logic formulas. This approach allows the BP designer to verify properties by using model checking techniques. [4] proposes a verification method based on Abductive Logic Programming (ALP) and, in particular, the SCIFF framework [5], that is an ALP rule-based language and a family of proof procedures for specification and verification of event-based systems. [3,4], are based on rigorous mathematical foundations but they propose a paradigm shift from traditional process modeling approaches that is difficult to be understood and, consequently, to be accepted by business people. Such approaches are mostly intended to complement and extend fully procedural languages rather than replace them, as in the case of Declare<sup>3</sup>, which is implemented within the YAWL<sup>4</sup> workflow management system.

PSL (Process Specification Language) [6], defines a logic-based neutral representation for manufacturing processes. A PSL ontology is organized into PSL-CORE and a partially ordered set of extensions. The PSL-CORE axiomatizes a set of intuitive semantic primitives (e.g., *activities*, *activity occurrences*, *time points*, and *objects*) enabling the description of the fundamental concepts of processes, while a set of extensions introduce new terminology and its logical formalization. Although PSL

---

<sup>3</sup> <http://www.win.tue.nl/declare/>

<sup>4</sup> <http://www.yawlfoundation.org/>

is defined in first order logic, which in principle makes behavioral specifications in PSL amenable to automated reasoning, we are not aware of PSL implementations for the modeling, verification or enactment of BPs, since it is intended mostly as a language to support the exchange of process information among systems.

Concurrent Transaction Logic (CTR) [17] is a formalism for declarative specification, analysis, and execution of transactional processes, that has been also applied to modeling and reasoning about workflows and services [18]. CTR formulas extend Horn clauses by introducing three new connectives: *serial conjunction*, which denotes sequential executions, *concurrent conjunction*, which denotes concurrent execution, and *isolation*, which denotes transactional executions. The model-theoretic semantics of CTR formulas is defined over paths, i.e., sequences of states. These formulas can be compiled for the execution<sup>5</sup> in a Prolog environment. Unlike a CTR formula, a BPAL process specification (i.e., the Horn clauses specifying the meta-model, the process schema, and the trace semantics) can be directly viewed as an executable logic program and, hence: (i) a BPAL specification can be queried by any Prolog system without need of a special purpose compiler, (ii) BPAL traces are explicitly represented and can be directly analyzed and manipulated, and (iii) other knowledge representation applications (e.g., ontology management systems) can easily be integrated by providing a suitable translation to logic programming.

### 3 The BPAL Language

BPAL is a logic-based language that has been conceived to provide a declarative modeling method capable of fully capturing the procedural knowledge in a business process. BPAL constructs are common to the most used and widely accepted BP modeling languages (e.g., BPMN, UML activity diagrams, EPC) and, in particular, its core is based on BPMN 2.0 specification [7]. Furthermore, the design principles of the language follow the MOF paradigm<sup>6</sup> with the four levels briefly reported below:

- M3: Meta-metalevel.* The top level is represented by the logical formalism that is applied to describe the lower levels. In particular we adopted Horn logic, due to its widespread popularity and the mature technological support provided by the numerous Prolog systems which are available.
- M2: Metalevel.* Here it is defined the meta-model, establishing the rules for building well-formed BPs.
- M1: Schema level.* This is the modeling level where it is defined the BP schema, in accordance with the given meta-model, that represents the business logic of the process.
- M0: Trace level.* This is the *ground* level, used to model the executions of a business process, in accordance with the corresponding BP schema.

---

<sup>5</sup> <http://flora.sourceforge.net/>

<sup>6</sup> OMG,(2006),Meta Object Facility (MOF) Core Specification V2.0,  
<http://www.omg.org/docs/formal/06-01-01.pdf>.

From a formal point of view, the BPAL language consists of two syntactic categories: (i) a set *Entities* of constants denoting entities to be used in the specification of a business process schema (e.g., business activities, events, and gateways) and (ii) a set *Pred* of *predicates* denoting relationships among BPAL entities. Finally, a BPAL business process schema (BPS) is specified by a set of ground *facts* (i.e., atomic formulas) of the form  $p(C_1, \dots, C_n)$ , where  $p \in \text{Pred}$  and  $C_1, \dots, C_n \in \text{Entities}$ .

The entities occurring in a BPS are represented by the following set of predicates:

*flow\_el(el)*: *el* is a *flow element*, that is, any atomic component appearing in the control flow. A flow element is either an activity or an event or a gateway;

*activity(act)*: *act* is a *business activity*, the key element of the business process;

*event(ev)*: *ev* is an *event* that occurs during the process execution. An event is of one of the following three types: (i) a *start* event, which starts the business process, (ii) an *intermediate* event, and (iii) an *end* event, which ends the business process. These three types of events are specified by the three predicates *start\_ev(start)*, *end\_ev(end)*, and *int\_ev(int)*;

*gateway(gat)*: *gat* is a *gateway*. A gateway is either a *branch* or a *merge point*, whose types are specified by the predicates *branch\_pt(gat)* and *mrg\_pt(gat)*, respectively. A branch (or merge) point can be either a *parallel*, or an *inclusive*, or an *exclusive* branch (or merge) point. Each type of branch or merge point is specified by a corresponding unary predicate.

Furthermore BPAL provides a set of relational predicates to model primarily the sequencing of activities. Then, in case of branching flows, BPAL provides *parallel* (i.e., AND), *exclusive* (i.e., XOR), and *inclusive* (i.e., OR) *branching/merging* of the control flow. Here we adopted the standard semantics for branching and merging points:

*seq(el1,el2)*: the flow element *el1* is immediately followed by *el2*.

*par\_branch(gat,el1,el2)*<sup>7</sup>: *gat* is a *parallel branch* point from which the business process branches to two sub-processes started by *el1* and *el2* executed in *parallel*;

*par\_mrg(el1,el2,gat)*: *gat* is a *parallel merge* point where the two sub-processes ended by *el1* and *el2* are synchronized;

*inc\_branch (gat,el1,el2)*<sup>8</sup>: *gat* is an *inclusive branch* point from which the business process branches to two sub-processes started by *el1* and *el2*. At least one of the sub-processes started by *el1* and *el2* is executed;

*inc\_mrg(el1,el2,gat)*: *gat* is an *inclusive merge* point. At least one of the two sub-processes ended by *el1* and *el2* must be completed in order to proceed;

---

<sup>7</sup> We represent only binary branches, while they are n-ary in the general case. This limitation is made for presentation purposes and can be easily removed.

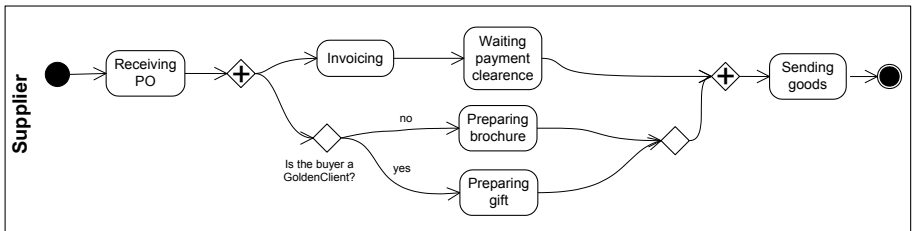
<sup>8</sup> *Inclusive* and *exclusive* gateways, in their general formulation, are associated with a condition. For instance, *exc\_dec* tests a condition to select the path where the process flow will continue.

$exc\_branch(gat,el1,el2)$ :  $gat$  is an *exclusive branch* point from which the business process branches to two sub-processes started by  $el1$  and  $el2$  executed in mutual exclusion;

$exc\_mrg(el1,el2,gat)$ :  $gat$  is an *exclusive merge* point. Exactly one of the two sub-processes ended by  $el1$  and  $el2$  must be completed in order to proceed;

To better present the BPAL approach, we briefly introduce as a running example a fragment of an eProcurement. An ACME supplier company receives a purchase order from a buyer and sends back an invoice. The buyer receives the invoice and makes the payment to the bank. In the meanwhile, the supplier prepares a gift for the buyer if she/he is classified as golden client, otherwise he prepares a brochure. After receiving the payment clearance from the bank, the supplier sends the goods to the buyer.

The Figure 1 reports a BPMN diagram that illustrates the fragment of the eProcurement process from the supplier perspective. The same process is reported in Table 1 encoded as a BPAL BPS.



**Fig. 1.** BPMN specification of a fragment of an eProcurement example

**Table 1.** BPAL BPS of the eProcurement example

$start\_ev(Start)$	$end\_ev(End)$
$activity(ReceivingPO)$	$seq(Start,ReceivingPO)$
$activity(Invoicing)$	$seq(ReceivingPO,Gat1)$
$activity(WaitingPaymentClearance)$	$seq(Invoicing,WaitingPaymentClearance)$
$activity(PreparingGift)$	$seq(Gat2,SendingGoods)$
$activity(PreparingBrochure)$	$seq(SendingGoods,End)$
$activity(SendingGoods)$	$par\_branch(Gat1,Invoicing,Gat3)$
$par\_branch\_pt(Gat1)$	$par\_mrg(WaitingPaymentClearance,Gat4,Gat2)$
$par\_mrg\_pt(Gat2)$	$exc\_branch(Gat3,PreparingBrochure,PreparingGift)$
$exc\_branch\_pt(Gat3)$	$exc\_mrg(PreparingBrochure,PreparingGift,Gat4)$
$exc\_mrg\_pt(Gat4)$	



## 4 BPAL Meta-model

The first service provided by BPAL enables the BP designer to check the compliance of a BPS with the BP meta-model, i.e., with a set of rules that constitute a guidance for the construction of the BP.

In this paper the main assumption imposed by the BPAL meta-model is the structuredness. According to [11], a **strictly structured** BP can be defined as follows: it consists of  $m$  sequential blocks,  $T_1 \dots T_m$ . Each block  $T_i$  is either elementary, i.e., it is an activity, or complex. A complex block  $i$ ) starts with a branch node (a parallel, inclusive or exclusive gateway) that is associated with exactly one merge node of the same kind that ends the block,  $ii$ ) each path in the workflow graph originating in a branch node leads to its corresponding merge node and consists of  $n$  sequential blocks (simple or complex). It is worth noting that removing the structured assumption leads to several weaknesses [12]. Among them, error patterns [13] such as deadlocks, livelocks and dead activities cannot manifest in a structured BPS.

The presence of a meta-model allows us to automatically prove the first fundamental property: the fact that a BPAL process schema has been built in the respect of the meta-model. We will refer to such a property as *well-formedness*.

In the rest of this section we describe the core of the meta-model of BPAL by means of a set of rules (i.e., a first order logic theory)  $MM$ , which specifies when a BP is well-formed, i.e., it is syntactically correct.  $MM$  consists of three sets of meta-rules<sup>9</sup>: (1) a set  $I$  of inclusion axioms among the BPAL entities, (2) a set  $K$  of *schema constraints* (in the form of first order formulas), and (3) a set  $F$  of *process composition rules* (in the form of Horn clauses).

The set  $I$  of *inclusion axioms* defines a taxonomy among the BPAL entities, as informally described in Section 3. They are reported in Table 2.

**Table 2.** BPAL inclusion axioms

$event(x) \rightarrow flow\_el(x)$	$mrg\_pt(x) \rightarrow gateway(x)$
$activity(x) \rightarrow flow\_el(x)$	$par\_branch\_pt(x) \rightarrow branch\_pt(x)$
$gateway(x) \rightarrow flow\_el(x)$	$exc\_branch\_pt(x) \rightarrow branch\_pt(x)$
$start\_ev(x) \rightarrow event(x)$	$inc\_branch\_pt(x) \rightarrow branch\_pt(x)$
$int\_ev(x) \rightarrow event(x)$	$par\_mrg\_pt(x) \rightarrow mrg\_pt(x)$
$end\_ev(x) \rightarrow event(x)$	$exc\_mrg\_pt(x) \rightarrow mrg\_pt(x)$
$branch\_pt(x) \rightarrow gateway(x)$	$inc\_mrg\_pt(x) \rightarrow mrg\_pt(x)$

The set  $K$  of schema constrains (Table 3) consists of three subsets: (i) the *domain constraints*, (ii) the *type constraints*, and (iii) the *uniqueness constraints*.

<sup>9</sup> All formulas in  $MM$  are universally quantified in front and, for sake of simplicity, we will omit to write those quantifiers explicitly.

**Table 3.** BPAL schema constraints and supporting examples

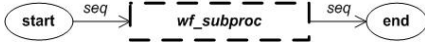
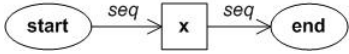

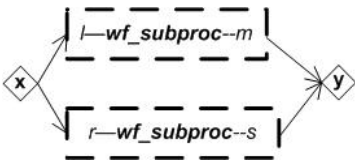
<i>Schema constraint</i>	<i>Example</i>
<b>Domain constraints</b> are formulas expressing the relationships among BPAL unary predicates.	A flow element cannot be an activity and an event at the same time. $activity(x) \rightarrow \neg event(x)$
<b>Type constraints</b> are rules specifying the types of the arguments of relational predicates.	A parallel branch is defined among a parallel branch point and two flow elements. $par\_branch(x,l,r) \rightarrow par\_branch\_pt(x) \wedge flow\_el(l) \wedge flow\_el(r)$
<b>Uniqueness Constraints</b> are rules expressing that the precedence relations between flow elements are specified in an unambiguous way:  <i>branching uniqueness constraints</i> asserting that every (parallel, inclusive, exclusive) branching point has exactly one pair of successors.  <i>merging uniqueness constraints</i> asserting that every merge point has exactly one pair of predecessors.  <i>sequence uniqueness constraints</i> asserting that, by the <i>seq</i> predicate, we can specify at most one successor and at most one predecessor of any flow element.	Example of sequence uniqueness constraint:  $seq(x,y) \wedge seq(x,z) \rightarrow y=z$ $seq(x,z) \wedge seq(y,z) \rightarrow x=y$

The set  $F$  of *process composition rules* provides the guidelines for building a well-formed BPS. Then, in formal terms, it is possible to verify if a process respects such rules by means of a predicate  $wf\_proc(s,e)$  which holds if the business process started by the event  $s$  and ended by the event  $e$  is *well-formed*. In Table 4, some rules are reported that inductively define what is a well-formed process ( $wf\_proc$ ) by means of the notion of sub-process and its well-formedness ( $wf\_sub\_proc$ ).

We are now ready to give a definition of the well-formedness of a BP schema  $B$ . We say that  $B$  is *well-formed* if:

- (i) every schema constraint  $C$  in  $K$  can be inferred from  $B \cup F \cup I$ , and
- (ii) for every start event  $S$  and end event  $E$ ,  $wf\_process(S,E)$  can be inferred from  $B \cup F \cup I$ .

**Table 4.** BPAL process composition rules and supporting diagrammatic description

<i>Process composition rule</i>	<i>Intuitive Diagram</i>
<p><b>F1.</b> A business process schema is <i>well-formed</i> if (i) it is started by a start event <math>s</math>, (ii) it is ended by an end event <math>e</math>, and (iii) the sub-process from <math>s</math> to <math>e</math> is a well-formed sub-process constructed according to rules F2-F6: <math>(start\_ev(s) \wedge wf\_sub\_proc(s,e) \wedge end\_ev(e)) \rightarrow wf\_proc(s,e)</math></p>	<p>A well-formed process:</p> 
<p><b>F2.</b> Any activity/event or sequence of two activities/events is a well-formed sub-process:  <math>activity(x) \rightarrow wf\_sub\_proc(x,x)</math>  <math>int\_ev(x) \rightarrow wf\_sub\_proc(x,x)</math>  <math>seq(x,y) \rightarrow wf\_sub\_proc(x,y)</math></p>	<p>So, the simplest well-formed process is graphically represented as:</p> 
<p><b>F3.</b> A sub-process is well-formed if it can be decomposed into a concatenation of two <i>well-formed</i> sub-processes:  <math>wf\_sub\_proc(x,y) \wedge wf\_sub\_proc(y,z) \rightarrow wf\_sub\_proc(x,z)</math></p>	<p>A well-formed sub-process:</p> 
<p><b>F4.</b> A sub-process started by a branch point <math>x</math> and ended by a merge point <math>y</math> is well-formed if (i) <math>x</math> and <math>y</math> are of the same type, and (ii) both branches contain two well-formed sub-processes<sup>10</sup>:  <math>par\_branch(x,l,r) \wedge wf\_sub\_proc(l,m) \wedge wf\_sub\_proc(r,s) \wedge par\_mrg(m,s,y) \rightarrow wf\_sub\_proc(x,y)</math></p>	<p>A well-formed sub-process including merge and branch points:</p> 

## 5 BPAL Execution Traces

An execution of a business process is a sequence of instances of activities called *steps*; the latter may also represent instances of events. Steps are denoted by constants taken from a set *Step* disjoint from *Entities*. Thus, a possible execution of a business process is a sequence  $[s_1, s_2, \dots, s_n]$ , where  $s_1, s_2, \dots, s_n \in Step$ , called a *trace*. The *instance* relation between steps and activities (or events) is specified by a binary

<sup>10</sup> The rules F5 and F6 defining the predicate  $wf\_sub\_process(x,y)$  in the cases where  $x$  is an inclusive or an exclusive decision gateway are similar and are omitted.

predicate  $inst(step,activity)$ . For example,  $inst(RPOI, ReceivingPO)$  states that the step  $RPOI$  is an activity instance of  $ReceivingPO$ .

**Table 2.** BPAL Trace rules

<p><b>T1.</b> A sequence <math>[s1, \dots, e1]</math> of steps is a <i>correct trace</i> if: (i) <math>s1</math> is an instance of a start event, (ii) <math>e1</math> is an instance of an end event, and (iii) <math>[s1, \dots, e1]</math> is a correct <i>sub-trace</i> from <math>s1</math> to <math>e1</math> constructed according to the sets of rules T2-T6:</p> $start\_ev(s) \wedge inst(s1,s) \wedge sub\_trace(s1,t,e1) \wedge end\_ev(e) \wedge inst(e1,e) \rightarrow trace(t)$
<p><b>T2.</b> Any instance of an activity/event or a sequence of instances of activities/events is a correct sub-trace.</p> $inst(x1,x) \wedge activity(x) \rightarrow sub\_trace(x1,[x1],x1)$ $inst(x1,x) \wedge int\_ev(x) \rightarrow sub\_trace(x1,[x1],x1)$ $inst(x1,x) \wedge inst(y1,y) \wedge seq(x,y) \wedge act\_or\_ev\_seq([x1,y1],t) \rightarrow sub\_trace(x1,t,y1)$ <p>where the predicate <math>act\_or\_ev\_seq([x1,y1],t)</math> holds iff <math>t</math> is the sequence obtained from <math>[x1,y1]</math> by deleting the steps which are not instances of activities or events.</p>
<p><b>T3.</b> A trace is correct if it can be decomposed into a concatenation of two correct sub-traces:</p> $sub\_trace(x1,t1,y1) \wedge sub\_trace(y1,t2,z1) \wedge concatenation(t1,t2,t) \rightarrow sub\_trace(x1,t,z1)$ <p>where the concatenation of <math>[x1, \dots, xm]</math> and <math>[y1, y2, \dots, yn]</math> is <math>[x1, \dots, xm, y2, \dots, yn]</math> if <math>xm = y1</math> and <math>[x1, \dots, xm, y1, y2, \dots, yn]</math> otherwise.</p>
<p><b>T4.</b> In the case where <math>x1</math> is an instance of a parallel branch point, the correctness of a sub-trace <math>t</math> from <math>x1</math> to <math>z1</math> is defined by the following rule<sup>11</sup>:</p> $inst(x1,x) \wedge inst(l1,l) \wedge inst(r1,r) \wedge par\_branch(x,l,r) \wedge inst(m1,m) \wedge sub\_trace(l1,t1,m1)$ $\wedge inst(s1,s) \wedge sub\_trace(r1,t2,s1) \wedge inst(y1,y), par\_mrg(m,s,y), interleaving(t1,t2,t)$ $\rightarrow sub\_trace(x1,t,y1)$ <p>where the predicate <math>interleaving(t1,t2,t)</math> holds iff <math>t</math> is a sequence such that: (i) the elements of <math>t</math> are the elements of <math>t2</math> together with the elements of <math>t2</math> and (ii) for <math>i=(1,2)</math> <math>x</math> precedes <math>y</math> in <math>ti</math> iff <math>x</math> precedes <math>y</math> in <math>t</math>.</p>

A trace is *correct* w.r.t. a well-formed business process schema  $B$  if it is conformant to  $B$  according to the intended semantics of the BPAL relational predicates (as informally described in Section 3). Below we present a formal definition of the notion of a correct trace. Let us first give some examples by referring to the example in Figure 1. Below we list two correct traces of the business process schema corresponding to the above BPMN specification:

$[s,r,i,pG,w,sG,e]$   
 $[s,r,i,w,pB,sG,e]$

<sup>11</sup> For sake of concision we omit the sets T5, T6 for the inclusive and exclusive branch points.

where  $inst(s, Start)$ ,  $inst(r, ReceivingPO)$ ,  $inst(i, Invoicing)$ ,  $inst(w, WaitingPayment Clearance)$ ,  $inst(pG, PreparingGift)$ ,  $inst(pB, PreparingBrochure)$ ,  $inst(sG, Sending Goods)$ ,  $inst(e, End)$ .

Note that the sub-traces  $[I, pG, w]$  of the first trace and  $[i, w, pB]$  of the second trace are the interleaving of the sub-trace  $[i, w]$  with the two branches going out from the exclusive branch point.

We now introduce a predicate  $trace(t)$ , which holds if  $t$  is a correct trace, with respect to a BP, of the form  $[s_1, s_2, \dots, s_n]$ , where  $s_1$  is an instance of a start event and  $s_n$  is an instance of an end event. The predicate  $trace(t)$  is defined by a set  $T$  of rules (in the form of Horn clauses), called *trace rules*. These rules have a double nature, since they can be used to check correctness but also for generating correct traces. Each trace rule corresponds to a *process composition rule* and, for lack of space, in Table 5 we list only the trace rules corresponding to the composition rules presented in Section 4. The trace axioms are defined by induction on the length of the trace  $t$ .

We say that a trace  $t$  is *correct* w.r.t. a BPAL BP schema  $B$  if  $trace(t)$  can be inferred from  $B \cup T$ .

## 6 The BPAL Platform

In this section we briefly present the logical architecture of the BPAL platform with the key implemented reasoning services: (1) verification of the well-formedness of a BPS, (2) trace analysis, and (3) trace generation.

In the BPAL platform the verification of well-formedness of a BPS is performed as depicted in Figure 2. The BPMN graphical specification of a given BP-1 business process is exported as XPDL<sup>12</sup> (XML Process Definition Language) and translated into a set of BPAL ground facts by means of the service XPDL2BPAL, thereby producing the BPS  $B$ . Then  $B$  is passed to the reasoning engine together with the meta-model  $MM$ , i.e., the set  $F$  of composition rules, the set  $K$  of constraints, and the set  $I$  of inclusion axioms. Thus, the union of  $B$ ,  $F$ ,  $I$ , and  $K$  makes up a knowledge base from which we want to infer that  $B$  is well-formed. This inference task is performed by the BPAL reasoning engine which is implemented by using the XSB logic programming and deductive database system [16]. XSB extends the conventional Prolog systems with an operational semantics based on *tabling*, i.e., a mechanism for storing intermediate results and avoiding to prove sub-goals more than once. XSB has several advantages over conventional Prolog systems based on SLDNF-resolution (such as, SWI-Prolog and SICStus Prolog): (i) in many cases XSB is more efficient than conventional systems, (ii) it guarantees the termination of queries to DATALOG programs (i.e., Prolog programs without function symbols), and (iii) it often avoids to return several times the same answer to a given query.

$B \cup F \cup I$  is a set of Horn clauses and, therefore, it is translated to a Prolog program in a straightforward way. The schema constraints in  $K$  are translated to Prolog queries. For instance, the domain constraint  $activity(x) \rightarrow \neg event(x)$  is translated to the query:

---

<sup>12</sup> <http://www.wfmc.org/xpdl.html>

```
?- activity(X), event(X).
```

For the eProcurement example of Figure. 1, XSB answers ‘no’ to this query, meaning that there is no  $X$  which is an activity and an event at the same time. Hence, this domain constraint is inferred from  $B \cup F \cup I$ .

Moreover, for any given start event  $start$  and end event  $end$ , we can check whether or not  $wf\_proc(start, end)$  is inferred from  $B \cup F \cup I$ , by running the query:

```
?- wf_proc(start, end).
```

For the eProcurement example of Figure. 1, XSB answers ‘yes’ to this query, meaning that  $wf\_proc(start, end)$  is inferred from  $B \cup F \cup I$  and, thus, the well-foundedness of  $B$  is verified.

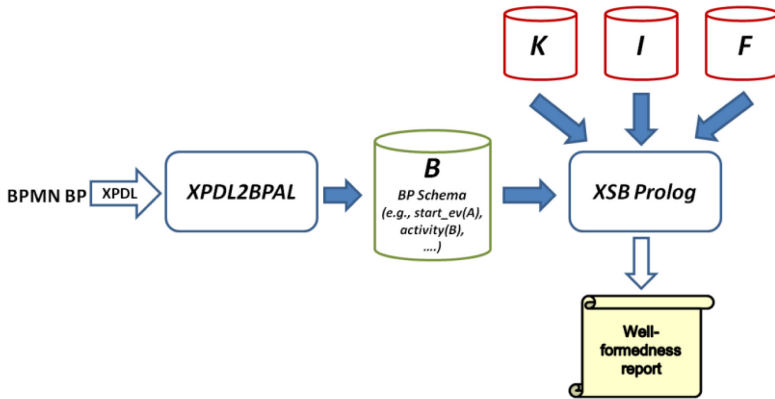


Fig. 2. Architecture of the well-formedness service

The trace analysis and trace generation services are performed by translating the theory  $T \cup B$  to a Prolog program in the reasoning engine. As above, this translation is straightforward, as  $T \cup B$  is a set of Horn clauses. The trace analysis service consists in checking whether or not a given trace  $t$  is correct w.r.t. the well-formed BPS  $B$ . This task is performed by the reasoning engine by running a query of the type  $trace(t)$ , where  $t$  is the trace to be checked (Figure 3.a). For instance, in the eProcurement example, XSB behaves as follows:

```
?- trace([s, r, i, pG, w, sG, e]).
yes
?- trace([s, r, i, pG, pB, w, sG, e]).
no
```

Indeed, the first sequence is a correct trace and the second is not.

The trace generation service consists in generating all correct traces (Figure. 3.b). This task is performed by running a query of the type  $\text{trace}(T)$ , where  $T$  is a free variable (Figure 3.a). In the eProcurement example, XSB behaves as follows:

```
?- trace(T).
T = [s,r,pG,i,w,sG,e];
T = [s,r,i,w,pG,sG,e];
T = [s,r,i,w,pB,sG,e];
T = [s,r,i,pG,w,sG,e];
T = [s,r,i,pB,w,sG,e];
T = [s,r,pB,i,w,sG,e];
no
```

meaning that the above sequences are all and only the correct execution traces of the given BPS.

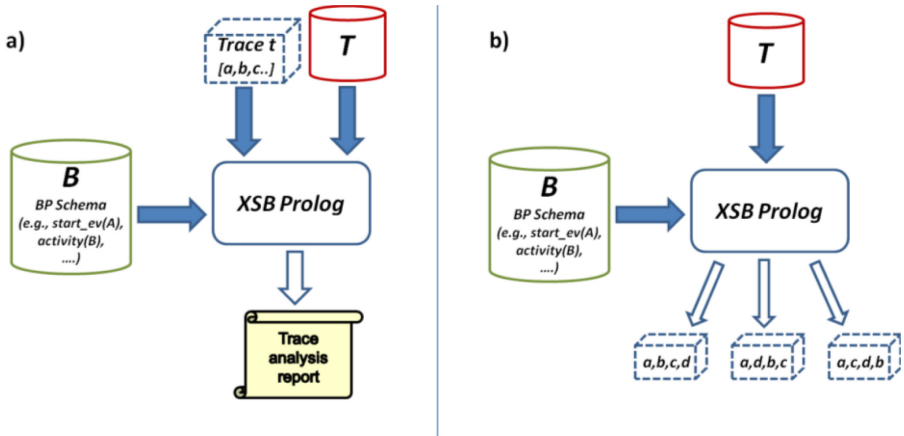


Fig. 3. a) Architecture of the trace analysis and b) traces generation service

## 7 Conclusions and Future Works

In this paper we presented a platform to complement existing business modeling tools by providing advanced reasoning services: the well-formedness verification service, the trace analysis and the trace generation service. The platform is centered around the logic-based BPAL language. A first evaluation of the services in the eProcurement domain shows the viability of the approach.

We intend to expand the BPAL platform in several directions. A first direction will be the tight integration of business ontologies (i.e., structural knowledge) represented by OPAL, and the behavioral knowledge, represented by BPAL. OPAL is an ontology representation framework supporting business experts in building a structural

ontology, i.e., where concepts are defined in terms of their information structure and static relationships. In building an OPAL ontology, knowledge engineers typically start from a set of upper level concepts, and proceed according to a paradigm that highlights the active entities (actors), passive entities (objects), and transformations (processes). The latter are represented only in their structural components, without modeling the behavioral issues, delegated to BPAL. As shown in [14], a significant core of an OPAL ontology can be formalized by a fragment of OWL, relying within the OWL-RL profile [19,20] an OWL subset designed for practical implementations using rule-based technologies such as logic programming.

Another direction concerns the modeling and the verification of Business Rules (BRs). This is motivated by the fact that in real world applications the operation of an enterprise is regulated by a set of BPs that are often complemented by specific business rules. We intend to enhance the BPAL platform so as to support the integrated modeling of BPs and BRs. New reasoning services will also be needed for analyzing those integrated models to check if, for instance, there are possible executions of processes that violate any given business rule.

Since BPs play a growing role in business realities, we foresee a scenario where huge repositories of process models developed by different designers have to be managed. In such a scenario there will be the need for advanced reasoning systems aimed at query processing, for the retrieval of process fragments to be used in the design of new BP models, and at verifying that some desired properties hold. We intend to enhance the BPAL platform in such a way that we can issue several types of queries, both at intensional and extensional level. In particular, we are interested in the following three types of queries and combinations thereof. (1) Queries over BP schemas. Querying the BPS allows the search for certain patterns adopted in the design phase and the verification of constraints that descend from structural requirements to be done. (2) Queries over BP traces. Here the behavior at execution time is of interest, and the properties to be verified regard the temporal sequencing of activities in the set of correct traces. (3) Queries over the Business Ontology. Here the focus is on the domain entities (processes, actors, objects) and their relationships.

Finally, on an engineering ground, we intend to investigate the problem of Business Process Reengineering, and explore the possibility of manipulating a set of business processes to produce a new, optimized (e.g., in terms of process length or aggregating sub-processes that are shared by different BPs) set of reengineered BPs.

## References

1. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and automated analysis of BPMN process models. Preprint 7115. Queensland University of Technology, Brisbane, Australia (2007)
2. Reisig, W., Rozenberg, G. (eds.): APN 1998. LNCS, vol. 1491. Springer, Heidelberg (1998)
3. Pesic, M., van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)



4. Montali, M., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verification from Declarative Specifications Using Logic Programming. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 440–454. Springer, Heidelberg (2008)
5. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the SCIFF framework. *ACM Transactions on Computational Logics* 9(4), 1–43 (2008)
6. Conrad, B., Gruninger, M.: Psl: A semantic domain for flow models. *Software and Systems Modeling* 4(2), 209–231 (2005)
7. OMG: Business Process Model and Notation. Version 2.0 (August 2009), <http://www.omg.org/spec/BPMN/2.0>
8. De Nicola, A., Lezoche, M., Missikoff, M.: An Ontological Approach to Business Process Modeling. In: 3rd Indian International Conference on Artificial Intelligence (IICAI 2007), Pune, India (Dicembre 17 -19, 2007)
9. Lloyd, J.W.: *Foundations of Logic Programming*. Springer, Berlin (1987)
10. Dumas, M., van der Aalst, W., ter Hofstede, A.H.M.: *Process-Aware Information Systems*. Wiley-Interscience, Hoboken (2005)
11. Eder, J., Gruber, W.: A Meta Model for Structured Workflows Supporting Workflow Transformations. In: Manolopoulos, Y., Návrát, P. (eds.) ADBIS 2002. LNCS, vol. 2435, pp. 326–339. Springer, Heidelberg (2002)
12. Combi, C., Gambini, M.: Flaws in the Flow: The Weakness of Unstructured Business Process Modeling Languages Dealing with Data. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009. LNCS, vol. 5870, pp. 42–59. Springer, Heidelberg (2009)
13. van Dongen, B.F., Mendling, J., van der Aalst, W.M.P.: Structural Patterns for Soundness of Business Process Models. In: Proceedings of EDOC 2006, Hong Kong, China. IEEE, Los Alamitos (2006)
14. D’Antonio, F., Missikoff, M., Taglino, F.: Formalizing the OPAL eBusiness ontology design patterns with OWL. In: Third International Conference on Interoperability for Enterprise Applications and Software, I-ESA (2007)
15. Scheer, A.W., Thomas, O., Adam, O.: Process Modeling Using Event-Driven Process Chains. In: Dumas, M., van der Aalst, W., ter Hofstede, A.H.M. (eds.) *Process-Aware Information Systems*, pp. 119–145 (2005)
16. The XSB Logic Programming System. Version 3.1 (August 2007), <http://xsb.sourceforge.net>
17. Bonner, A.J., Kifer, M.: Concurrency and Communication in Transaction Logic. In: Joint International Conference and Symposium on Logic Programming (1996)
18. Roman, D., Kifer, M.: Reasoning about the Behavior of Semantic Web Services with Concurrent Transaction Logic. In: VLDB (2007)
19. OWL 2: Profiles, <http://www.w3.org/TR/owl2-profiles>
20. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. In: Proceedings of the 12th International Conference on World Wide Web. ACM, New York (2003)

# Discovering Highly Informative Feature Sets from Data Streams

Chongsheng Zhang\* and Florent Masseglia

AxIS team, INRIA, 2004 Route des lucioles, 06902 Sophia-Antipolis, France  
chongsheng.zhang@yahoo.com, florent.masseglia@inria.fr

**Abstract.** How to select interesting feature sets from data streams is a new and important research topic in which there are three major challenges. First of all, instead of discovering features individually and independently, we are interested in comprehensively selecting a subset of features whose joint importance or weight is the highest. Secondly, we are concerned with the problem of selecting feature sets over dynamic, large and online data streams which are only partly available when we are selecting the features. This problem distinguishes itself over the data streams from the ones on the static data which is completely available before the feature selection. Finally, data streams may evolve over time, requiring an online feature selection technique which can capture and adapt to such changes. We introduce the problem of online feature selection over data streams and we provide a heuristic solution. We also demonstrate the effectiveness and efficiency of our method through experiments on real-world mobile web usage data.

## 1 Introduction

Feature selection is the task of selecting interesting or important features, and removing irrelevant or redundant ones. There are a lot of existing works on feature selection [7]. However, feature selection is usually task and application relevant, especially for the unlabeled data. And depending on different interests and applications, we may have different interestingness measures to assess the significance of the features or feature sets [8]. For instance, for utility based measures, given a utility function, we select features or feature sets with high utility scores; the measure could also be coverage-based in which we select the most frequent features or feature sets with the highest frequency. Unfortunately, existing feature selection algorithms were designed towards static data, and are not adaptable to data streams.

A Data stream is an infinite sequence of data which flows dynamically and rapidly. Moreover, data streams are usually of large volumes, making it difficult to extract expected patterns or summaries. Online feature set selection over data streams is a new and important research issue but it has received little attention.

---

\* This work was partially funded by ANR, grant number ANR-07-MDCO-008-01/MIDAS.

Nevertheless, it is significant and meaningful because it can help us select the most important features from huge data streams. To tackle the problem, we have to overcome several difficulties. Firstly, we must provide an incremental feature selection algorithm which fits to data streams; secondly, data streams could contain so many features with changing weights over time that we should take the evolving weighted values of features (and feature sets) into account when we are selecting the feature sets. Furthermore, in data streams, new features may appear and some features could become obsolete. As a result, the attributes of related feature sets, which we are analyzing, will be influenced. In other words, not only the weights of an feature set, but also its attributes may vary with time. In short, it is very challenging to develop an intelligent algorithm which can self-adaptively discover the important feature sets as the data flows and evolves. In this paper, we are interested in information theory based measure and we will select the most informative feature sets from data streams. This method could possibly be used in various applications such as classification and information retrieval. Example 1 addresses the problem of feature selection for document retrieval.

*Example 1.* In this application, we would like to retrieve documents from table 1, in which the columns of  $O_1, O_{10}$  are 10 documents, and the attributes of  $A, B, C, D, E$  are some features (key words) in the documents, where the value “1” means the the feature is contained in the document, and “0” not. It is easy to find that  $(D, E)$  is a frequent featureset, because features  $D$  and  $E$  occur together in nearly every document. However, it provides little help for document retrieval. By contrast,  $(A, B, C)$  is an infrequent featureset, as its member features rarely or never appear together in the data. And it is troublesome to summarize the value patterns of featureset  $(A, B, C)$ . Providing it with the values  $\langle 1, 0, 0 \rangle$  we could find the corresponding document  $O_3$ ; similarly, given the values  $\langle 0, 1, 1 \rangle$  we will have the according document  $O_6$ . Although  $(A, B, C)$  is infrequent, it contains lots of useful information which is hard to summarize. We call it an informative feature set.

**Table 1.** Features in the documents

Features	Documents									
	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$	$O_8$	$O_9$	$O_{10}$
A	1	1	1	1	1	0	0	0	0	0
B	0	1	0	0	1	1	0	1	0	1
C	1	0	0	1	0	1	1	0	1	0
D	1	0	1	1	1	1	1	1	1	1
E	1	1	1	1	1	1	1	1	1	1

From the above example, we can see that it is useful and helpful to discover informative feature sets. Information theory provides strong supports to measure the informativeness of the feature sets. Unfortunately, it requires us to compute the probabilities of all the features and possible feature subsets. Thus,

it is extremely time consuming and exhaustive, and the case is much worse for streaming data, because the probabilities of the features and feature sets are always changing. To solve this problem, we introduce the heuristic *StreamHI*. It is based on a candidate generation principle and a pruning principle. The former will keep and monitor as many candidates as possible, while the latter will remove the redundant and hopeless ones. Our contributions are i) a definition of the problem of online informative feature set selection from the data streams ii) *StreamHI*, a heuristic method for mining the informative feature sets in real time. We also run *StreamHI* against naive methods through a series of experiments which demonstrate the superiority and effectiveness of our proposal.

We will first give a formal definition of informative feature sets in section 2. Section 3 is an overview of the related works. Afterwards, data management techniques, and two heuristic algorithms for informative feature set mining are presented in section 4. They will be evaluated in section 5.

## 2 Definitions

This section aims at proposing formal definitions of transaction data streams and the problem of mining informative featuresets.

### 2.1 Preliminary Definitions

**Definition 1.** A **transaction** is a tuple  $\langle oid, time, feature \rangle$  where *oid* is a user with unique identifier, *time* is a timestamp and *feature* is the feature associated to *oid* at that timestamp. A **transaction data stream** is a stream of transactions. Be noted that, several tuples would share the same identifier although they arrive in different time. Thus, one identifier's complete profile (item set) may be spread across several related tuples. Namely, to summarize each user's itemset, we will have to search all the tuples for those sharing the same identifiers.

**Definition 2.** A **featureset** is a set composed of one or more features. The size of a featureset is the number of unique features in the featureset.

*Example 2.* Let  $T_1 \langle o_1, d_1, f_1 \rangle$  be a transaction, which associates feature  $f_1$  to object  $o_1$  at time  $d_1$ . Let  $DS_1 = \{T_1 \langle o_1, d_1, f_1 \rangle, T_2 \langle o_2, d_1, f_1 \rangle, T_3 \langle o_1, d_2, f_2 \rangle, T_4 \langle o_3, d_2, f_4 \rangle, T_5 \langle o_1, d_3, f_6 \rangle, T_6 \langle o_2, d_4, f_2 \rangle\}$  be a subset of  $DS$ , a data stream.  $DS_1$  is a transaction data stream and  $(f_1, f_2, f_6)$  is the featureset of size 3 corresponding to  $o_1$ . If we let both the size of featuresets and the support be 2, then  $(f_1, f_2)$  is a frequent featureset, as both objects  $o_1$  and  $o_2$  selected  $f_1$  and  $f_2$ .

**Definition 3.** The **entropy** of a feature, in the data stream, is a measure of the expected amount of information needed to specify the state of uncertainty or disorder for this feature. Let  $X$  be a feature,  $P(X = n)$  is the probability that  $X$  has value  $n$  in the data (we consider categorical data, where the value will be '1' if the object has the feature and '0' otherwise). The entropy of feature  $X$  is given by  $H(x) = -(P(X = 0)\ln(P(X = 0)) + P(X = 1)\ln(P(X = 1)))$

**Definition 4.** *The joint entropy of a couple/pair of features  $(X, Y)$ . Let  $X$  and  $Y$  be two features and let  $x$  and  $y$  be the two possible values respectively. Let  $P(x, y)$  the probability that  $(X, Y)$  has values  $(x, y)$  in the stream. The joint entropy for  $(X, Y)$  is given by  $H(X, Y) = -\sum_{x, y \in \{0, 1\}^2} P(x, y) \ln(P(x, y))$  Where  $\{0, 1\}^2$  is the set of possible couples of 0 and 1.*

$$\text{MAX}(H(X), H(Y)) \leq H(X, Y) \leq H(X) + H(Y) \quad (1)$$

Formula 1 illustrates the monotonicity of joint entropy: the more features involved, the larger the joint entropy. The difference is that with some features we can have much more information gain, while with other features we can have just a little or no increase.

**Definition 5.**  *$H(I)$ , the joint entropy of a featureset  $I$  in the stream (or entropy of  $I$ ), measures the amount of information given by this featureset. Let  $\{0, 1\}^{|I|}$  be the set of all possible combinations of values for featureset  $I$ . The entropy of  $I$  is calculated thanks to  $P(I = C)$  for each combination  $C$  in  $\{0, 1\}^{|I|}$  where  $(I = C)$  is the instantiation of  $I$  with the values of  $C$ .  $H(I)$  is given by formula 2.*

$$H(I) = - \sum_{C \in \{0, 1\}^{|I|}} P(I = C) \ln(P(I = C)) \quad (2)$$

By definition,  $0 \leq H(X) \leq 1$  with  $X$  a feature and  $H(I) \geq 0$  with  $I$  a featureset. The higher  $H(I)$ , the more information we gain from it.

## 2.2 Problem Definition

**Definition 6.** *An Informative Featureset in the data stream is a featureset which is informative, according to its joint entropy. The larger the joint entropy of a featureset, the more informative it is.*

**Definition 7.** *A Highly Informative Featureset (HI) is the  $k$ -featureset that has the largest entropy value, as expressed by formula 3.*

$$H(HI) = \text{MAX}\{H(IS_k), IS_k \in SS_k\} \quad (3)$$

Where  $IS = \{I_1, I_2, \dots, I_n\}$  is the set of all  $n$  possible features,  $IS_k = \{I_{m_1}, I_{m_2}, \dots, I_{m_k}\}$  is a  $k$ -featureset (with  $k < n$ ),  $SS_k = \{IS_k\}$  is the set of all possible featuresets of size  $k$ .

To discover the highly informative itemset over data streams, the first challenging problem is how to discover and summarize the patterns on the fly over the data streams although the records of each identifier arrive separately. The second challenge is, in a transaction data stream, how to online extract the Highly Informative featureset (HI) and dynamically update them when the data stream evolves. To the best of our knowledge, this is the first proposal for this interesting problem.

### 3 Related Works

Featureset mining is a very important topic that has received much attention for static data [13, 6, 5] and for data streams [9, 2].

#### 3.1 Featureset Mining and Entropy in Static Data

The authors of [5] propose two algorithms designed to extract the sets of both low entropy and high entropy featuresets from static databases. They provide a strong theoretical background and an analysis of joint entropy properties that allow fast computation of featuresets i) in a level-wise approach for low entropy sets (based on monotonicity of entropy) and ii) by means of Bayes networks for high entropy sets. However, as the authors point out, the size of the output can be very large. Even with a tree structure, designed to lower this size, the number of extracted trees varies from 532 to 44156, in a dataset of 2405 observations and 5021 features. It is thus very important to filter out that result. [6] proposes a heuristic algorithm designed to extract only one featureset of size  $k$  with the highest possible entropy. Such featuresets are called *miki*. The *ForwardSelection* algorithm performs multiple scans over the dataset. Within each scan, *miki*'s size increments by 1, by adding a new feature  $f$  with which the new *miki* achieves the highest entropy. The authors show the advantage of *ForwardSelection* over the brute force algorithms that evaluate the entropies of all the possible subsets of size  $k$ .

#### 3.2 Mining Frequent Featuresets in Data Streams

Let  $T$  be a batch of transactions and  $\sigma$  be a user threshold. A frequent featureset (or *frequent itemset* in the literature) is a set of features that occurs in at least  $\sigma \times |T|$  transactions in  $T$ . In [2], the authors propose FPStream, an algorithm based on a FP-Tree data structure where FPGrowth [3] is applied to each batch of transactions in a data stream. After the frequent patterns in each batch have been extracted, they are stored into the FP-Tree and their history is kept with a decaying factor (recent events are stored with a finer granularity). In [9] the authors devised *FTP-DS*, a regression-based algorithm to mine frequent temporal patterns over data streams. Primarily, the data is managed in a sliding window model and the variations of support of each pattern are transformed into numerical time series. Thereafter, in order to meet the space constraint, the time series for each pattern is compacted into a representative synopsis and a temporal regression is adopted to approximately model the relation between the support of the pattern and time.

## 4 Mining Highly Informative Featuresets from Data Streams

In this section, we will present how to extract the *HI*s from the transaction data streams. We will first give *NaiveHI*, a heuristic algorithm relying on

*ForwardSelection* [6] in a batch environment. After the analysis of the drawbacks of *NaiveHI*, an improved heuristic algorithm, *StreamHI*, will be introduced.

#### 4.1 Naive Mining Algorithm *NaiveHI*

*NaiveHI* (Figure 1) is based on the following principle: the transaction data stream is divided into batches and for each batch, we perform *ForwardSelection* [6]. The idea of *ForwardSelection* is, with the initial size starting from 1, to iteratively scan the data and enlarge the *HI* until its size reaches  $k$ ; At each step  $i$ , *ForwardSelection* selects the featureset having the largest entropy among all the candidates. At step  $i + 1$ , the *HI* of step  $i$  will be used as the base featureset in order to generate the new candidates. The generating principle is to add one more feature to the current *HI*. For instance, let  $\{A, B, C, D, E, F, G\}$  be the feature set and let the current *HI* be  $(A, B)$ , then the set of candidates will be  $\{(A, B, C), (A, B, D), (A, B, E), (A, B, F), (A, B, G)\}$ .

**Heuristic algorithm:** *NaiveHI*

**Input:**  $DS$ , a transaction data stream,  $k$ , the desired length of *HI* and  $M$ , the batch size

**Output:** For each batch of size  $M$ , the corresponding *HI* having length  $k$ .

**For each batch in  $DS$  Do**

$HI \leftarrow ForwardSelection(batch, k)$ .

**Done**

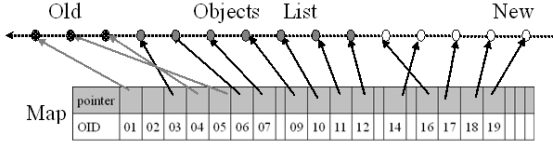
**End *NaiveHI***

**Fig. 1.** Heuristic algorithm *NaiveHI*

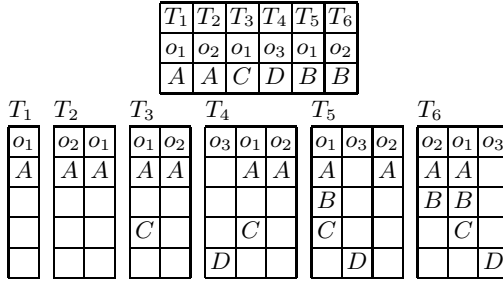
We can see that for *NaiveHI*, we have to determine a batch size, worse still, as it is transaction data streams, the patterns of objects could be truncated by batches and the connections between batches are cut. Accordingly, the informative featureset selected by *NaiveHI* may not be the real one. In the following section, we will first introduce a data structure which can help us manage the transaction data streams.

#### 4.2 Transaction Data Stream Management

Our data structure allows observing the stream by accumulating transactions in the largest possible time period. *ASW* (Accumulating Sliding Window), our structure, is based on a FIFO (First In First Out) principle to manage the objects. The data structure of *ASW* is described in Figure 2. *ASW* has two main components: an Objects List(Window) and a hash table with user ID as the key. Object list maintains the objects in a FIFO way, and each object carries the data for the corresponding user. In order to quickly check whether the identifier already existed in the current list/window, we keep a hash table, whose key is the identifier, and the value is the according object in the list, thus the hash table will take on the responsibility of searching for the according object in the list through its user ID in the hash table. Each object  $o_i$  in *ASW* is represented



**Fig. 2.** Data Structure of ASW



**Fig. 3.** A transaction data stream and its representation in ASW

by a node containing a hash table that indexes the features associated to  $o_i$ . Notice that, for the objects list, once an object is updated, it will be moved from its current position to the end of the list. When a new transaction  $T_i = \langle object_o, date_d, feature_f \rangle$  arrives, there are two options. If  $object_o$  already exists in ASW, then it is updated with  $feature_f$  and it is moved to the beginning (head) of ASW. Otherwise,  $object_o$  is created at the beginning of ASW and its hash table only indexes  $feature_f$ . When the size of ASW is exceeded, the last object (tail of ASW) is removed. Example 3 explains how ASW manages the data in transaction data streams.

*Example 3.* Let us consider the data in the upper table of figure 3. Other tables in the figure correspond to the sequential status of the objects' patterns in ASW after the orderly arrivals of transactions from  $T_1$  to  $T_6$ . For instance, the last table represents the status of objects in ASW after having read transaction  $T_6$ .

Clearly, the patterns (item sets) for the objects/users ( $o_1, o_2$  and  $o_3$ ) are changing as new transaction comes, but ASW can help capture and add the new features to the patterns of the objects. Next we will describe how *StreamHI* discovers highly informative featuresets in ASW.

### 4.3 Streaming Algorithm *StreamHI*

The goal of *StreamHI* (Figure 4) is to maintain the *HI* on the fly in ASW, with the precondition that we manage one or several data structures which record:

1. the entropy of each feature.
2. the joint entropy of each pair of features.



**Heuristic algorithm:** StreamHI

**Input:**  $DS$ , a transaction data stream,  $k$ , the desired length of  $HI$   
 $M$ , the size of  $ASW$  and  $F$ , the set of features

**Output:**  $HI$ , the Highly Informative Featureset having length  $k$ .

1. Initialize  $HI$  of length  $k$  using certain algorithm such as *ForwardSelection*
2.  $Top_F \leftarrow k+1$  features that have the highest entropies and that do not occur in  $HI$
3.  $CI \leftarrow GenerateCand(HI, Top_F)$
4. **While** not end of the stream **Do**
  - (a) Read the next transaction  $\langle cid, time, f \rangle$
  - (b) Update or create  $cid$  in  $ASW$  with  $f$
  - (c) Maintain the entropy statistics of  $f$  and  $(f, g) \forall g \in F$
  - (d) **If**  $ASW$  is full **Then**
    - i. remove the last object of  $ASW$
    - ii. update the entropy of each  $f \in F$  and each  $(f, g) \in F \times F$
  - (e)  $Top_F \leftarrow$  top  $k+1$  features having the highest entropies (and not in  $HI$ )
  - (f) **If**  $Top_F$  has changed **Then**
    - i.  $HI \leftarrow$  featureset having the highest entropy in  $CI$
    - ii.  $CI \leftarrow GenerateCand(HI, Top_F)$
    - iii. *CandPruning*( $CI$ )
    - iv. Scan  $ASW$  and evaluate the entropy of each featureset in  $CI$
5. **Done**

**End** StreamHI

**Fig. 4.** Heuristic algorithm *StreamHI*

Initially, we will use certain algorithm such as *ForwardSelection* to have a first  $HI$ . Based on this initial  $HI$ , we will generate the candidates. As described in subsection 4.2, each time we read a new transaction  $\langle cid, time, f \rangle$ , we have to check whether  $cid$  already exists in  $ASW$ . After that we will maintain the entropy of each feature and each pair of features.

The most time consuming part of *StreamHI* is the evaluation of candidates entropy. Therefore, it is desirable to reduce their number and the frequency of their evaluations over  $ASW$ . To that end, we rely on two principles:

1. New Candidates are evaluated only if there is a change in the top- $k$  features (sorted by entropy). That step corresponds to line (e) in *StreamHI*.
2. Candidates that cannot be the new  $HI$ , because of the bounds on their entropy, are filtered out thanks to our pruning strategy.

Unlike *NaiveHI*, which relies on the batch environment, *StreamHI* allows, at any time, incremental updates of the candidate featuresets in order to extract the one having the highest entropy (*i.e.*  $HI$ ).

The method for generating  $HI$  candidates is important, for both the efficiency and the effectiveness. Actually, if there are not enough candidates, we may miss the real  $HI$ . However, if we have too many candidates, we have to spend too much time and computation for maintaining their entropy statistics and evaluating their values. Our candidate generation algorithm, *GenerateCand* is described

**Algorithm:** GenerateCand  
**Input:**  $HI$ , a featureset and  $Top_F$ , a set of features.  
**Output:**  $CI$ , a set of candidate featuresets.  
 $CI \leftarrow \{HI\}$   
**Foreach**  $f$  in  $Top_F$  **Do**  
    **For**  $i=1$  to  $|HI|$  **Do**  
         $Cand \leftarrow HI$   
         $Cand[i] = f$   
         $CI \leftarrow CI \cup \{Cand\}$   
    **End for**  
**End foreach**  
**End** GenerateCand

**Fig. 5.** Algorithm *GenerateCand*

by Figure 5. The generating principle is based on  $Top_F$ , the top- $k$  (or top- $mk$ ,  $m \leq k$ ) features which are not included in the current  $HI$ . Iteratively, each feature of  $HI$  will be replaced by a feature of  $Top_F$ . By using the top features (sorted by decreasing entropy) we aim at proposing candidates that will have the highest possible entropy.

The time complexity for generating the candidates is  $O(m * k * |HI|)$ ,  $|HI|$  is the number of features in  $HI$ . The hypothesis under *GenerateCand* is that,  $HI$  will be gradually replaced, and the individual features with higher entropies are more likely to have greater contributions to the overall entropy of the featureset it resides in. However, this is not always the case. As an example, let features  $A$  and  $B$  have the same (large) entropy, but let them always appear together, such that the entropy for the pair of  $(A, B)$  is almost the same as that of  $A$ , therefore, although  $B$  has high entropy, it does not contribute to the entropy of  $(A, B)$ . To include as many important features as possible, and to lower the risk of missing the real  $HI$ , when we are generating the  $HI$  candidates, we also take into account the features in the top- $k$  (or top- $mk$ ) featuresets of size 2 (a pair of features). Even though this remedy still can not guarantee the accurate result, it can greatly reduce the risk. That is why we keep extra data structures to monitor and compute the entropies of pair features. We would like to mention that it is heuristic to generate the candidates, but once the candidates are determined, we will monitor and compute their exact entropy values.

Another hypothesis is, assuming that the information shared among features remains the same if the top- $k$  features do not change, and the rank of entropies for the featuresets will not change consequently, the  $HI$  will be the same one. To strengthen it, we can add a second condition that the top- $k$  (or top- $mk$ ) featureset of size 2 do not change. In the end, this hypothesis can help us decrease the computation costs for discovering  $HI$ .

#### 4.4 Pruning Strategy

One possible problem with candidate generation is that, we would have too many candidates to evaluate, which is costly. Therefore, we need a strategy to detect

and prune irrelevant candidates. We introduce a pruning technique, based on the mutual information between two features.

**Definition 8.** The **Mutual Information** ( $MI$ ) measures the mutual dependence of two features,  $X$  and  $Y$ . It is given by formula [4](#).

$$MI(X, Y) = \sum_{y \in Y} \sum_{x \in X} P(x, y) \ln \frac{P(x, y)}{P(x)P(y)} \tag{4}$$

where  $P(x, y)$  is the joint probability distribution function of  $X$  and  $Y$ , and  $P(x)$  and  $P(y)$  are the probability distribution functions of  $X$  and  $Y$  respectively. The mutual information between two discrete variables can be equivalently given by  $MI(XY) = H(X) + H(Y) - H(X, Y)$ . In order to estimate the lower and upper bounds on the joint entropy of a feature set, we use the following inequality based on the proofs of [10,4](#):

$$\sum_{i=1}^n H(X_i) - H(X_1, X_2, \dots, X_n) \geq \frac{1}{n-1} \sum_{i < j} MI(X_i, X_j) \tag{5}$$

Based on formula [5](#), we can infer the upper (formula [6](#)) and lower (formula [7](#)) bounds for  $H(X_1, X_2, \dots, X_n)$ .

$$H(X_1, X_2, \dots, X_n) \leq \sum_{i=1}^n H(X_i) - \frac{1}{n-1} \sum_{i=1}^n \sum_{j>i} MI(X_i, X_j) \tag{6}$$

$$H(X_1, X_2, \dots, X_n) \geq MAX(H(X_i, X_j)) \tag{7}$$

Once we are provided with the lower and upper bounds on the entropies of the candidate featuresets, we can use the following lemma to reduce the set of candidates.

**Lemma 1.** If the upper bound of a candidate  $C$  is lower than the maximum lower bound of all the candidates, then  $C$  cannot be the new  $HI$ .

Lemma [1](#) is based on the comparison of candidates with each others. Let  $Upper(I)$  and  $Lower(I)$  be the upper and lower bounds of a featureset  $I$ . Let  $W \in CI$  be the candidate having the highest lower bound. Let  $C \in CI$  be a candidate such that  $Upper(C) < Lower(W)$ . Then,  $W$  will always have a higher entropy than  $C$ . Therefore,  $C$  can be removed from the set of candidates. According to Lemma [1](#), we can reduce the number of candidates as described in algorithm *CandPruning* in Figure [6](#).

### 4.5 Discussion

**Candidate Generation.** Because of the combination curse when the feature number is large, *ForwardSelection* is introduced to select the highly informative featuresets and it has been validated through experiments [6](#). However, it

**Algorithm:** CandPruning

**Input:**  $CI$ , a set of featuresets.

**Output:**  $CI$ , where the irrelevant candidates have been removed.

$MaxLow \leftarrow$  the maximum lower bound of a featureset in  $CI$ .

**Foreach**  $C$  in  $CI$  **Do**

**If**  $UpperBound(C) < MaxLow$  **Then**

$CI \leftarrow CI \setminus \{C\}$

**End** CandPruning

**Fig. 6.** Algorithm *CandPruning*

could not adapt to the streaming data. *StreamHI* generates and evaluates several *HI* candidates, and maintains the according statistics for the computation of entropy, in order to avoid the multiple scans which *ForwardSelection* suffered from. In order to be efficient, *StreamHI* adopts another heuristic strategy: if the set of features having the highest entropies do not change, we believe that the previous *HI* does not change; otherwise, we will generate new candidates. Moreover, our pruning strategy allows for weak candidates detection and avoids unnecessary evaluations.

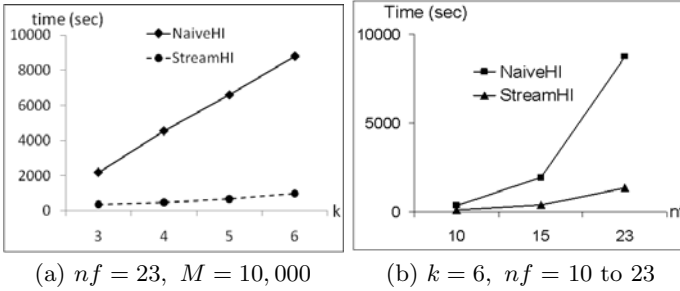
**Obtaining the Entropy of a Featureset.** For both *NaiveHI* and *StreamHI*, we need an optimal scan of *ASW* in order to evaluate the entropy of each candidate. Since  $0 \log 0 = 0$ , it is not compulsory to compute the probability of all the possible combinations (i.e.  $C \in \{0, 1\}^{|I|}$ ) for a featureset  $I$ , as proposed by formula 2. Actually, we just have to count the probabilities of existing combinations for featureset  $I$  in *ASW*. At the end of the scan, we can compute  $H(I) = -\sum_{C \in \{0, 1\}^{|I|}} P(I = C) \ln(P(I = C))$  by accumulating the probabilities of existing combinations, thus avoiding the enumeration of all the possible combinations.

## 5 Experiments

In this section we discuss the experimental evaluations for the algorithms, with efficiency and effectiveness as the evaluation criteria. All the programs were implemented in C++ and ran on a PC with Fedora7 operating system, an Intel 2.35 GHz CPU and a 3 GB main memory.

### 5.1 Data

We have a 24 GB real usage data from Orange (European mobile service provider). The data is a log of 3 months portal visits by the clients through their mobile devices, from May 2008 to July 2008. As the raw data contains 3 integer and 4 string features, by removing 4 uninteresting features and replacing datetime string features with integer feature, we finally have a 1.2 GB data, with each line a transaction record  $\langle cid, time, page \rangle$ , where *cid* is the unique client number, *time* is the transaction time and *page* is the page requested by the client. All these three attributes are in integer formats, and the feature values are in range of [1, 23]. The 1.2 GB data is the final dataset we will use for experimental evaluations.



**Fig. 7.** Execution times for *NaiveHI* and *StreamHI* depending on the size of the discovered featureset (a) and on the number of features in the dataset (d)

## 5.2 Evaluation Results

As there exists no algorithm we could refer to, we will compare the two heuristic algorithms. We will use the running time as the measure of efficiency, and effectiveness is measured by precision based on *NaiveHI*. Meanwhile, parameters will be taken into account when we are evaluating the methods. We have two user input parameters:  $k$  is the size of the *HI* and  $M$  stands for the size of the batch (in *NaiveHI*) or the size of *ASW* (in *StreamHI*). We will also consider  $nf$ , the number of features in the data stream, and observe its impact on the response time.

**Efficiency.** The diagrams of Figure 7 give the difference of efficiency between *NaiveHI* and *StreamHI* from two points of view. On the one hand, Figure 7 (a) gives the time response of both algorithms with a varying size of  $k$ , the size of the featureset to be discovered. We can observe that a larger value of  $k$  implies higher response times and, under the same value of  $k$ , *StreamHI* is one order of magnitude more efficient than *NaiveHI*. The results are very similar with  $M = 1000$  and  $M = 100,000$ . The main reason is *NaiveHI* is an iterative method and it will scan the batch many times to discover *HI*, whereas our algorithm usually scan the data once, also, the dynamic maintaining of the entropies for the candidates, and the pruning of irrelevant candidates can save lots of computation time. On the other hand, Figure 7 (b) shows the compared response times of *NaiveHI* and *StreamHI* with  $k = 6$  when  $nf$ , the number of features in the dataset, varies from 10 to 23. The size of the search space depends on the number of features in the data.

**Effectiveness.** The tables of figure 8 describe the effectiveness difference between *NaiveHI* and *StreamHI*, when  $M$  varies. For each batch, we measure the actual entropy of the featuresets discovered by both algorithms. The column “Same” gives the number of featuresets that are the same for *NaiveHI* and *StreamHI*. The column “Worse” gives the number of batches where the featureset discovered by *StreamHI* has lower entropy than the one of *NaiveHI*. The

nf=23, M=1000				nf=23, M=10,000				nf=23, M=100,000			
k	Same	Worse	Better	k	Same	Worse	Better	k	Same	Worse	Better
3	11865	19	80	3	1167	9	20	3	103	6	10
4	11837	42	85	4	1044	89	63	4	90	5	24
5	11827	30	107	5	1083	87	26	5	94	19	6
6	11709	24	231	6	1154	14	28	6	114	5	0
7	11477	38	449	7	1180	3	13	7	119	0	0
8	11300	63	601	8	1180	3	13	8	119	0	0
9	11321	52	591	9	1186	1	9	9	119	0	0
10	11455	32	477	10	1175	0	21	10	113	0	6
15	11029	18	917	15	1178	1	17	15	118	0	1

Fig. 8. Comparison of effectiveness

column “Better” gives the number of featuresets discovered by *StreamHI* that have higher entropy. Under the same parameters of  $nf$ ,  $k$  and  $M$ , we find that *StreamHI* shares more than 95% results with *NaiveHI*. Moreover, compared with *NaiveHI*, we can find that there are more better results than worse ones in *StreamHI*. Therefore, we can achieve approximately the same or even better results than *NaiveHI*, but with much less time.

## 6 Conclusion

We studied the new problem of mining highly informative featuresets from transaction data streams where the result may evolve over the stream. This problem is important because it allows building classifiers or retrieving information from data streams. We proposed a heuristic algorithm, *StreamHI*, which showed through experiments on real data sets i) effective as experimental results are approximately the same as the existing algorithm on static data, and ii) efficient and suitable for data streams.

## References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD '93: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
2. Giannella, C., Han, J., Pei, J., Yan, X., Yu, P.S.: Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In: Kargupta, H., Joshi, A., Sivakumar, K., Yesha, Y. (eds.) Next Generation Data Mining. AAAI/MIT (2003)
3. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 1–12 (2000)
4. Han, T.S.: Nonnegative entropy measures of multivariate symmetric correlations. *Information and Control* 36(2), 133–156 (1978)

5. Heikinheimo, H., Hinkkanen, E., Mannila, H., Mielikäinen, T., Seppänen, J.K.: Finding low-entropy sets and trees from binary data. In: KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 350–359. ACM, New York (2007)
6. Knobbe, A.J., Ho, E.K.Y.: Maximally informative k-itemsets and their efficient discovery. In: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 237–244. ACM, New York (2006)
7. Liu, H., Motoda, H.: Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series). Chapman & Hall/CRC (2007)
8. Molina, L.C., Belanche, L., Nebot, À.: Feature selection algorithms: A survey and experimental evaluation. In: IEEE ICDM '02, p. 306. IEEE Computer Society, Los Alamitos (2002)
9. Teng, W.-G., Chen, M.-S., Yu, P.S.: A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In: VLDB 2003, pp. 93–104 (2003)
10. Zhang, X., Pan, F., Wang, W., Nobel, A.: Mining non-redundant high order correlations in binary data. In: Proc. VLDB Endow., vol. 1(1), pp. 1178–1188 (2008)

# Continuous Probabilistic Skyline Queries over Uncertain Data Streams

Hui Zhu Su<sup>1</sup>, En Tzu Wang<sup>1</sup>, and Arbee L.P. Chen<sup>2</sup>

<sup>1</sup> Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.  
g9762635@oz.nthu.edu.tw, m9221009@em92.ndhu.edu.tw

<sup>2</sup> Department of Computer Science, National Chengchi University, Taipei, Taiwan, R.O.C.  
alpchen@cs.nccu.edu.tw

**Abstract.** Recently, some approaches of finding probabilistic skylines on uncertain data have been proposed. In these approaches, a data object is composed of instances, each associated with a probability. The probabilistic skyline is then defined as a set of non-dominated objects with probabilities exceeding or equaling a given threshold. In many applications, data are generated as a form of continuous data streams. Accordingly, we make the first attempt to study a problem of continuously returning probabilistic skylines over uncertain data streams in this paper. Moreover, the sliding window model over data streams is considered here. To avoid recomputing the probability of being not dominated for each uncertain object according to the instances contained in the current window, our main idea is to estimate the bounds of these probabilities for early determining which objects can be pruned or returned as results. We first propose a basic algorithm adapted from an existing approach of answering skyline queries on static and certain data, which updates these bounds by repeatedly processing instances of each object. Then, we design a novel data structure to keep dominance relation between some instances for rapidly tightening these bounds, and propose a progressive algorithm based on this new structure. Moreover, these two algorithms are also adapted to solve the problem of continuously maintaining top- $k$  probabilistic skylines. Finally, a set of experiments are performed to evaluate these algorithms, and the experiment results reveal that the progressive algorithm much outperforms the basic one, directly demonstrating the effectiveness of our newly designed structure.

**Keywords:** Uncertain data, Data stream, Continuous query, Probabilistic skyline.

## 1 Introduction

Recently, the skyline query processing has attracted much research attention due to its wide applications such as multi-criteria decision making. Given a set of data objects in a  $d$ -dimensional space, the skyline operator returns the data objects not *dominated* by any other data objects in the set. The returned objects are named *skyline objects*. A data object  $a = (a[1], a[2], \dots, a[d])$  is defined to dominate the other data object  $b = (b[1], b[2], \dots, b[d])$ , if  $a[i]$  is no worse than  $b[i]$ ,  $\forall i$ , and at least in one dimension  $j$ ,  $a[j]$  is



better than  $b[j]$ . In reality, the objects can be uncertain in some applications. Referring to the NBA player dataset as shown in Figure 1, a player regarded as an object may have different performances in different games. The performance record in a certain game of a player can be regarded as an *instance* of an object. Accordingly, in the NBA player dataset, an object is composed of a set of instances. Such objects are called *uncertain objects* in [10]. In [10], Pei et al. first define the dominance relation between uncertain objects and also define the probability of an uncertain object to be a skyline object, called *skyline probability*. Then, the *probabilistic skyline query* is defined in [10] as follows: given a user-defined threshold  $\delta$  with a range of (0, 1], the uncertain objects with skyline probabilities no less than  $\delta$  are returned to users. Two approaches are proposed in [10] to answer the probabilistic skyline query. However, these approaches are only designed for static data rather than data streams.

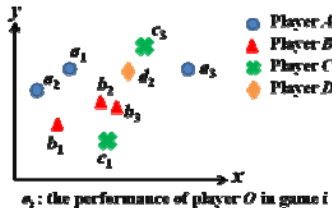


Fig. 1. An NBA player dataset

Intuitively, the instances of uncertain objects may continuously be generated as time goes by, such as the performance records of the NBA players, thus forming the environment of an *uncertain data stream*. Moreover, compared with the newly generated instances, the earlier instances may not be as significant as the new ones and therefore can be ignored. Accordingly, we make the first attempt to consider a new problem on continuously returning the probabilistic skylines over uncertain data streams using a *sliding window model* in this paper, which means that we continuously return the uncertain objects with skyline probabilities no less than a given threshold, regarding the up-to-date instances contained in the current window. In addition, referring to the NBA player dataset, since more than one performance record are generated after playing a game due to multiple players, the *time-based sliding window* is taken into account here. That is, we only concern the instances arriving at the last  $w$  time units, where  $w$  is the size of the sliding window, and moreover, a non-fixed number of instances are generated at a time unit. We define the *survival time* of an instance as in the time interval between its incoming time unit  $i$  to the time unit  $i+w$ , i.e. its expiring time unit.

Whenever the window slides, the incoming and expiring instances may influence the skyline probabilities of the uncertain objects including the objects with no instances contained in the incoming and expiring time units. A straightforward solution to this problem can therefore be designed as follows: whenever the window slides, the skyline probability of each uncertain object is recomputed according to the instances contained in the current window and then checked on whether the corresponding object needs to be returned. Obviously, this solution is very costly. Instead of accurately recomputing

the skyline probabilities for all uncertain objects, the kernel idea of our proposed solutions is to quickly estimate the upper and lower bounds of the skyline probabilities of the uncertain objects to early determine which objects need to be returned as results and which objects can be directly pruned.

In this paper, we first propose a basic algorithm rooted in the SFS approach [3] to repeatedly tighten these bounds by sequentially processing instances of each uncertain object. Notice that, other sort-based approaches such as LESS [15] or SaLSa [16], can also be adapted in place of SFS. The SFS approach proposed in [3] is a fundamental method for finding skylines from static and certain data, in which the data objects are sorted according to a special function for reducing the times of dominance checking. We then propose a progressive algorithm adopting a new data structure extended from the *dominant graph* proposed in [12] for continuously maintaining the dominance relation between some instances. Through the dominance relation maintained, we can obtain more tightened bounds of the skyline probabilities, thus efficiently determining a huge number of uncertain objects to be returned or pruned without needing the computation of the accurate skyline probabilities. The dominant graph proposed in [12] keeps the dominance relation between the data objects in the whole dataset for answering the top- $k$  queries, while our newly designed structure keeps only the dominance relation between a few instances.

Our contributions are summarized as follows. 1) We make the first attempt to address a new problem on continuously returning probabilistic skylines over uncertain data streams; 2) we design a new data structure to continuously maintain the dominance relation between instances and propose a progressive algorithm based on the novel structure, resulting in a performance much better than that of our basic algorithm also proposed in this paper; and 3) we adapt our two algorithms to solve the problem of continuously maintaining top- $k$  probabilistic skylines over uncertain data streams, which continuously maintains  $k$  uncertain objects with the highest skyline probabilities.

The remainder of this paper is organized as follows. The related works are reviewed in Section 2. Section 3 introduces the preliminaries and formalizes the problem on continuously returning probabilistic skylines over uncertain data streams. After that, a basic algorithm and a progressive algorithm for solving this problem are respectively presented in Sections 4 and 5. Moreover, we adapt these two algorithms to solve the problem on continuously maintaining top- $k$  probabilistic skylines in Section 6. The experiment results on evaluating these two algorithms are presented and analyzed in Section 7. Finally, Section 8 concludes this whole work.

## 2 Related Works

After Börzsönyi et al. [2] first propose the skyline operator in the database field, many approaches have been developed for efficiently computing skylines from two points of view: 1) using pre-computed indexes, such as NN [6], ZSearch [9], and OSP [14], and 2) without using indexing, such as D&C [2], BNL [2], SFS [3], and LESS [5]. Then, since many applications may need to face the environment of data streams, the problem of finding skylines over data streams are addressed in [8] and [11], which adopt the sliding window model. To avoid recomputing the whole skyline regarding the data

objects contained in the current window whenever the window slides, a common idea used in [8] and [11] for reducing computation costs is to keep a buffer which only stores the data objects that either are skyline objects or have chances of being skyline objects in the future. More specifically, if a data object is dominated by the other data object arriving later, the object is discarded from the buffer, since it will be always dominated during its survival time, having no chances at all of being a skyline object.

The above approaches focus on certain data. However, data can be uncertain in some applications as mentioned in Section 1. Accordingly, Pei et al. [10] first define the skyline probabilities of uncertain objects, and then develop two approaches to find the probabilistic skylines. Different from Pei et al. [10] assuming that the probability of each instance of an uncertain object is identical, Atallah and Qi [1] assume that the probability of each instance of an uncertain object may not be identical and moreover, the sum of the probabilities of all instances of an uncertain object is at most equal to 1. The existing works focusing on finding probabilistic skylines over uncertain data streams are proposed in Li et al. [7] and Zhang et al. [13]. Far from the problem to be solved in this paper that follows the definition of the uncertain objects proposed in [10], i.e. an uncertain object is composed of a set of instances, [7] and [13] both assume that each data object arrives in the system with a probability, therefore with uncertainty. To the best of our knowledge, we make the first attempt to solve the problem on continuously returning probabilistic skylines over uncertain data streams under the environment that an uncertain data stream is a sequence of instances belonging to some uncertain objects.

### 3 Preliminaries

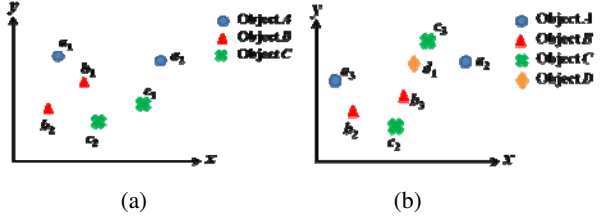
In this section, we first introduce the notations and terms to be used, and then formally define the problem to be solved in this paper. Moreover, the kernel idea of our solutions to the problem is also illustrated here.

#### 3.1 Problem Definition

Following the definition described in [10], an *uncertain object*  $O$  is modeled using a collection of its *instances*, i.e.  $O = \{o_1, o_2, \dots, o_k\}$ , where  $o_i$  is an instance of  $O$ ,  $i = 1$  to  $k$ . The number of instances of  $O$  is denoted as  $|O|$ , i.e.  $|O| = k$ . Similar to [10], uncertain objects are assumed to be independent. That is, the occurrence of an instance of an uncertain object does not affect the occurrences of the instances of the other uncertain objects. Moreover, each instance of an uncertain object is associated with a probability, and the sum of the probabilities of all the instances of an uncertain object is 1. For simplicity, we assume that each of the instances of an uncertain object  $O$  has an identical probability  $1/|O|$ . An *uncertain data stream*  $US$  in this paper is then defined as an unbounded sequence of instances belonging to some uncertain objects. In each time unit  $t_i$ ,  $i = 1, 2, 3, \dots$ , a non-fixed number of instances will arrive in  $US$ . If an instance  $o_i \in O$  arrives at  $t_x$ , then it is denoted as  $o_i^x$ . In this paper, we only concern the data arriving at the most recent  $w$  time units. We therefore use a sliding window  $S_c$  keeping all instances that arrive between  $t_{c-w+1}$  and  $t_c$ , where  $t_c$  is the current time unit.

**Table 1.** An example of an uncertain data stream

time unit	Instances
$t_1$	$a_1, b_1, c_1$
$t_2$	$a_2, c_2$
$t_3$	$b_2$
$t_4$	$d_1, a_3, b_3, c_3$


**Fig. 2.** The instances in  $S_3$  and those in  $S_4$ 

*Example 1:* An uncertain data stream defined in this paper is illustrated in Table 1. As can be seen, a non-fixed number of instances may arrive at each time unit. Suppose that the window size  $w$  is equal to 3. The sliding window of  $S_3$  includes the instances arriving between  $t_1$  and  $t_3$ , as shown in Figure 2a. As the current time unit is  $t_4$  the window slides, leading to the expiry of the instances at  $t_1$ . The current sliding window  $S_4$  therefore includes the instances arriving between  $t_2$  and  $t_4$ , as shown in Figure 2b. Referring to Figure 2a, since  $A$  has two instances including  $a_1$  and  $a_2$ , the probability of being  $a_1$  and that of being  $a_2$  are both equal to  $1/|A|$ , i.e.  $1/2$ . ■

*Definition 1 (Domination):* Let  $o_i(k)$  be the value of the  $k^{\text{th}}$  dimension of an instance  $o_i$ . Given two instances  $o_i \in O$  and  $p_j \in P$  in a  $d$ -dimensional space, where  $O$  and  $P$  are uncertain objects, if  $o_i(k) \leq p_j(k), \forall k = 1, \dots, d$ , and at least in one dimension  $m, o_i(m) < p_j(m)$ , we say that  $o_i$  dominates  $p_j$ , denoted  $o_i \prec p_j$ . ■

*Definition 2 (Skyline Probability of an Instance):* The skyline probability of an instance  $o_i \in O$ , representing the probability that  $o_i$  is not dominated by any other uncertain objects, is defined as

$$Psky(o_i) = \prod_{\forall p \neq o} \left( 1 - \frac{|\{p_j \mid p_j \in P \wedge p_j \prec o_i\}|}{|P|} \right) \quad (1)$$

where  $O$  and  $P$  are uncertain objects. ■

*Definition 3 (Skyline Probability of an Uncertain Object):* The skyline probability of an uncertain object  $O$ , representing the probability that  $O$  is not dominated by any other uncertain objects, is defined as

$$Psky(O) = \frac{1}{|O|} \sum_{\forall o_i \in O} Psky(o_i). \quad (2)$$

*Example 2:* Refer to the sliding window  $S_3$ , shown in Figure 2a. To  $a_1$ , since no other instances of  $B$  or  $C$  dominate  $a_1$  except  $b_2$  of  $B$ ,  $Psky(a_1) = (1 - 1/2) \times (1 - 0/2) = 0.5$ . To  $a_2$ , since all instances of  $B$  and all instances of  $C$  dominate  $a_2$ ,  $Psky(a_2) = (1 - 2/2) \times (1 - 2/2) = 0$ . To  $b_1$ , since no instances of  $A$  or  $C$  dominate  $b_1$ ,  $Psky(b_1) = (1 - 0/2) \times (1 - 0/2) = 1$ . Since  $Psky(a_1) = 0.5$  and  $Psky(a_2) = 0$ , the skyline probability of  $A$ , i.e.  $Psky(A)$ , is equal to  $1/2 \times (Psky(a_1) + Psky(a_2)) = 0.25$ . ■

The problem to be solved in this paper is then defined as follows. Given a threshold  $\delta$  and a sliding window with a size of  $w$ , we issue a *continuous probabilistic skyline query*

that continuously returns the uncertain objects with skyline probabilities no less than  $\delta$ , regarding the most recent  $w$  time units.

*Example 3:* Refer to  $S_3$  and  $S_4$ , shown in Figure 2. Let  $\delta = 0.5$ . Since  $Psky(A) = 0.25$ ,  $Psky(B) = 1$ , and  $Psky(C) = 0.75$  in  $S_3$ , the query results in  $S_3$  are  $B$  and  $C$ . On the other hand, since  $Psky(A) = 0.5$ ,  $Psky(B) = 0.75$ ,  $Psky(C) = 0.5$ , and  $Psky(D) = 0$  in  $S_4$ , the query results in  $S_4$  are  $A$ ,  $B$ , and  $C$ . ■

### 3.2 The Kernel Idea of Our Solutions

A straightforward solution to solve the continuous probabilistic skyline query is as follows: whenever the window slides, the skyline probability of each uncertain object is recomputed and then, the query results can be obtained. To avoid repeatedly computing the accurate skyline probability of each uncertain object, which is very costly, our kernel idea in this paper is to estimate the upper and lower bounds of the skyline probability for each uncertain object. The bounds of the skyline probability of an uncertain object help to quickly determine whether the object should be or should not be returned as one of the results. The *decision rule* is described as follows.

*Decision Rule:* Given a threshold  $\delta$ , 1) if the upper bound of the skyline probability of an uncertain object  $O$ , denoted as  $Psky_{ub}(O)$ , is less than  $\delta$ ,  $O$  can be pruned. 2) If the lower bound of the skyline probability of  $O$ , denoted as  $Psky_{lb}(O)$ , is larger than or equal to  $\delta$ ,  $O$  is sure to be one of the results. 3) If both of the above two statements cannot be satisfied, the refinements of  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  are needed. ■

Eventually, the refinements of  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  lead to the accurate skyline probability of  $O$ , i.e.  $Psky_{lb}(O) = Psky(O) = Psky_{ub}(O)$ .

Similar to Equation 2,  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  are respectively computed using the upper and lower bounds of the skyline probability of each instance of  $O$  as follows:

$$Psky_{ub}(O) = \frac{1}{|O|} \sum_{\forall o_i \in O} Psky_{ub}(o_i) \quad (3), \quad Psky_{lb}(O) = \frac{1}{|O|} \sum_{\forall o_i \in O} Psky_{lb}(o_i) \quad (4)$$

where  $Psky_{lb}(o_i) \leq Psky(o_i) \leq Psky_{ub}(o_i)$ . Accordingly, the following solutions proposed in Sections 4 and 5 focus on the issue of efficiently computing the bounds of the skyline probability of an instance.

## 4 A Basic Algorithm

The Basic algorithm derived from the SFS approach [3] is presented in this section. In SFS, data objects are sorted by a monotonic function as Equation 5, where  $t$  is a data object in a  $d$ -dimensional space and  $t(k)$  is the value of the  $k^{th}$  dimension of  $t$ .

$$E(t) = \sum_{k=1}^d \ln(t(k) + 1) \quad (5)$$

This sorting guarantees that the data objects with higher values cannot dominate those with lower values, thus reducing the times of dominance checking. In the following, we first introduce a data structure used to keep all the instances in the sliding window and then describe the Basic algorithm.

#### 4.1 A Data Structure for Keeping All Instances in the Sliding Window

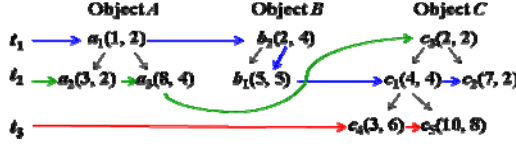


Fig. 3. An example of using heaps to keep all instances in  $S_c$ .

We maintain a *heap* (min-heap) to store all instances of an uncertain object in the current window  $S_c$ , as shown in Figure 3. Accordingly, the number of heaps is equal to the number of uncertain objects contained in the current window. Each instance  $o_i$  has a *key*  $E(o_i)$ , calculated using Equation 5, e.g., the key of  $a_1(1, 2) = 1.79$ . Heaps are constructed according to the keys of instances. Similar to SFS [3], the instances with high key values cannot dominate those with low key values. Moreover, we use a *linked list* to link together the instances arriving at the same time unit for identifying the expiring instances. Whenever the window slides, the instances arriving at the current time unit and those arriving at the expiring time unit need to be respectively inserted in and deleted from the corresponding heaps. These updates follow the standard operations of heaps.

#### 4.2 The Basic Algorithm

---

##### The Basic algorithm

---

- 1: Whenever the window slides
  - 2: Update the corresponding heaps of the instances in the current and expiring time units
  - 3: For each instance  $u_k$  in the current window
  - 4:     Set  $Psky_{ub}(u_k) = 1$  and  $Psky_{lb}(u_k) = 0$
  - 5: For each uncertain object  $O$  in the current window
  - 6:     Count = 0 // counting the number of popped instances of  $O$
  - 7:     While ( $O$  is not yet pruned/returned )
  - 8:          $o_i = \text{pop}(O \text{ 's heap})$
  - 9:         Count = Count + 1
  - 10:          $Psky(o_i) = 1$
  - 11:         For each object  $P$ , where  $P \neq O$
  - 12:             While ( $P$  's heap is not empty and  $p_j$ 's key  $\leq o_i$ 's key)
  - 13:                  $p_j = \text{pop}(P \text{ 's heap})$
  - 14:                 Check whether  $p_j \prec o_i$
  - 15:             
$$Psky(o_i) = Psky(o_i) \times \frac{|\{p_j \mid p_j \in P \wedge p_j \prec o_i\}|}{|P|}$$
  - 16:         
$$Psky_{ub}(O) = \frac{\sum_{\forall o_i \in \text{popped instances of } O} Psky(o_i) + |O| - \text{Count}}{|O|}$$
  - 17:         
$$Psky_{lb}(O) = \frac{\sum_{\forall o_i \in \text{popped instances of } O} Psky(o_i)}{|O|}$$
  - 18:         Check whether  $O$  can be pruned/returned
-

Initially, the upper and lower bounds of the skyline probability of each instance in the current window are respectively set to 1 and 0. For each uncertain object  $O$  in the current window, to check whether it is one of the results, the instances of  $O$  are sequentially popped from its corresponding heap and then compared with the instances of the other objects. While the popped instance of  $O$ , e.g.,  $o_i$ , is compared with the instances of the other object, e.g.,  $P$ , the comparing order also follows the popping sequence of  $P$ . Let the popped instance of  $P$  be  $p_i$ . If the key of  $p_i$  is less than or equal to the key of  $o_i$ , we need to check whether  $p_i$  can dominate  $o_i$ , and also pop the next instance of  $P$ , say  $p_j$  from  $P$ 's heap until the key of  $p_j$  is larger than the key of  $o_i$ . After  $o_i$  is compared with all the other objects, we obtain the accurate skyline probability of  $o_i$ , and then update the upper and lower bounds of the skyline probability of  $O$  to decide whether  $O$  can be pruned/returned using the decision rule. If the status of  $O$  is uncertain, we repeatedly pop the next instance from  $O$ 's heap to refine these bounds until we are sure that  $O$  can be pruned/returned. Notice that, if we traverse all instances of  $O$ , the accurate skyline probability of  $O$  can be obtained.

*Example 4:* Let  $\delta$  be 0.5 and consider the example shown in Figure 3, we discuss whether  $A$  is one of the results in the following. Initially,  $Psky_{ub}(a_i)$  and  $Psky_{lb}(a_i)$  are respectively set to 1 and 0,  $\forall a_i \in A$ . Then, one instance is popped from the heap of  $A$  (i.e.  $a_1$ ) and compared with the instances of  $B$ . Notice that the comparing order follows the popping sequence of  $B$ . Since the key of  $b_2$  (first popped) is larger than the key of  $a_1$ , all instances of  $B$  cannot dominate  $a_1$ . Similarly,  $a_1$  is compared with the instances of  $C$ . Since no instances dominate  $a_1$ ,  $Psky(a_1) = 1$ . Moreover, as  $a_2$  and  $a_3$  are not yet processed in this moment  $Psky_{ub}(a_2) = Psky_{ub}(a_3) = 1$  and  $Psky_{lb}(a_2) = Psky_{lb}(a_3) = 0$ . Hence,  $Psky_{ub}(A) = (1+1+1)/3 = 1$  and  $Psky_{lb}(A) = (1+0+0)/3 = 1/3$ . However, since we are not sure whether  $A$  is one of the results, the refinement of the bounds keeps going. That is, another instance (i.e.  $a_2$ ) is popped for calculating its accurate skyline probability. We obtain that  $Psky(a_2) = 4/5$ . Then,  $Psky_{ub}(A)$  is updated as  $14/15$  and  $Psky_{lb}(A)$  is updated as  $3/5$ . Consequently,  $A$  is returned as one of the results since  $Psky_{lb}(A) = 3/5 \geq 0.5$ .

## 5 A Progressive Algorithm

An instance not dominated by any other instances is defined as a *skyline instance*. For example, given that  $a_1 = (1, 1)$ ,  $a_2 = (1, 2)$ , and  $b_1 = (2, 3)$  in the current window, only  $a_1$  is a skyline instance, since  $a_1$  dominates  $a_2$  and  $b_1$ . Keeping skyline instances with respect to the current window can help to quickly obtain the rough bounds of skyline probabilities of instances/objects. For example, if  $o_i$  is the only skyline instance and  $|O| = 2$  in the current window,  $Psky(p_j)$  is at most equal to 0.5, where  $p_j$  is an instance of the uncertain object  $P$ ,  $P \neq O$ , and moreover,  $Psky(O)$  is at least equal to 0.5. In this section, we propose a progressive algorithm that keeps the skyline instances with respect to the current window for efficiently computing the bounds of the skyline probabilities of instances/objects.

### 5.1 Time Dominant Graph

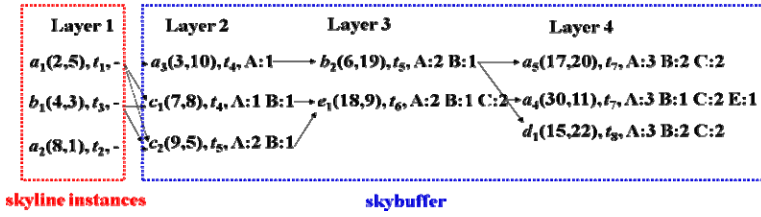


Fig. 4. An example of the time dominant graph in  $S_c$

The skyline instances in the current window  $S_c$  may change as time goes by. Some skyline instances expire, making the non-skyline instances dominated by the expiring skyline instances to have chances of becoming skyline instances. Accordingly, to efficiently maintain the skyline instances with respect to  $S_c$ , we need to further take into account the non-skyline instances with respect to  $S_c$ , which have chances of becoming skyline instances in the future. An intuitive observation is as follows: to an instance  $o_i$  arriving at the time unit  $t_x$ , i.e.  $o_i^x$ , if an instance  $p_j^y$  exists such that  $p_j^y < o_i^x$  and  $y \geq x$ ,  $o_i^x$  has no chances at all of becoming a skyline instance during its survival time. We therefore keep all instances in  $S_c$  except such instances as  $o_i^x$  mentioned above using an efficient data structure named *time dominant graph (TDG)* in which the dominance relation between the kept instances can be indicated.

The time dominant graph used in this paper is adapted from *dominant graph (DG)* proposed in [12], composed of (*instance nodes*). The original DG in [12] is designed for keeping the whole dataset, different from our use of TDG. Each node in TDG keeps 1) the identifier/values of an instance, 2) the arriving time of the corresponding instance, and 3) the number of instances that dominate the corresponding instance, i.e. the number of *dominating instances*, with respect to each uncertain object. Moreover, each node belongs to a *layer*, and the skyline instances are contained in the nodes at Layer 1. For each instance  $o_i$  contained in a node at Layer  $K$ , there must be at least one instances dominating  $o_i$ , contained in the nodes at layer  $(K - 1)$ , and with arriving time earlier than that of  $o_i$ . Moreover, the instances in the identical layer cannot dominate each other. The instances contained in the nodes not at Layer 1 can be regarded as being kept in a *buffer* for continuously maintaining the skyline instances. We call this conceptual buffer as *skybuffer*. An example of the time dominant graph is shown in Figure 4. Notice that, the node containing an instance  $o_i$  is directly identified by  $o_i$  in the following discussion.

### 5.2 Maintenance of TDG

*Observation 1:* Let the current time unit be  $t_c$ . To a newly arrived instance  $o_i^c$ , if we have the other newly arrived instance  $p_j^c$  such that  $p_j^c < o_i^c$ ,  $o_i^c$  need not be inserted into TDG. ■

Obviously,  $o_i^c$  has no chances at all of becoming a skyline instance, since it is dominated by  $p_j^c$  during its survival time, thus it need not be inserted into TDG.



*Observation 2: Whenever an instance  $o_i$  is inserted into TDG, the other instances kept in TDG, which can be dominated by  $o_i$ , are deleted from TDG.* ■

The arriving time of all instances in TDG is earlier than or equal to that of  $o_i$ , since  $o_i$  is newly inserted. Accordingly, the instances kept in TDG, which can be dominated by  $o_i$ , have no chances at all of becoming skyline instances during their survival time due to  $o_i$ , thus being deleted from TDG.

The insertion procedure and deletion procedure of TDG are described as follows.

**Insertion.** While an instance  $o_i$  is inserted to TDG, we need to find out the lowest layer, e.g., Layer  $K$ , at which no instances dominate  $o_i$  by comparing  $o_i$  to each instance at each layer, starting from Layer 1. Then  $o_i$  is inserted at Layer  $K$  and moreover, all the instances at Layer  $K$  to the last layer, which can be dominated by  $o_i$ , are deleted from TDG. Notice that, while finding out the corresponding lowest layer for a newly arrived instance, some dominance checking is performed. The number of dominating instances of  $o_i$ , with respect to each uncertain object can therefore be obtained.

**Deletion.** While an instance  $o_i$  at Layer  $K$  is deleted, we need to find out the instances at Layer  $(K + 1)$ , which are only dominated by  $o_i$ , and then, make these instances to go forward to Layer  $K$ . Moreover, we need to construct the dominance relation between the dominating instances of  $o_i$  at Layer  $(K - 1)$  and the newly moved instances at Layer  $K$ . Obviously, moving the instances originally at Layer  $(K + 1)$  forward to Layer  $K$  can be regarded as deleting these instances from Layer  $(K + 1)$ . The deletion operation is thus recursively performed until the whole TDG is completely updated.

Whenever the window slides, TDG in  $S_c$  is updated as follows. First, the newly arrived instances that are not dominated by any other newly arrived instances are inserted into TDG by performing the insertion algorithm. Then, the expiring skyline instances are deleted from TDG by performing the deletion algorithm.

The time dominant graph has some interesting properties as follows.

*Property 1: Given a TDG in  $S_c$ , once an instance is deleted from TDG, it cannot be inserted into TDG again during its survival time.* ■

If an instance  $o_i^x$  is deleted from TDG, either  $o_i^x$  expires or the other instance  $p_j^y$  dominating  $o_i^x$  is inserted into TDG, where  $x < y$ . Accordingly,  $o_i^x$  cannot become a skyline instance during its survival time due to  $p_j^y$ , thus not being inserted again.

*Property 2: Given a TDG in  $S_c$ , to any two instances  $o_i^x$  at Layer  $K$  and  $p_j^y$  at Layer  $L$ ,  $K < L$  and  $x < y$ , if  $o_i^x \prec p_j^y$ .* ■

*Property 3: Given a TDG in  $S_c$ , for any instance  $o_i$  not kept in TDG and not expiring, we can find out at least one instance  $p_j$  kept in TDG such that  $p_j \prec o_i$ .* ■

If an instance  $o_i$  is not kept in TDG, either  $o_i$  was not inserted into TDG or  $o_i$  had been deleted from TDG. If the first condition holds, at least one instances dominating  $o_i$  and arriving at the same time with  $o_i$  must exist. If the second condition holds, at least one instances dominating  $o_i$  and arriving later than  $o_i$  must exist.

### 5.3 The TDG Algorithm

The TDG algorithm using the time dominant graph for efficiently computing the bounds of the skyline probabilities of instances/objects is presented in this subsection. Similar to the Basic algorithm, all the instances of each uncertain object in the current window are kept using a heap in the TDG algorithm. Moreover, we additionally maintain a TDG for the current window in this algorithm. The main concept of the algorithm is described as follows. When the window slides, a linked list is constructed to link together all instances just arriving at the current time unit. Then, TDG in  $S_c$  is updated according to the expiring and incoming instances. Moreover, the corresponding heaps are also updated using the standard heap operations. After that, the upper and lower bounds of the skyline probability of each uncertain object are calculated.

Step 1: For each uncertain object  $O$ , if its instance  $o_i, \forall i$ , is not kept in TDG,  $Psky_{ub}(o_i)$  and  $Psky_{lb}(o_i)$  are respectively set to 1 and 0. Otherwise, we can use the information contained in the corresponding node of  $o_i$  in TDG to calculate  $Psky_{ub}(o_i)$  and  $Psky_{lb}(o_i)$ . For example, to an instance  $a_1$ , if 3 instances of  $B$  ( $|B| = 5$ ) and 2 instances of  $C$  ( $|C| = 4$ ) dominate  $a_1$  from the corresponding node of  $a_1$  kept in TDG,  $Psky_{ub}(a_1) = (1 - 3/5)(1 - 2/4) = 0.2$ . Moreover, let the number of instances of  $B$ , not kept in TDG, and the number of instances of  $C$ , not kept in TDG, be respectively equal to 1 and 2. Then,  $Psky_{lb}(a_1) = (1 - (3 + 1)/5)(1 - (2 + 2)/4) = 0$ . While  $o_i$  is kept in TDG,  $Psky_{ub}(o_i)$  is computed by assuming that  $o_i$  is only dominated by the instances being recognized as dominating  $o_i$  in TDG. On the other hand,  $Psky_{lb}(o_i)$  is computed by assuming that in addition to the instances being recognized as dominating  $o_i$  in TDG, all the instances not kept in TDG also dominate  $o_i$ . After obtaining  $Psky_{ub}(o_i)$  and  $Psky_{lb}(o_i)$  for each instance  $o_i \in O$ ,  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  can be respectively calculated using Equations 3 and 4 to decide whether  $O$  is pruned/returned.

Step 2: For each uncertain object  $O$  not pruned/returned after executing Step 1,  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  need to be further refined by considering the instances of  $O$ , not kept in TDG, since the skyline probabilities of these instances have very rough bounds, either 1 or 0 (for the upper or lower bounds). For each instance  $o_i \in O$ , not kept in TDG, we compare  $o_i$  with all the instances kept in TDG, starting from Layer 1 and layer by layer, to find out a suitable layer for  $o_i$ , i.e. the lowest layer at which no instances dominate  $o_i$ . This procedure is very similar to the insertion procedure of TDG except really inserting  $o_i$  in TDG. Accordingly, similar to computing the bounds for the instances kept in TDG, we can obtain the tighter  $Psky_{ub}(o_i)$  and  $Psky_{lb}(o_i)$  instead of 1 and 0, and then obtain the tighter  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$ .

Step 3: For each uncertain object  $O$  not pruned/returned after executing Step 2, we have to calculate the accurate skyline probability for each instance of  $O$  as done in the Basic algorithm.

**The TDG algorithm**

- 
- 1: Whenever the window slides
  - 2: Update TDG and the corresponding heaps of the instances in the current and expiring time units
  - 3: For each instance  $u_k$  in the current window
  - 4: Set  $Psky_{ub}(u_k) = 1$  and  $Psky_{lb}(u_k) = 0$
  - 5: For each uncertain object  $O$  in the current window
  - 6: **Step 1:** For each instance  $o_i$  kept in TDG
  - 7: 
$$Psky_{ub}(o_i) = \prod_{\forall p \neq o} 1 - \frac{|\{p_j \mid p_j \in P \wedge p_j \prec o_i\}|}{|P|}$$
  - 8: 
$$Psky_{lb}(o_i) = \prod_{\forall p \neq o} 1 - \frac{|\{p_j \mid p_j \in P \wedge p_j \prec o_i\}| + |P| - |\{p_k \mid p_k \in P \cup TDG\}|}{|P|}$$
  - 9: Compute  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  using Equations 3 and 4
  - 10: If ( $O$  is not pruned/returned)
  - 11: **Step 2:** While ( $O$  is not yet pruned/returned and  $O$ 's heap is not empty )
  - 12:  $o_i = \text{pop}(O \text{'s heap})$
  - 13: If( $o_i$  is not kept in TDG)
  - 14: Count the instances dominating  $o_i$  in TDG using an adaption of the insertion procedure of TDG
  - 15: 
$$Psky_{ub}(o_i) = \prod_{\forall p \neq o} 1 - \frac{|\{p_j \mid p_j \in P \wedge p_j \prec o_i\}|}{|P|}$$
  - 16: 
$$Psky_{lb}(o_i) = \prod_{\forall p \neq o} 1 - \frac{|\{p_j \mid p_j \in P \wedge p_j \prec o_i\}| + |P| - |\{p_k \mid p_k \in P \cup TDG\}|}{|P|}$$
  - 17: Compute  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  using Equations 3 and 4
  - 18: **Step 3:** If ( $O$  is not pruned/returned)
  - 19: Apply the Basic algorithm to  $O$
- 

## 6 An Extension

The proposed algorithms can be adapted to find the top- $k$  probabilistic skylines over uncertain data streams. That is, given a number of  $k$  and a sliding window with a size of  $w$ , we issue a continuous top- $k$  probabilistic skyline query that continuously returns  $k$  uncertain objects having the highest skyline probabilities, regarding the most recent  $w$  time units. To adapt the Basic and the TDG algorithms to deal with this problem, we only need to modify the decision rule mentioned in Subsection 3.2 as follows.

*Decision Rule:* Given a number of  $k$ , find out the  $k^{\text{th}}$  highest upper bound and the  $k^{\text{th}}$  highest lower bound among the upper and lower bounds of the skyline probabilities of all uncertain objects and name these two values as  $Psky_{ub}^k$  and  $Psky_{lb}^k$ , respectively. 1) If  $Psky_{ub}(O)$  is smaller than  $Psky_{lb}^k$ ,  $O$  can be pruned. 2) If  $Psky_{lb}(O)$  is larger than or equal to  $Psky_{ub}^k$ ,  $O$  is sure to be one of the results. 3) If the above two statements cannot be satisfied, the refinements of  $Psky_{ub}(O)$  and  $Psky_{lb}(O)$  are needed. Moreover,  $k$  is continuously updated by subtracting the number of obtained results from the original  $k$ .

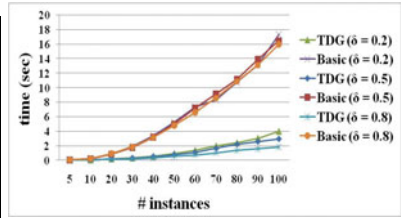
## 7 Performance Evaluation

To the best of our knowledge, there are no existing approaches. Therefore, solving the same problem defined in this paper, we perform a set of experiments to evaluate only

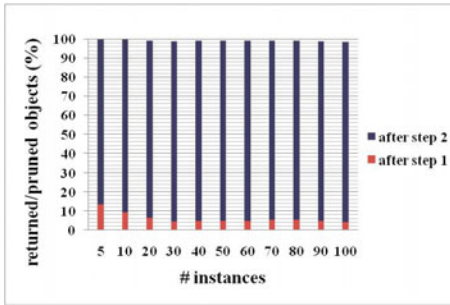
the two proposed algorithms in this section. We use a NBA dataset downloaded from *databaseSports.com* [4] to be the test dataset. The number of players regarded as uncertain objects in the test dataset is 3,573, including 19,112 records regarded as instances. Moreover, both of these two algorithms are implemented in JAVA, and all of the experiments are performed on a PC with the Intel Core 2 Quad 2.4GHz CPU, 2GB of main memory, and under the Windows XP operating system. The parameters used in the experiments are summarized in Table 2.

**Table2.** The parameters used in the experiments

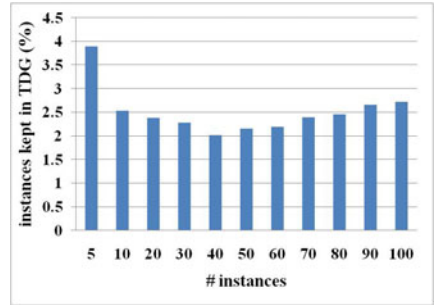
Factor	Default	Range	w: the number of time units in the sliding window # instances: the number of instances arriving at a time unit # dimensions: the number of dimensions of an instance $\delta$ : a given threshold
$w$	10	10 ~ 100	
# instances	10	5 ~ 100	
# dimensions	5	2 ~ 10	
$\delta$	0.2\0.5\0.8	-	



**Fig. 5.** The processing time w.r.t. # instances



**Fig. 6.** The confirmed objects using TDG w.r.t. # instances,  $\delta = 0.5$ .



**Fig. 7.** The instances kept in TDG w.r.t. # instances

First, we focus on the processing time of the algorithms on the varying number of instances at a time unit, shown in Figure 5. To obtain the processing time in the experiments, we let the window slide 100 times and then, compute the average processing time. Obviously, the larger the number of instances in a time unit is, the longer the processing time of the two algorithms will be. From Figure 5, we also find that the TDG algorithm much outperforms the Basic algorithm in terms of the processing time under being given different thresholds as 0.2, 0.5, and 0.8. Figure 6 shows the effectiveness of the time dominant graph used in the TDG algorithm. After executing Step 1 in the TDG algorithm, only a few uncertain objects are confirmed (either pruned or returned), since most of the instances are not kept in the time dominant graph. Although Step 1 only confirms a small number of uncertain objects,

TDG is useful in that over 90% of uncertain objects can be confirmed after executing Step 2. That is, we only need to perform the Basic algorithm on a very few uncertain objects. Moreover, as shown in Figure 7, only a small number of instances, i.e. around 2% of the instances in the current window are kept in TDG, which is very space-efficient.

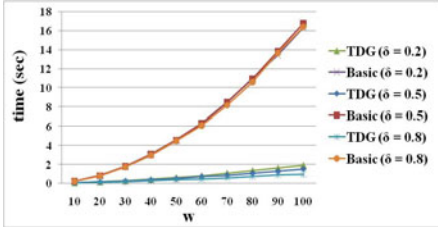


Fig. 8. The processing time w.r.t.  $w$ , using a fixed number of instances

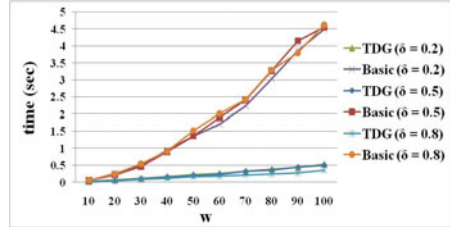


Fig. 9. The processing time w.r.t.  $w$ , using a non-fixed number of instances

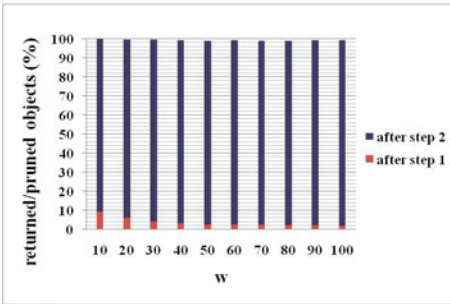


Fig. 10. The confirmed objects using TDG w.r.t.  $w$ , using a fixed number of instances,  $\delta = 0.5$

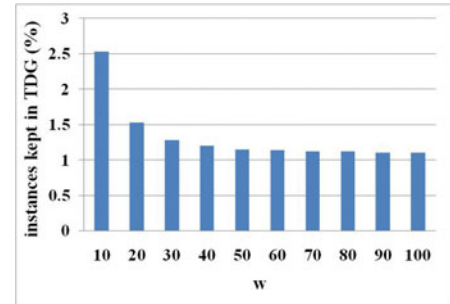


Fig. 11. The instances kept in TDG w.r.t.  $w$ , using a fixed number of instances

Second, we focus on the processing time of the algorithms on a varying  $w$ , which is shown in Figures 8-9. In Figure 8, the number of instances arriving at a time unit is fixed and set to the default value, i.e. 10. On the other hand, to simulate the condition that a non-fixed number of instances may arrive at a time unit, the instances arriving at a time unit are randomly generated with a number from 0 to 10 in Figure 9. Either in Figure 8 or in Figure 9, the larger the window size  $w$  is, the longer the processing time of the two algorithms will be. As  $w$  increases, the ratio of expiring/incoming time unit to  $w$  decreases. Since the instances of the expiring/incoming time unit may have chances of influencing the skyline probabilities of the instances in the other time units, the processing time of the TDG algorithm slowly increases. On the other hand, the processing time of the Basic algorithm increases quickly with the rising  $w$ . Figures 10 and 11 respectively show the effectiveness and space-efficiency of the time dominant graph used in the TDG algorithm on a varying  $w$ .

In the following, we focus on the processing time of the algorithms on a varying number of dimensions, which is shown in Figures 12-13. Similarly, the number of instances arriving at a time unit is fixed with respect to Figure 12, and the instances

arriving at a time unit are randomly generated with a number from 0 to 10 with respect to Figure 13. Intuitively, the processing time of the algorithms increases with the increasing of the number of dimensions, even that, the TDG algorithm still much outperforms the Basic algorithm. Figures 14 and 15 respectively show the effectiveness and space-efficiency of the time dominant graph used in the TDG algorithm on a varying number of dimensions.

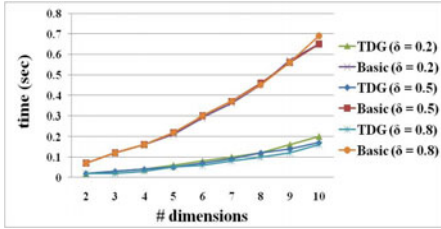


Fig. 12. The processing time w.r.t. # dimensions, using a fixed number of instances

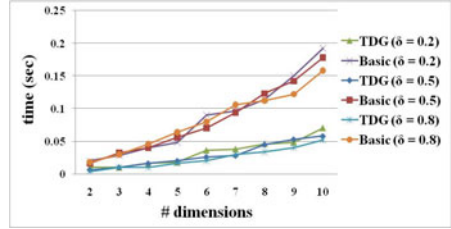


Fig. 13. The processing time w.r.t. # dimensions, using a non-fixed number of instances

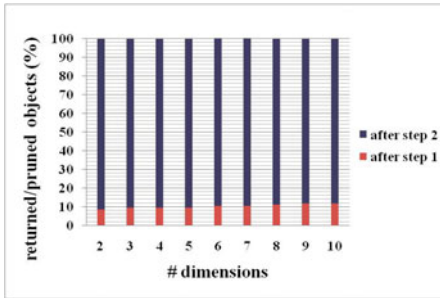


Fig. 14. The confirmed objects using TDG w.r.t. # dimensions, using a fixed number of instances,  $\delta=0.5$

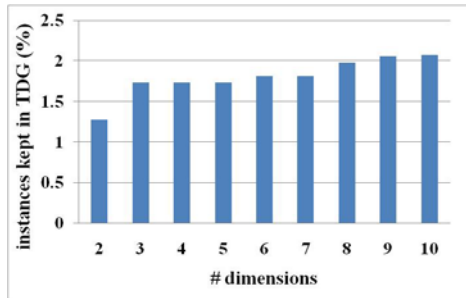


Fig. 15. The instances kept in TDG w.r.t. # dimensions, using a fixed number of instances

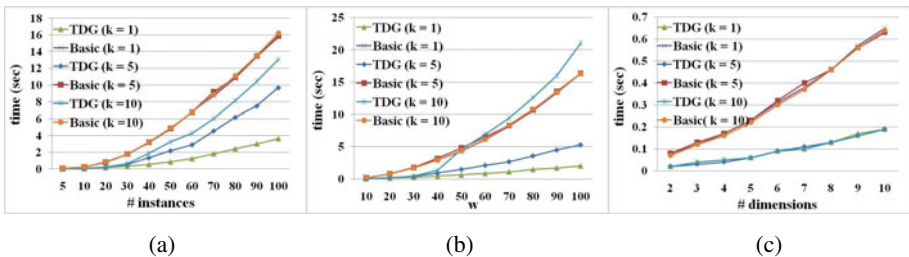


Fig. 16. The processing time on continuous top- $k$  probabilistic skyline query w.r.t. # instances,  $w$ , # dimensions

Finally, we also implement the adaption of the two algorithms for solving the continuous top- $k$  probabilistic skyline query. In the experiments,  $k$  is respectively set to

1, 5, and 10. As shown in Figure 16, the TDG algorithm outperforms the Basic algorithm in most of the cases. It is worth mentioning that as  $k$  becomes large, e.g., 10, the slope of the line indicating the processing time of the TDG algorithm in Figures 16(a) and 16(b) becomes large too. This is because when  $k$  is large,  $Psky_{ub}^k$  and  $Psky_{lb}^k$  become small, thus reducing the decision power of the time dominant graph used in the TDG algorithm. Moreover, the decision power of TDG may also be a bit low under the condition that the accurate skyline probability of each uncertain object is almost identical. Overall, the TDG algorithm is more efficient than the Basic algorithm in the experiments. Moreover, the time dominant graph used in the TDG algorithm is very space-efficient and effective. It is worth exchanging a little maintenance cost of TDG for reducing a lot of processing time in dealing with the problem on continuously returning probabilistic skylines.

## 8 Conclusion

In this paper, we make the first attempt to solve the problem on continuously returning probabilistic skylines over uncertain data streams. Given a user-defined threshold, the uncertain objects with skyline probabilities no less than the threshold are continuously returned, regarding the current sliding window. Without directly calculating the accurate skyline probability for each uncertain object whenever the window slides, which is very costly, the bounds of skyline probabilities of instances/objects are estimated in our proposed two algorithms to quickly identify which uncertain objects need to be pruned/returned. For efficiently computing the upper and lower bounds of skyline probabilities, we design a novel data structure named TDG extended from a published data structure named DG to continuously maintain the skyline instances with respect to the current sliding window. By the information kept in TDG and comparing the instances not kept to those kept in TDG, the useful bounds of skyline probabilities can be estimated, thus substantially reducing the processing time as shown in the experiments. Moreover, we also adapt our approaches to deal with the problem on continuously maintaining the top- $k$  probabilistic skylines over uncertain data streams. Similarly, the TDG algorithm outperforms the Basic algorithm in most of the cases.

## References

- [1] Atallah, M.J., Qi, Y.: Computing all skyline probabilities for uncertain data. In: PODS 2009, pp. 279–287 (2009)
- [2] Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE 2001, pp. 421–430 (2001)
- [3] Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE 2003, pp. 717–816 (2003)
- [4] <http://www.databaseSports.com/>
- [5] Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: VLDB 2005, pp. 229–240 (2005)
- [6] Kossmann, D., Ramsak, F., Rost, S.: Shooting starts in the sky: An online algorithm for skyline queries. In: VLDB 2002, pp. 275–286 (2002)

- [7] Li, J.J., Sun, S.L., Zhu, Y.Y.: Efficient maintaining of skyline over probabilistic data stream. In: ICNC 2008, pp. 378–382 (2008)
- [8] Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: ICDE 2005, pp. 502–513 (2005)
- [9] Lee, K.C.K., Zheng, B., Li, H., Lee, W.C.: Approaching the skyline in Z order. In: VLDB 2007, pp. 279–290 (2007)
- [10] Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: VLDB 2007, pp. 15–26 (2007)
- [11] Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. IEEE TKDE 18(2), 377–391
- [12] Zou, L., Chen, L.: Dominant Graph: An efficient indexing structure to answer top-k queries. In: ICDE 2008, pp. 536–545 (2008)
- [13] Zhang, W., Lin, X., Zhang, Y., Wang, W., Yu, J.X.: Probabilistic skyline operator over sliding windows. In: ICDE 2009, pp. 1060–1071 (2009)
- [14] Zhang, S., Mamoulis, N., Cheung, D.W.: Scalable skyline computation using object-based space partitioning. In: SIGMOD 2009, pp. 483–494 (2009)
- [15] Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: VLDB 2005, pp. 229–240 (2005)
- [16] Bartolini, I., Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. ACM TODS 33(4), 31–49



# DBOD-DS: Distance Based Outlier Detection for Data Streams

Md. Shiblee Sadik and Le Gruenwald

School of Computer Science, University of Oklahoma  
110 W. Boyd St., Norman, OK 73019, USA  
{shiblee.sadik, ggruenwald}@ou.edu

**Abstract.** Data stream is a newly emerging data model for applications like environment monitoring, Web click stream, network traffic monitoring, etc. It consists of an infinite sequence of data points accompanied with timestamp coming from external data source. Typically data sources are located onsite and very vulnerable to external attacks and natural calamities, thus outliers are very common in the datasets. Existing techniques for outlier detection are inadequate for data streams because of its metamorphic data distribution and uncertainty. In this paper we propose an outlier detection technique, called Distance-Based Outline Detection for Data Streams (DBOD-DS) based on a novel continuously adaptive probability density function that addresses all the new issues of data streams. Extensive experiments on a real dataset for meteorology applications show the supremacy of DBOD-DS over existing techniques in terms of accuracy.

**Keywords:** Data stream, outlier detection, probability density function.

## 1 Introduction

Applications like environment monitoring, Web click stream, and network traffic monitoring use a new data model to represent their never ending series of data called data streams. Data stream has received a great deal of attention in the research community in recent years due to its novel characteristics. On the other hand every real life dataset has outliers in it [6]; therefore outlier detection is a very important part of data acquisition. In most of the cases the work done on outlier detection for data streams [1], [3], [8] is adopted from outlier detection techniques for regular data with ad-hoc modifications and do not address all the novel characteristics of data streams. In this paper we propose a novel outlier detection technique to fill the gap. Before going further we briefly discuss the novel characteristics of data streams and data stream processing requirements.

Applications for data streams are significantly different from those for regular data in many facets. In data stream applications, data have the essence of time, are mostly append only and, in many cases, are transient [2], [5]; therefore offline store and process approaches are not very suitable for online data stream; consequently data processing has to be online and incremental [25]. Data are continuously coming in a

streaming environment in a very fast rate with changing data distribution [17], and thus any fixed data distribution is not adequate enough to capture the knowledge. On top of this, in many cases uncertainty in data streams makes processing more complicated. The novel characteristics of data streams bring the outlier detection problem out on the open again. The next paragraph introduces the problem of outlier detection in a real life dataset.

An outlier refers to a data point which does not conform well to the pattern of the other data points or normal behaviors or conform well to the outlying behavior [4], [6]. Pragmatically, normal behaviors are easy to identify and every possible outlying behavior are difficult to compile; nonetheless the outlying behaviors are changing over time. Almost all real datasets have outliers [6]. The major reasons behind the outliers are malicious activity or intrusion, instrumental error or setup error, change in environment, human error, etc. Evidently, outlier detection is not a new topic at all. It has been in the literature since the eighteenth century [4]. Even though the problem has been in the literature for so many years it is still very popular; this is because nobody knows the real outliers and the detection of outliers is very subjective to the application. The outlier detection with perfect confidence in regular data is still not an easy problem. This is because of the inherent vagueness in the definition of outlier, like how to define regular behavior, to what extend an outlier needs to be not conforming to the regular behavior, etc. The problem of outlier detection becomes more complicated when considering new characteristics of data streams, such as unbounded data, varying data distribution, data uncertainty, and temporal dimension. None of the existing outlier detection techniques addresses all of these characteristics. In this paper, we present a novel outlier detection technique for data streams based on the concept of probability density function, called Distance-Based Outlier Detection for Data Streams (DBOD-DS), that addresses all the characteristics of data streams. We then present the results of the experiments that we have conducted on a real dataset obtained from a meteorological data stream application [7] to compare the accuracy and execution time of DBOD-DS with the two outlier detection techniques existing in the literature: ART [8] and ODTS [3].

The rest of the paper is organized as follows: Section 2 discusses the work related to outlier detection in data stream; Section 3 describes our approach and its implementation; Section 4 presents the experimental results we have obtained, and finally Section 5 provides our conclusions and future research.

## 2 Related Work

Most of the outlier detection techniques for data streams use a sliding window to capture the recent data values and detect the outliers inside the window [1], [3], [26] with multi-pass algorithms. Data streams change over time and an outlier for a particular window may appear as an inlier in another window; hence the notion of outlier in a data stream window is not very concrete. Nevertheless, an inlier can be shown as an outlier by changing the window size [3]; thus the outlier detection techniques that use a sliding window work well if the window size is chosen carefully. However, different techniques interpret window size differently; in most

situations, it is difficult for the domain expert to choose the window size correctly without knowing the interpretation of a particular technique.

Auto-regression based techniques for outlier detection are very popular for time series outlier detection [4]. Some outlier detection techniques for data streams adopt auto-regression [8], [22]. Most of the auto-regression based techniques work similarly in which a data point is compared with an estimated model and a metric is computed based on the comparison. If the metric is beyond a certain limit (called cut-off limit), the data point is identified as an outlier. The advantages of auto-regression based models are that they are computationally inexpensive and they provide an estimated value for the outlier. However, the success of this method depends on the quality of the auto-regression model and the efficacy of the cut-off limit. Different data streams show different natures in their changing patterns; therefore it is very difficult to select an appropriate auto-regression model for data streams [8]. The selection of a magic cut-off point not only depends upon the data but also the auto-regression model chosen.

Outlier detection techniques for multiple data streams have been proposed in the literature [16], [10], [11], [26]. The underlying assumptions are the availability of multiple homogeneous data streams and their synchronous behavior. These may not be the case as multiple homogeneous data streams may not be available or one data stream may behave very differently from the others. In the later case comparing two heterogeneous data streams does not help to point out the outliers.

Statistical [4] and machine learning [9] based techniques assume a fixed distribution for the data and if the probability of a data point is very low it is identified as an outlier by statistical and machine learning based techniques. Data streams are highly dynamic in nature and their distribution changes over time. No fixed data distribution is good enough for the entire data stream; hence summarizing a dynamic data stream with a static data distribution produces questionable results.

Data clustering algorithms produce outliers as a bi-product [21], [24]; but as outlier detection is not the focus of clustering algorithms, they are not optimized for outlier detection. Keogh *et al* argued that most of the clustering algorithms for time series/data stream produce meaningless results [18]; hence their efficacy and correctness are still in question.

However none of the existing outlier detection technique considers the uncertainty, concept drift and the transient property of the data stream. Moreover, not all the outlier detection algorithms are truly incremental rather they store a subset of the data points and use multi-pass algorithms to detect the outliers in the subset. While designing a technique of outlier detection for data streams, one needs to consider the uncertainty, the drift of concepts, the transient property, the temporal characteristic of the data points, etc. On top of this, every computation has to be online and incremental. To fill the gap, we have designed our technique addressing the fact that data points in a data stream are very uncertain. We also address temporal characteristics of the data points. Moreover we do not assume any type of fixed data distribution to address the fact that the concept drift occurs in data stream. Next section (3) portrays the details of our algorithm with the implementation issues.

### 3 Proposed Technique: Distance-Based Outlier Detection for Data Streams (DBOD-DS)

In this section, we first provide an overall description of our proposed technique, DBOD-DS. We then discuss our novel probability density function, which is the basis of our technique, and algorithms to implement it.

#### 3.1 Overall Approach

Our approach is motivated by distance-based outlier [26], [19] and based on a probability density function  $f(x)$  which resembles data distribution where  $x$  is a random variable. As each data point  $d$  with the value  $v$  comes in we compute the probability of occurrence of the values  $p(v, r)$  within user defined radius  $r$  from the data value  $v$  ( $p(v, r)$ ) by integrating the probability density function  $f(x)$  from  $v - r$  to  $v + r$ ,  $p(v, r) = \int_{v-r}^{v+r} f(x) dx$ . The probability of occurrence resembles the neighbor density around the data value [19]; if the neighbor density is very low the data point is more likely to be an outlier. According to our approach, if the probability of occurrence  $p(v, r)$  is less than the user defined minimum probability of occurrence ( $q$ ) i.e.,  $p(v, r) < q$  the data point  $d$  is identified as an outlier.

As we receive each data point  $d$ , we update the probability density function  $f(x)$  by increasing the probability of occurrence of data value  $v$ . To address the data uncertainty characteristic of data stream, when we receive the data point  $d$  we not only increase the probability of the data value ( $v$ ) by  $p_v$  but also increase the probability of other values by a fraction of  $(1 - p_v)$  where  $p_v$  is the probability of occurring the data value  $v$  while there is a data uncertainty.

To address the temporal characteristic of the data streams, when we compute the probability density function  $f(x)$  the data points ( $d_1, d_2, \dots, d_n$ ) are weighted based on their freshness. The most recent one receives the highest weight and the oldest one receives the lowest weight. If the respective values are ( $v_1, v_2, \dots, v_n$ ) where the  $v_n$  is the most recent one and  $v_1$  is the oldest one, we weight them by  $(\lambda^{n-1}, \lambda^{n-2}, \dots, 1)$ , respectively; therefore for the value ( $v_i$ ) we update the probability density function  $f(x)$  by increasing the probability of occurrence of  $v_i$  by  $p_{v_i} \lambda^{i-1}$  and the probability of others values by a fraction of  $(1 - p_{v_i}) \lambda^{i-1}$ .

To address the varying data distribution characteristic of data streams, our probability density function  $f(x)$  does not assume any particular fixed data distribution; rather we adjust our probability density function on-the-fly; therefore our probability density function ( $f(x)$ ) never becomes obsolete due to a change in data distribution (concept drift [17]), rather our probability density function ( $f(x)$ ) always provide the most recent data distribution.

Now at any particular time if we integrate our probability density function from  $v - r$  to  $v + r$  we obtain the probability of occurrence  $p(v, r)$  of a data value  $v$  within  $v - r$  to  $v + r$ . If  $p(v, r)$  is large, then the data value  $v$  has a very high probability of occurrence or neighbor density in recent time. Therefore our approach requires two user defined parameters, radius  $r$  and minimum probability of occurrence  $q$ . However if the probability function density function  $f(x)$  is continuous, the same result can be produced by a different set of  $(r, q)$ , thus by fixing

the value of  $q$  and changing the value of  $r$  we can obtain the optimal result, which reduces the curse of having two parameters to one. We fix the value of  $q$  and change the value of  $r$  to receive the optimal performance. The next section (3.2) presents the detail of our proposed probability density function.

### 3.2 Proposed Probability Density Function

Our proposed probability density function is based on a kernel probability density estimator. Several techniques exist in literature to estimate probability density function like histogram [15], wavelet [14], kernel estimation [26], etc. Among those techniques we choose kernel probability density estimator (in short kernel estimator) for our approach. We will justify our choice in the next paragraph.

The kernel estimator estimates the probability density function based on the data values. For each data value  $v$  the kernel estimator increases the probability of occurrence of  $v$  by  $p_v$  and increases the probability of occurrence of other values by a fraction of  $1 - p_v$  which fits our requirements excellently. Due to data uncertainty when we receive a data point  $d$  with value  $v$ , we cannot assert the data value with full confidence; therefore we cannot increase the probability of occurrence of  $v$  by 1. Since the value  $v$  is uncertain, it might be induced by other data values other than  $v$ . Thus to address the uncertainty of data streams, we do not increase the probability of occurrence of  $v$  by 1. Kernel estimator increases the probability of occurrence of  $v$  by  $p_v$  and distributes the rest of the probability of occurrence ( $1 - p_v$ ) into the other data values which are close to the value  $v$ . Formally, if  $(x_1, x_2, \dots, x_n)$  are  $n$  sample data points, their respective values are  $(v_1, v_2, \dots, v_n)$  and the probability density function  $f(x)$  is defined by equation (1) where  $k(x)$  is called the *kernel function*.  $v_i$  can be a scalar or vector.

$$f(x) = \frac{1}{n} \sum_{i=1}^n k(v_i - x) \quad (1)$$

The kernel function is responsible for distributing the probability of occurrence induced by the data value  $v_i$ . Various researchers have proposed various kernel functions (e.g., Uniform kernel function, Triangle kernel function, Epanechnikov kernel function, Normal kernel function etc. [23]). Different kernel function distributes the probability of occurrence differently. Interestingly, the choice of a kernel function does not affect the probability density function very much [23], [26]. Typically the kernel function distributes the probability of occurrence into the neighbor data values which reside within a range called *bandwidth* ( $h$ ) (Normal kernel function distributes the probability of occurrence from  $-\infty$  to  $+\infty$  [23]). A kernel function along with the bandwidth ( $h$ ) (is denoted by  $k_h(x)$  where  $k(x) = hk_h(x)$ ). Although the choice of the kernel function is not very significant, the choice of the bandwidth is very important for probability density function estimation. A detailed discussion about the choice of kernel function and bandwidth selection can be found in [23]. In our approach we choose a data-based approach for bandwidth selection. Scott's rule provides a data-based bandwidth selection where  $h = \sqrt{5}\sigma n^{-1/5}$  where  $\sigma$  is the standard deviation and  $n$  is the number of data points used for density estimation [26].

In a kernel estimator the probability of occurrence is distributed into the equal number of neighbor values for each data point, but in a variable kernel estimator the probability of occurrence is distributed into different number of neighbor values for each data point. Hence at any specific point of time, if data values are close to each other (in terms of value) the bandwidth becomes small and if the data points are far (in terms of value) from each other the bandwidth becomes large. Let  $(x_1, x_2, \dots, x_n)$  be our data points with values  $(v_1, v_2, \dots, v_n)$  at times  $(T - n, T - n + 1, \dots, T)$ , and our corresponding bandwidths be  $(h_1, h_2, \dots, h_n)$ . The probability density function ( $f(x)$ ) at time  $T$  becomes equation (2) where  $f_T(x)$  is the probability distribution function at time  $T$ . In our approach we use variable kernel estimator.

$$f_T(x) = \frac{1}{n} \sum_{i=1}^n k_{h_i}(v_i - x) \quad (2)$$

The use of variable kernel estimator is twofold: the variable kernel estimator offers variable bandwidth for each data points, therefore the bandwidth can be computed on-the-fly using Scott's rule for each data point and the variable kernel selects the bandwidth based on recent data values only.

We modify the variable kernel estimator to address the temporal characteristic of data streams. Recent data points are more interesting than old data points; therefore, when we estimate the probability density function we need to consider the freshness of data points. Heuristically, the recent data items should have more *weight* than the old data points [22], [20], [27]. Here weight is defined as how a data point contributes to the probability density function; thus, in our probability density function, instead of giving all data points the same weight we weight them according to their freshness. The most recent data point receives the highest weight while the oldest one receives the lowest weight. Exponential forgetting is a weight assigning scheme which gives more weight to the recent data points and less weight to the old data points and the weight is decreasing exponentially from present to past [28]. According to exponential forgetting the relative weight among two consecutive data points is constant, called forgetting factor ( $\lambda$ ) where  $0 < \lambda \leq 1$ . Among the two consecutive data points, the recent data point receives weight 1 and the old one receives weight  $\lambda$ . In case of a series of data points, at any particular time the most recent data point receives the weight 1 and all other data points receive the weights according to their relative positions to the most recent data point. If  $(x_1, x_2, \dots, x_n)$  are the data points with data values  $(v_1, v_2, \dots, v_n)$ , at time  $(T - n, T - n + 1, \dots, T)$  respectively, the corresponding weights are  $(\lambda^{n-1}, \lambda^{n-2}, \dots, 1)$ . We weight the kernel function with an exponential forgetting factor. Adding the exponential forgetting factor  $\lambda$  to the equation (2), the probability density function becomes equation (3) where  $\sum_{i=1}^n \lambda^{n-i}$  is the total weight.

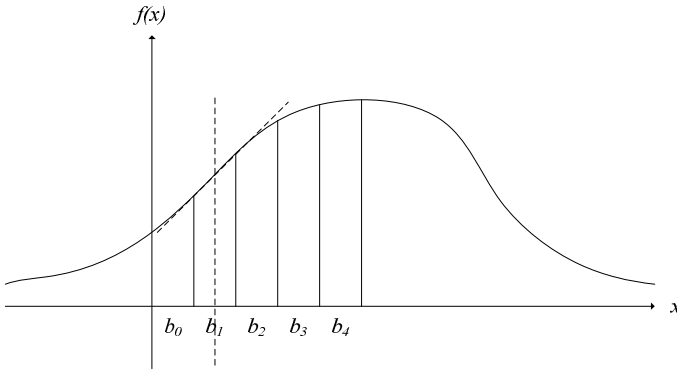
$$f_T(x) = \frac{\sum_{i=1}^n \lambda^{n-i} k_{h_i}(v_i - x)}{\sum_{i=1}^n \lambda^{n-i}} \quad (3)$$

One advantage of using exponential forgetting factor is that it can be computed incrementally, which eases one-the-fly implementation for data streams [28]. The  $\lambda$  is the parameter which decides how many data points contribute to the probability

estimation; the value 0 implies no history, only the previous data point, while value 1 implies all previous data points. Brailsford *et al* [28] proposed a  $\lambda$  selection scheme based on a bootstrap method; we adopt this approach for  $\lambda$  selection. The details about and  $\lambda$  selection are omitted due to page limitation, the detail can be found in [28]. The next section (3.3) discusses the online implementation of our proposed probability density function.

### 3.3 Implementation of Proposed Probability Density Function

The kernel estimator requires a large amount of computation. Binned implementation is a popular, fast implementation for the kernel estimator [13]. In this approach the entire range of data points is divided into some equally spaced bins and data are distributed into bins. Each bin has a representing value and all the data point in a bin are represented by the representing value. The key idea is that lots of values are practically close to each other and binned implementation reduces the number of evaluations; but this popular binned implementation still requires multiple passes and cannot be computed incrementally.



**Fig. 1.** Binned implementation of kernel estimator

In our approach we also divide the entire range of data values into equally spaced bins. A representing value is selected for each bin ( $b_0, b_1, b_2, \dots$  in the Figure 1). Instead of binning the data points, for each bin, we store the value of probability density function of the representing value  $b_i$ ,  $f(b_i)$  and the derivative of the probability density function  $f'(b_i)$ .  $f(b_i)$  and  $f'(b_i)$  are stored for each representing value  $b_i$ .  $f(b_i)$  and  $f'(b_i)$  are the sum of the value of the kernel function and the sum of the derivative of the kernel function at representing value  $b_i$ , respectively. The kernel function and the derivative of the kernel function for each representing value are computed on-the-fly and added to the previous sum; hence this is an online incremental implementation.

Fig 1 shows the binned implementation of our proposed probability density function. By carefully selecting the bin width we can assume each bin is a trapezoid as shown in Figure 1 and we can approximate the probability of any value within a

```

1  procedure update(dataItem d, timestamp t)
2     $s_1 \leftarrow \lambda s_1 + d$ ; //  $s_1$  is the sum of data value and  $\lambda$  is our forgetting
   factor
3     $s_2 \leftarrow \lambda s_2 + d^2$ ; //  $s_2$  is the sum of the square of the data value
4     $\omega \leftarrow \lambda \omega + 1$ ; //  $\omega$  is the total data weight
5     $\mu_1 \leftarrow s_1/\omega$ ; //  $\mu_1$  the first moment
6     $\mu_2 \leftarrow s_2/\omega$ ; //  $\mu_2$  the second moment
7     $\sigma \leftarrow \sqrt{\mu_2 - \mu_1^2}$ ; //  $\sigma$  is the standard deviation
8     $h \leftarrow \sqrt{5}\sigma\omega^{-1/5}$ ; //  $h$  is the bandwidth
9     $c \leftarrow h/\text{binWidth}$ ; //  $c$  is the cell count
10    $b \leftarrow \text{indexLookup}(d)$ ; //  $b$  is the middle cell
11   for  $i = b - c$  to  $b + c$ , //  $i$  is the index of the cell, where  $i \geq 0$ 
   and  $i \leq \text{maximum index}$ .
       //  $b_i$  is the representing value of the bin/cell( $c_i$ ) and  $\alpha_i$  and  $\beta_i$  are
   the starting value and the end value of the bin.
       // distance between two consecutive time stamp is 1.
12    $c_i[f(b_i)] \leftarrow \lambda^{(t - c_i[\text{timestamp}])} c_i[f(b_i)] + k_h(d - b_i)$ ;
13   if ( $k_h(d - x_i)$  is not discontinuous at  $x_i$ )
14      $c_i[f'(b_i)] \leftarrow \lambda^{(t - c_i[\text{timestamp}])} c_i[f'(b_i)] - k'_h(d - b_i)$ ;
15   else
16      $c_i[f'(b_i)] \leftarrow \lambda^{(t - c_i[\text{timestamp}])} c_i[p'(b_i)] - (k_h(d - \beta_i) -$ 
 $k_h(d - \alpha_i))/\text{binWidth}$ ;
17    $c_i[\text{timestamp}] \leftarrow t$ ;
18   end for
19   end procedure
20   procedure indexLookup(dataItem d)
21     return  $\lfloor (d - \text{minValue})/\text{binWidth} \rfloor$ ;
22   end procedure
23   procedure probability(x)
24      $i \leftarrow \text{indexLookup}(x)$ ;
25      $f(x) \leftarrow (c_i[f(b_i)] + c_i[f'(b)](x - b_i))/\omega$ ;
26     return  $p(x)$ ;

```

**Fig. 2.** Update and probability of occurrence lookup algorithm

bin. The top of the trapezoid is a straight line (shown in Figure 1 as the dotted line touching the probability density function) and we store the passing point as well as the derivative; hence using the straight line equation of the line we can estimate the probability of occurrence of any data value within a bin. The bin width should be such that the average error is minimum. Fan and Marron [13] stated that four hundred bins is often optimal, fewer than four hundred bins often deteriorates the quality of the results and more than four hundred bins offer very little improvement. In our approach we use the optimal four hundred bins. Due to page limitation we omit the detail discussion about bin width selection.



The data structure for binned implementation of probability density function is composed of grid cells. As each time a data point comes in, we update the necessary grid cells on-the-fly. Each cell corresponds to a bin. Each cell contains the value of probability density function at  $b_i$ ,  $f(b_i)$ , derivative of the probability density function  $f'(b_i)$  at  $b_i$  and the timestamp ( $t$ ) when the cell is last updated. The next section (3.3.1) describes the algorithms for updating the probability density function using our data structure and computing the probability of occurrence of a data value.

### 3.3.1 Algorithms

Figure 2 shows the online incremental update and probability of occurrence lookup algorithms for our proposed probability density function and outlier detection technique. The update algorithm updates the data structure as each data point comes in and the probability computation algorithm computes the probability of occurrence of a given value ( $x$ ). The update algorithm takes a data point and its timestamp as input. It starts with the updating of the weighted summation (lines 2 & 3), where  $s_1$  is the weighted summation of the data values and  $s_2$  is the weighted summation of the square of the data values. The  $\omega$  in line 4 is the total weight of the data.  $s_1$  and  $s_2$  are required to calculate the current standard deviation ( $\sigma$ ) and hence the bandwidth ( $h$ ). In line 9 we calculate the number of cells we need to update. Some kernel function updates the values in the range from  $-\infty$  to  $\infty$  (e.g., Normal kernel function [23]); in that case we restrict it to  $minValue$  and  $maxValue$ , which represent the minimum and maximum allowable values for a data point, respectively. Now for each bin we update the sum of the kernel function and the latest timestamp when the bin is updated. If the kernel function is continuous at the representing point ( $b_i$ ) then we store the derivative of the kernel function at  $b_i$  else we store the gradient from the starting point ( $\alpha_i$ ) to the end point ( $\beta_i$ ) of the bin. The probability lookup algorithm is fairly simple; it finds the appropriate bin which contains the sum of the kernel function values. Finally the probability is achieved by dividing the sum of the kernel function values by the total weights.

## 4 Performance Analysis

We conducted experiments using a real dataset collected from California Irrigation Management to compare the performance of our algorithm in terms of detection accuracy and execution time with that of the two existing algorithms: We compare our algorithms (DBOD-DS) with two other algorithms ART [8] and ODTS [3] from the literature. ART is an auto-regression based outlier detection technique for wireless sensor network which estimates the value using an auto-regression model and compare the estimated value with the data value received; if the distance is greater than a user defined threshold the data point is identified as outlier., The ODTS is an outlier detection technique for time series. It uses a sliding window to store the recent subset of the data and compare each data point with median value, if the distance is greater than user defined threshold the point is identified as outlier. Since ODTS uses a sliding window, we run the experiments with the window sizes 10, 15, 20, ..., 100 and report the average performances. In this section, we first describe the dataset and simulation model, and then present the experimental results.

## 4.1 Dataset

The California Irrigation Management Information System (CIMIS) manages a network of over 120 automated weather stations in California [7]. Each weather station collects data in every minute and calculates hourly and daily values. The data are analyzed and stored in the CIMIS database and publicly available. The measured attributes are solar radiation, air temperature, relative humidity, wind speed, soil temperature, etc. For our experiments, we use the daily soil temperature data collected from 1998 to 2009, and implanted the random synthesized outliers in them along with inherent outliers. We use fifty stations in random and report the average results. On average each station has 4000 rounds of data (total 200,000 data rounds). The first 500 data points are used for bootstrapping from each data stream. This dataset has consecutive rounds of inherent outliers. We use 7% outliers for all of our experiments, except for those experiments in which we vary the percentage of outliers to study its impacts on the algorithms' performance.

## 4.2 Simulation Model

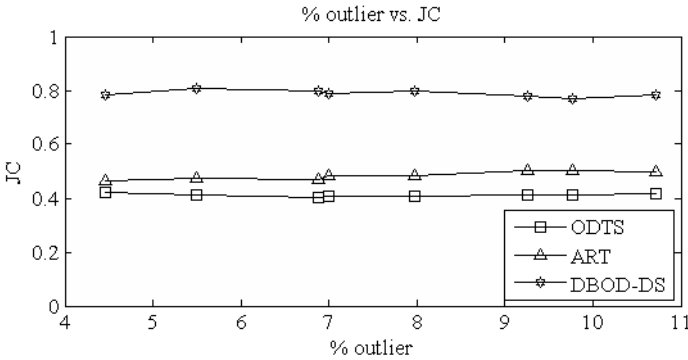
In our simulation model we mimic the typical data streams architecture. Each data source produces one data stream. We create the virtual data sources and the virtual base station. Each virtual data source obtains a data value at a fixed interval and sends it to the virtual base station. The virtual base station receives one data point from one data stream at a time and processes it. We execute DBOD-DS, ART and ODTS, one technique at a time, at the base virtual base station to detect the outliers. The entire simulation model is built on the Java platform and we ran the simulation using GNU Compiler for the Java version 1.4.2. The GNU was running on Red Hat Linux Enterprise 5 [29]. We use the Cluster Supercomputer at the University of Oklahoma to run our simulation experiments. The comparison is fair since each technique is run on the same machine.

## 4.3 Accuracy

We measure the accuracy in terms of Jaccard Coefficient (JC) and Area Under the receiver operator characteristic Curve (AUC). A good outlier detection technique is the one which maximizes true positive (TP) and minimizes false negative (FN) and false positive (FP). Basu and Meckesheimer [3] proposed the use of Jaccard Coefficient (JC) as a performance metric for outlier detection. Mathematically JC defined as  $JC = \frac{TP}{FP+FN+TP}$ . The metric (JC) consider the true positive, false negative and false positive. JC is inversely proportional to the wrong classification and directly proportional to the correct classification, and assigns equal weight to the correct classification and the wrong classification [3]. So, the better JC an outlier detection algorithm yields the more accurate results the algorithm provides. However, JC is not independent of the distribution of inliers and outliers. We use the receiver operator characteristic (ROC) curve to establish a distribution independent performance metric. On top of this, ROC curve has two other fascinating properties: 1) the ROC curve is not sensitive to a particular choice of the cut-off value and 2) the Area Under the ROC curve (AUC) provides a single scalar value which represents the

performance of the classifier [12]. The ROC curve is a two dimensional graph in which the true positive rate (TPR,  $TPR = \frac{TP}{TP+FN}$ ) goes along the y-axis and the false positive rate (FPR,  $FPR = \frac{FP}{FP+TN}$ , TN is true negative) goes along the x-axis. TPR is the rate of correct classification (called benefit) and FPR is the wrong classification (called cost); hence the ROC curve is the graph of cost vs. benefit [12]. The algorithm which has higher AUC is considered as a better algorithm. The optimal algorithm will increase the TPR without increasing the FPR; if we push it further it will increase the FPR only because there is no room for improvement of TPR; hence the graph will be two line segments joining (0,0) to (0,1) which is called conservative region and (0,1) to (1,1) which is called flexible region. So the better algorithm will follow the curve of the optimal algorithm. The results for ART, ODTS and DBOD-DS are reported for the optimal cut-off value which maximizes the Jaccard Coefficient of the respective algorithms. The next two sections (4.3.1 and 4.3.2) compare the three algorithms in terms of JC and ROC, respectively.

**4.3.1 Jaccard Coefficient (JC)**

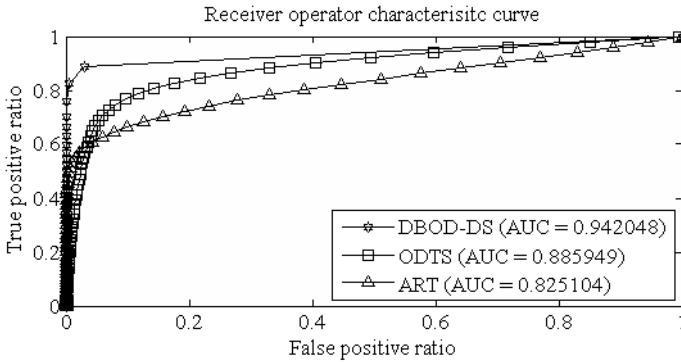


**Fig. 3.** JC of each algorithm

Figure 3 shows the Jaccard Coefficient with respect to different percentages of outliers for our dataset. DBOD-DS outperforms all other algorithms regardless of the percentage of outliers. The JC for DBOD-DS is almost twice of the JC of the other two algorithms. DBOD-DS, ART and ODTS show constant JC with respect to change of the percentage of outliers. If the percentage of outlier increases, the true positive increases along with false negative and false positive; hence the increment of the numerator and denominator in the JC formula makes JC constant with respect to the percentage of outliers.

**4.3.2 Receiver Operator Characteristic Curve**

Mostly the performance of the ART, ODTS and DBOD-DS depends on the correctness of their respective thresholds. Hence, we compare DBOD-DS with those two algorithms to establish a parameter less comparison metric. Figure 4 shows the



**Fig. 4.** Receiver operator characteristic curve

ROC curve for DBOD-DS, ART and ODTs for the dataset. DBOD-DS performs very well in the conservative region [12]; it correctly identifies the outliers without increasing the false positive ratio after the true positive ratio reaches 0.8, which is very close to the optimal performance. The optimal performance in the conservative region resembles the fact that DBOD-DS is capable of identifying true positives without increasing false positives (the sharp transition from the conservative region to the flexible region in Figure 4 confirms this fact). The most important plus point for the ROC curve is that the area under the curve (AUC) resembles a single metric for performance comparison among two classifiers. The AUC for DBOD-DS is 0.94 and the AUC for ART and ODTs are 0.82 and 0.88, respectively. Interestingly, the performance of ODTs is better than that of ART in terms of AUC. This is because ODTs produces fewer false negatives than ART. The most appealing characteristic of AUC is that it resembles the probability of correct classification regardless of the percentage of outliers; therefore, in terms of AUC, DBOD-DS is much more superior to ART and ODTs.

#### 4.4 Execution Time

The DBOD-DS performs much better than the other two algorithms in terms of JC and AUC, but this performance benefit does not come without cost. The DBOD-DS takes more execution time compared to ART and ODTs. Figure 5 shows the execution time for the algorithms with respect to the change of the percentage of outliers. The time is recorded for each round from receiving a data point to identifying its outlier-ness. On an average DBOD-DS takes twice more time than ART and 20 times more time than ODTs but the execution time for DBOD-DS is less than 1.5 milliseconds. The outlier detection takes place within two rounds and this time is practically enough for any type of data stream. In a typical data stream application the data source is kept onsite and the data values travel from the data source to the base station. Sending frequency lower than 1 millisecond is impractical for most of the current data stream applications. The execution time increases a little bit with the increase of the percentage of outlier; this is because if the percentage of outliers increases, the dispersion of the probability density function increases, hence more bin

needs update for each data point. In our opinion the extra time is worthy for DBOD-DS because it offers a significant performance improvement over ART and ODTs in terms accuracy.

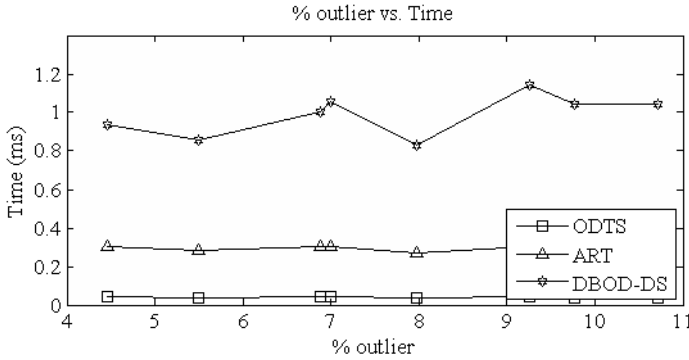


Fig. 5. Execution time for each algorithm with respect to percentage of outliers

## 5 Conclusions and Future Research

We have developed an outlier detection algorithm for data stream applications based on our novel probability density estimation function. The performance of our algorithm compared with that of the existing algorithms in the literature is shown by extensive empirical studies on a real dataset. Our algorithm outperforms the existing algorithms in terms of accuracy, but requires more time to execute. However, the time our algorithm needs is less than 1.5 milliseconds, which is much smaller than the time required sending and receiving data in many data stream applications. From our empirical studies it is clear that our algorithm can perform excellently for a reasonable percentage of outliers. Even though we designed the algorithm considering both single dimensional data and multi dimensional data, our experiments so far have focused on the former case. In our future experiments we want to cover multi dimensional data. In addition, we want to extend our novel probability density function to estimate the data values and to detect concept drifts. In our technique we require user-defined parameters to identify outliers; it would be interesting to make our approach completely intelligent so that it would not expect any parameter from the user.

## References

1. Anguiulli, F., Fassetti, F.: Detecting Distance-Based Outliers in Streams of Data. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 811–820 (2007)
2. Babcock, B., Babu, S., Mayur, D., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: Proceedings of 21st ACM Symposium on Principles of Database Systems (PODS 2002), pp. 1–16 (2002)

3. Basu, S., Meckesheimer, M.: Automatic outlier detection for time series: an application to sensor data. *Knowledge Information System*, 137–154 (2007)
4. Barnett, V., Lewis, T.: *Outliers in Statistical Data*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons Inc., Chichester (1994)
5. Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.: *Monitoring Streams – A New Class of Data Management Applications*. In: *Proceedings of the 28th International Conference on Very Large Data Bases*, pp. 215–226 (2002)
6. Chandola, V., Banarjee, A., Kumar, V.: *Outlier Detection: A Survey*. Technical Report, University of Minnesota (2007)
7. California Irrigation Management Information System, web-link, <http://www.cimis.water.ca.gov/cimis/welcome.jsp> (accessed January, 2010)
8. Curiac, D., Baniyas, O., Dragan, F., Volosencu, C., Dranga, O.: *Malicious Node Detection in Wireless Sensor Networks Using an Autoregression Technique*. In: *Proceedings of the Third International Conference on Networking and Services*, pp. 83–88 (2007)
9. Eskin, E.: *Anomaly Detection over Noisy Data using Learned Probability Distributions*. In: *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 255–262 (2000)
10. Franke, C., Gertz, M.: *Detection and Exploration of Outlier Regions in Sensor Data Streams*. In: *IEEE International Conference on Data Mining Workshop*, pp. 375–384 (2008)
11. Franke, C., Gertz, M.: *ORDEN: outlier region detection and exploration in sensor networks*. In: *Proceedings of the 35th SIGMOD International Conference on Management of Data*, pp. 1075–1078 (2009)
12. Fawcett, T.: *Roc graphs: Notes and practical considerations for data mining researchers*. Technical report hpl-2003-4, HP Laboratories, Palo Alto, CA, USA (2003)
13. Fan, J., James, S.J.S.: *Fast Implementations of Nonparametric Curve Estimators*. *Journal of Computational and Graphical Statistics* 3(1), 35–56 (1994)
14. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: *Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries*. In: *Proceedings of the 27th International Conference on Very Large Databases*, pp. 79–88 (2001)
15. Guha, S., Koudas, N.: *Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation*. In: *Proceedings 18th International Conference on Data Engineering*, pp. 567–676 (2002)
16. Ishida, K., Kitagawa, H.: *Detecting current outliers: Continuous outlier detection over time-series data streams*. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) *DEXA 2008*. LNCS, vol. 5181, pp. 255–268. Springer, Heidelberg (2008)
17. Jiang, N., Gruenwald, L.: *Research issues in Data Stream Association Rule Mining*. *ACM Sigmod Record* 35(1), 14–19 (2006)
18. Keogh, E., Lin, J., Truppel, W.: *Clustering of Time Series in Meaningless: Implications for Previous and Future Research*. In: *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 56–65 (2003)
19. Knorr, E.M., Ng, R.T.: *Algorithms for Mining Distance-Based Outliers in Large Datasets*. In: *Proceedings of the 24th International Conference on Very Large Data Bases*, pp. 392–403 (1998)
20. Madsen, H.: *Time Series Analysis: Texts in Statistical Science*. Chapman & Hall/CRC (2007)

21. Ng, R.T., Han, J.: Efficient and Effective Clustering Methods for Spatial Data Mining. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 144–155 (1994)
22. Puttagunta, V., Kalpakis, K.: Adaptive Methods for Activity Monitoring of Streaming Data. In: Proceedings of International Conference on Machine Learning and Applications, pp. 197–203 (2002)
23. Scott, D.W.: Multivariate Density Estimation. A Wiley-Interscience Publication, Hoboken (1992)
24. Sheikholeslami, G., Chatterjee, S., Zhang, A.: WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In: Proceedings of the 24rd International Conference on Very Large Data Bases, pp. 428–439 (1998)
25. Stonebraker, M., Çetintemel, U., Zdonik, S.: The 8 Requirements of Real-Time Stream Processing. ACM SIGMOD Record 34(4), 42–47 (2005)
26. Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Online Outlier Detection in Sensor Data Using Non-Parametric Models. In: Proceedings of the 32nd International Conference on VLDB, pp. 187–198 (2006)
27. Gruenwald, L., Chok, H., Aboukhamis, M.: Using Data Mining to Estimate Missing Sensor Data. In: Seventh IEEE International Conference on Data Mining Workshops, pp. 207–212 (2007)
28. Brailsford, T.J., Penm, J.H.W., Terrell, R.D.: Selecting the forgetting factor in Subset Autoregressive Modelling. *Journal of Time Series Analysis* 23, 629–650 (2002)
29. OU Supercomputer Resources: web-link,  
<http://www.oscer.ou.edu/resources.php> (accessed May, 2010)

# Effective Bitmap Indexing for Non-metric Similarities

Claus A. Jensen, Ester M. Mungure, Torben Bach Pedersen,  
Kenneth Sørensen, and François Deliège

Department of Computer Science, Aalborg University

**Abstract.** An increasing number of applications include recommender systems that have to perform search in a non-metric similarity space, thus creating an increasing demand for efficient yet flexible indexing techniques to facilitate similarity search. This demand is further fueled by the growing volume of data available to recommender systems.

This paper addresses the demand in the specific domain of music recommendation. The paper presents the *Music On Demand framework* where music retrieval is performed in a continuous, stream-based fashion. Similarity measures between songs, which are computed on high-dimensional feature spaces, often do not obey the triangular inequality, meaning that existing indexing techniques for high-dimensional data are infeasible.

The most prominent contribution of the paper is the proposal of an indexing approach that is effective for non-metric similarities. This is achieved by using a number of bitmap indexes combined with effective bitmap compression techniques. Experiments show that the approach scales well.

## 1 Introduction

Recommender systems are becoming increasingly present in many applications. The growing volume of data available to these recommender systems has created an increasing demand for efficient yet flexible indexing techniques able to facilitate similarity search. However, in many emerging domains, the similarity spaces are non-metric, meaning that existing approaches are not feasible. Therefore, a major challenge is to develop an indexing technique for non-metric similarity spaces. This paper tackles this challenge in the specific context of music recommendation, but the approach can be applied to a wide variety of systems.

The current tendency in music distribution is that personal music collections are being replaced by the notion of *streaming music* from commercial on-line music providers. Here, the challenge is to support query functionalities on *vast music collections* where only limited or no prior knowledge about the content of the music collection is available. However, similarity measures between songs are most often complex formulas computed on high-dimensional feature spaces. These similarity measures, such as the Earth Mover's Distance, often behave "strangely", as they are non-metric. In particular, the triangular inequality often does not hold. This means that existing indexing techniques for high-dimensional data are infeasible. The aim of this paper is to propose an indexing solution that supports the "worst case", namely *non-metric* similarity measures, well.



This paper introduces the *Music On Demand framework*, referred to as the MOD framework, and presents two major contributions. The first contribution is an indexing approach that is effective for non-metric similarity measures. The second major contribution is effective query processing techniques to perform similarity search. Our approach essentially relies on using bitmap indexes combined with effective bitmap compression techniques.

This approach ensures efficient management of both metadata and content-based similarity. Representing the entire music collection as well as subsets thereof as bitmaps, we are able to use bit-wise operations to ensure efficient generation of multi attribute subsets representing, e.g., all songs by Madonna released this year. These subsets may in turn be applied as restrictions to the entire music collection. Similarly, using bitmaps to represent groupings of similar songs with respect to a given base song, we are able to identify and retrieve similar/dissimilar songs using bit-wise operations.

Extensive experiments show that the proposed approach scales with respect to the data set size and the number of concurrent requests. Query performance, throughput, and storage requirements are presented on a prototypical version of the MODframework. For example, the prototype system running on a standard laptop is able to support 36,000 simultaneous users on a database of 100,000 songs. Comparing our implementation with an equivalent B-tree solution, we improve the query execution time by an order of magnitude on a music collection containing 100,000 songs.

The paper is organized as follows. Section 2 describes related work. Section 3 presents an informal description of a data and query model for dynamic playlist generation. In Section 4, we elaborate on the application of bitmaps followed by an examination of the associated query processing techniques in Section 5. In Section 6, we discuss and evaluate the experiments conducted using bitmap indexing. Finally, in Section 7 we conclude and present directions for future work.

## 2 Related Work

Within the field of Music Information Retrieval, much effort has been put into the task of enabling music lovers to explore individual music collections [12, 14]. Within this context, several research projects [17, 13] have been conducted in order to pursue a suitable similarity measure for music, for which purpose a feature representation of the musical content is required. In accordance with the different feature representations of musical content, the current research is going in the direction of automating the task of finding similar songs within music collections [2, 18]. However, due to the subjectiveness of musical perception, the similarity measure described disobey the triangular inequality and are thus said to be *non-metric*. Several non-metric similarity measures exist [3, 11, 21].

When considering indexing of high dimensional musical feature representations, existing indexing techniques such as, e.g., the M-grid [7] and the M-tree [6] can be applied. However, as discussed above, the *triangular inequality* property of the metric space typically cannot be obeyed for a similarity measure. Hence, as the M-tree and the M-grid, and many other high-dimensional indexing techniques, rely on the use of a metric space, they turn out to be insufficient. In contrast, the approach presented in this paper does not require the triangular inequality to hold.

To ensure efficient retrieval of read-mostly data, bitmap indexes are popular data structures for use in commercial data warehouse applications [9]. In addition, bitmap indexes are used with respect to bulky scientific data in order to represent static information. One approach is related to High-Energy Physics [22]. However, to our knowledge, bitmap indexing has so far not been applied within music retrieval.

Many different approaches exist to support accurate music recommendation. A first approach uses musical content to find similar songs, where immediate user interaction in terms of skipping behavior is used to restrict the music collection [19]. Unlike this approach we do not rely on the actual distances when determining what song to return, as songs are clustered into groups of similar songs. A second approach maps the task of finding similar songs to the Traveling Salesman problem (TSP) [21]. A single circular playlist consisting of all tracks from the entire music collection is generated, and the ability to intervene in the construction of playlist is taken away from the listener. In contrast, the single song approach presented in this paper, ensures that the construction of a playlist may be influenced dynamically. Unlike the present paper, none of these two approaches provide an indexing approach capable of handling arbitrary similarity measures.

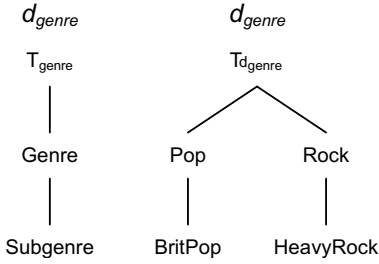
Finally, in most commercial media players such as Winamp<sup>TM</sup>, the metadata of music presumes a flat structure. However, to enable an enriched description of the metadata of music, we choose explicitly to view metadata in the form of a *multidimensional cube* known from the literature of multidimensional databases [20, 23]. The metadata of music is thus considered as a number of metadata dimensions, modelled in a hierarchical manner, which constitutes a multidimensional cube. Through this approach we are able to select songs in accordance with the individual levels of a given hierarchy of a metadata dimension.

### 3 Data and Query Model

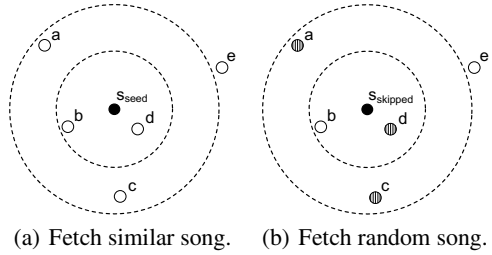
We briefly present the underlying music data model and the associated query functionalities of the MOD framework. The full details can be found in another paper [8].

Initially, we introduce a *metadata dimension* in order to apply an abstraction to a hierarchical representation of the music metadata. As shown in Figure 1, the hierarchical ordering of the metadata is described as two posets (partially ordered sets). The first poset represents the hierarchical ordering of dimension levels and the second poset represents the hierarchical ordering of the dimension values.

A metadata dimension consists of both dimension levels and dimension values, where a dimension level has a number of associated dimension values. Using posets to model hierarchies we achieve that both regular and irregular dimension hierarchies are supported. Irregular hierarchies occur when the mappings in the dimension values do not obey the properties stating that a given hierarchy should be *onto*, *covering* and *strict* [20]. Informally, a hierarchy is *onto* if the hierarchy is balanced, *covering* when no paths skip a level and *strict* if a child in the hierarchy has just one parent. The metadata of music is composed of descriptive attributes such as artist, title, etc. The metadata attributes are presented as dimension values where a metadata item and the corresponding schema are defined. To ensure that the model supports querying with respect to traditional navigational methods as well as musical similarity, a *song* is defined in terms of



**Fig. 1.** Schema (left) and instance (right) for the metadata dimension  $d_{genre}$



**Fig. 2.** Usage of the distance store

both tangible music metadata and musical content. In this context, metadata represents elements such as artist, genre, etc. and musical content corresponds to an arbitrary feature representation of the music. In addition, an arbitrary distance function  $dist$  may be used to calculate the content-based similarity between two songs with respect to their associated feature representations. The distance function is allowed to be *non-metric* as long as the identity property is retained, i.e.,  $dist(x, x) = 0$ .

When considering the content-based similarity between songs, we introduce the *distance store* as an abstraction over an arbitrary distance function. A distance store is a complete partitioning of the distance domain, implying that no partitions are omitted and that no partitions should overlap. Thus, each partition constitutes a unique and non-overlapping distance interval.

The two retrieval operators *SimilarSong* and *RandomSong* constitute the main point of interacting with the framework. The purpose of *SimilarSong* is to retrieve a song similar to a given seed song, while in the same time avoiding that the retrieved songs resembles possible skipped songs. As the name indicates, the task of *RandomSong* is to retrieve a randomly chosen song from the music collection. In this connection the aspects of skipped songs also apply. Moreover, as the listener may choose to intervene in the construction of the playlist at any point in time, either of the operators only return a single song at a time. To fetch a similar song, using the *SimilarSong* operator, we are initially presented a specific seed song as shown in Figure 2(a). Assuming that each circle represents a partition of the distance store associated with song  $s_{seed}$ , we are to return a song from the innermost partition containing valid songs. In this context, a valid song is a song neither restricted nor skipped. As any song within the appropriate partitions are valid candidate songs, either song  $b$  or  $d$  may be returned as a song similar to  $s_{seed}$ . However, say that  $b$  is more similar to an already skipped song,  $d$  is the better candidate.

The distance store shown in Figure 2(b) constitutes a composite distance store representing all skipped songs as discussed above. When fetching a randomly chosen song, using the *RandomSong* operator, we initially pick a number of candidate songs among the valid songs in the music collection. In this case songs  $a$ ,  $c$  and  $d$  are picked. As the candidate songs are chosen randomly within a vast music collection, chances are that even a small number of songs, e.g., 10, ensures retrieval of an acceptable song. Locating the positions of the candidate songs within the composite skip distance store we return the song most dissimilar to any of the skipped songs. In this case either song  $a$  or  $c$  is returned.

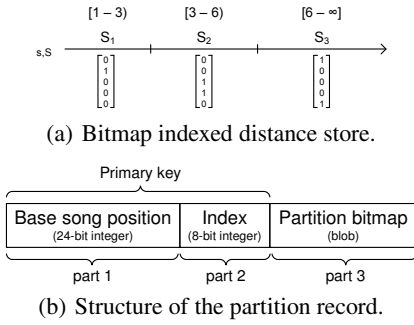


Fig. 3. A distance store and a partition record

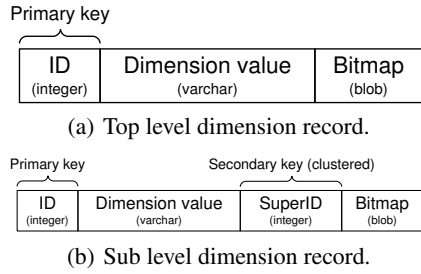


Fig. 4. Structure of the metadata records

Finally, the MOD framework apply the ability to restrict the entire music collection in accordance with relevant music metadata such as genre, artist, etc.

## 4 Distance and Metadata Indexes

### 4.1 Distance Management

As indicated earlier,  $n$  distance stores are required for a music collection containing  $n$  songs, and each distance store has to hold all  $n$  songs. Thus, to cope with this  $n^2$  space complexity a compact representation of the subsets is greatly needed. Moreover, as we deal with vast music collections that potentially may contain millions of songs, latency may occur when querying songs. By use of bitmap indexes [5], we have obtained not only a compact representation of the songs but moreover a very good query performance of the implemented music data model reducing the overall latency.

Assume now that a certain known order of the songs within a music collection exists, e.g., the order of insertion. A subset of songs from the music collection can then be represented by a bitmap, i.e., a sequence of bits, following the same order as the order of the songs within the music collection, where 1 (set) bits are found only for the songs contained in the subset. Thus, aside from representing the overall music collection of available songs, bitmaps may moreover be used to represent subsets of the music collection such as songs having a similar metadata attribute, the skipped songs or the songs contained in the history of played songs. Moreover, having a known order of the songs within the music collection a single song is uniquely identified by its position within the music collection, indicating that the first song is located at position one. In the following we assume a 32 bit computer architecture, whereby a single bitwise instruction computes 32 bits at once.

Using the *equality encoding* scheme for bitmap indexes, each distinct attribute value is encoded as one bitmap having as many bits as the number of records in the relation, i.e., the music collection [1, 5].

Selecting music in accordance with multiple attributes across several relations, bitwise bitmap operations may be used to replace expensive joins performed between the involved relations [15, 16]. Suppose that a listener wishes to select all music performed

by the artists Madonna and U2 released in the year 2005. For this particular example, a bit-wise OR is performed on the appropriate bitmaps of the artist relation in order to generate the combined bitmap representing the songs performed by both Madonna and U2. In addition, performing a bit-wise AND on the combined bitmap and the bitmap representing all songs released in the year 2005 associated with the release relation, the wished selection is achieved. Additionally, using bitmaps to represent the history of played songs and the collection of skipped songs, bit-wise operations may be used to ensure that neither songs recently played nor songs contained in the collection of skipped songs are returned to the listener.

Even though bitmap indexes constitute a compact representation of bulky data the nature of bitmaps provides the means for ensuring an even more compact representation by applying different techniques. In this paper, we discuss the use of two well-known bitmap compression scheme called the *WAH (Word-Aligned Hybrid)* [24], and an alternative representation technique known as *AVD (Attribute Value Decomposition)* [5].

In Section 3 the concept of the distance store was initially presented. To elaborate on the technical aspects of the distance management, this section describes how a number of distance stores constitute the handling of the distances between the songs managed by the MOD framework.

As described in Section 3, a distance store consists of a number of partitions each corresponding to an associated distance interval. Hence, each partition is represented by a single bitmap indicating the songs belonging to the associated distance interval. The collection of bitmaps required for a single distance store, constitutes a bitmap index for the distances of the songs with respect to the base song of the distance store. In Figure 3(a) an abstraction of a bitmap index of a distance store is illustrated for a collection of 5 songs having song  $s$  as the base song for the distance store. The music collection  $S$  is grouped into separate subsets  $S_1$  through  $S_3$  which constitute the individual partitions and their associated distance intervals. Considering the aspects of the WAH compression scheme, we need to consider whether a compression of the distance store is achievable. For that purpose we initially turn our attention to the structure of the partition record shown in Figure 3(b). Part 1 represents the positions of a given song and part 2 represents a numbered index indicating the position of the partition within the distance store. Together these constitutes a composite primary key. Part 3 contains the partition bitmap, which identifies all songs contained in the partition defined by part 1 and 2.

The space occupied by the first two parts of the partition record increases linearly as the number of songs increases. The space occupied by the partition bitmap, i.e., the third part in the partition record presented, constitutes the most crucial part of the total space required for the distance management. In this worst-case analysis it is assumed that the 1 bits within a partition bitmap are located at certain places to achieve the worst possible compression. In addition, all 1 bits are distributed equally among the given number of partitions. To illustrate, having just a single 1 bit represented in each word to compress, it implies that no space reduction is achievable when applying WAH compression, as all compressed words are literal words. As the number of distance stores increases, the bit density in each store decreases and leads to a better compression ratio.

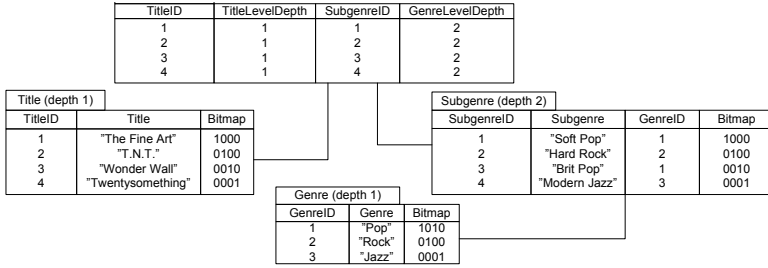


Fig. 5. Snowflake schema having the  $d_{title}$  and  $d_{genre}$  metadata dimensions

## 4.2 Metadata Management

To represent the multidimensional cube in a relational database, we adopt the *snowflake schema* known from multidimensional databases [23]. The snowflake schema is composed of a central fact table and a set of associated dimensions. The snowflake schema satisfies the structure of the metadata hierarchies by allowing a metadata dimension to be represented as a number of dimension tables. Each dimension level in the metadata hierarchy corresponds to a dimension table. While this saves space, it is known to increase the number of dimension tables thus resulting in more complex queries and reduced query performance [10]. However, as the purpose of the multidimensional cube in the MOD framework is to find the bitmaps, no expensive join queries are to be performed, as selections based on multiple attributes are performed by applying bit-wise operations on the corresponding bitmaps.

As stated, a metadata dimension in a relational database is represented as a number of dimension tables, where each dimension table corresponds to a level in a metadata hierarchy. According to the snowflake schema representing the metadata within the MOD framework, there exists two types of relations used as dimension tables. Records of both types of relations can be seen in Figure 4. The *level record* in Figure 4(a) is used for the highest level within each dimension. For efficient access, the relation is defined as clustered having the id attribute as the primary key. The *sub level record* in Figure 4(b) is clustered in accordance with the super id attribute, that is associated with a given superordinate level. This ensures an efficient foundation for hierarchical metadata navigation, as, e.g., the subgenres of a given genre are stored consecutively within the relation. However, as metadata may be accessed using ids, we maintain an index on the id attribute of the relation. The bitmap contained within each of the records, represents the songs which are associated with the dimension value of the records.

In Figure 5 we consider the structure of the snowflake schema representing the fact table and dimension tables discussed above with respect to the metadata dimensions  $d_{title}$  and  $d_{genre}$ . From the fact table it appears that the involved music collection is represented by a bitmap with four bits. The first bit in the each bitmap corresponds to the first song in the managed music collection, the second bit to the second song, etc. Along with the foreign keys in the fact table, the level depths are shown. From these it can be seen, that the most specific dimension value of all the songs corresponds to the bottom level of the hierarchies. In addition, aggregations of the bitmaps from a sub level

to a superordinate level are applied within the dimension hierarchy by use of bit-wise operations. As the bitmaps of the dimensions tables to a great extent contain only few 1 bits, these bitmaps may be highly compressed using the WAH compression scheme.

## 5 Query Processing

Having described the application of the bitmaps for both distance management and metadata management, these bitmaps may be combined in a single query in order to request songs from a restricted music collection. In this section we use pseudo algorithms to describe how to process such queries when requesting similar or random songs.

To describe the retrieval operators *RandomSong* and *SimilarSong* introduced in Section 3, we introduce the two helper functions *GenerateCompSkip* and *FetchRandomSongs*. The task of *GenerateCompSkip* is to cache the composite distance stores representing the distance stores of all skipped songs for each of the individual music players interacting with the MOD framework. The composite distance store representing the distance stores of all skipped songs is denoted as the *composite skip distance store*. Using a unique user id representing a specific music player, the cached composite skip distance store is accessible for retrieval and manipulation. The purpose of *FetchRandomSongs* is to enable the possibility to retrieve a specified number of randomly chosen songs from a given music collection represented by a bitmap.

The music collection initially passed to the respective two retrieval operators is denoted as the *search collection* and constitutes either the entire music collection or a subset of the entire collection. The search collection is a subset of the entire collection if a metadata restriction has occurred. Once the search collection has been restricted by the skipped songs and the songs contained in the history of played songs, the collection of the remaining songs is denoted as the *valid collection*. Performing a further restriction by all songs similar to the skipped songs we end up with a collection of songs denoted as the *candidate collection*.

All restrictions, i.e.,  $p \setminus q$ , are performed using the syntax  $p$  AND ( $p$  XOR  $q$ ) where  $p$  is the collection to restrict and  $q$  is the collection to restrict by. The alternative syntax,  $p$  AND NOT  $q$ , is unusable as the size of the entire music collection can not be derived from the individual bitmaps where consecutive 0 bits are omitted from the end of the bitmaps as described in Section 4. The prefix notation  $b_$  is used to denote a bitmap.

The task of *RandomSong*, is to find a subset of randomly chosen candidate songs from which the song least similar to any of the skipped songs is to be returned. The purpose of the selected candidate songs is to constitute a quality measure for the song to return. The operator *RandomSong* described in Algorithm 1 takes as input parameters three bitmaps representing the current search collection, the history and the set of skipped songs. In addition, an integer  $q$  is passed in order to specify the number of candidate songs among which to choose the song to return. Finally, the operator takes a parameter representing a user id indicating the music player currently interacting with the MOD framework. The id is used to identify a cached composite skip distance store.

The task of *SimilarSong*, is to find and return a single song considered most similar to a given seed song. In this context it is ensured, that no songs close to any skipped songs is returned. As input parameters, the operator *SimilarSong* presented in Algorithm 2 takes three bitmaps representing the search collection, the history and the set

**Algorithm 1.**Pseudo code for *RandomSong*.

---

```

RANDOMSONG
(b_coll, b_hist, b_skip, q, userId)
1 filePath ← empty string
2 songPosition ← Null
3 b_validColl ← b_coll AND
  (b_coll XOR (b_skip OR b_hist))
4 b_randomColl ←
  FETCHRANDOMSONGS(q, b_validColl)
5 compositeSkipDS ←
  GENERATECOMPSKIP(b_skip, userId)
6 for each partition b_p in compositeSkipDS
  starting with the partition representing the least
  similar songs.
7   do ▷ Check if candidate songs are available
8   b_candidateColl ←
    b_randomColl AND b_p
9   if BITCOUNT(b_candidateColl) > 0
10    then ▷ Choose a position for a random song
11    songPosition
    ← RANDOM(b_candidateColl)
12    break
13 if songPosition <> Null
14 then ▷ Fetch the file path for the song found
15   songRecord
    ← FACTTABLELOOKUP(songPosition)
16   filePath
    ← CUBELOOKUP(
      songRecord.filenameID,
      "Filename")
17 return filePath

```

---

**Algorithm 2.**Pseudo code for *SimilarSong*.

---

```

SIMILARSONG
(b_coll, b_hist, b_skip, seedSongPosition, userId)
1 filePath ← empty string
2 songPosition ← Null
3 b_validColl ← b_coll AND
  (b_coll XOR (b_skip OR b_hist))
4 compositeSkipDS
  ← GENERATECOMPSKIP(b_skip, userId)
5 seedSongDS
  ← DISTANCESTORELOOKUP
  (seedSongPosition)
6 b_accSkip ← empty bitmap
7 for each partition b_p in compositeSkipDS
  and b_q in seedSongDS starting with the partition
  representing the most similar songs
8   do ▷ Check if candidate songs are available
9   b_accSkip ← b_accSkip OR b_p
10  b_candidateColl ← validColl AND
  (b_q AND (b_q XOR b_accSkip))
11  if BITCOUNT(b_candidateColl) > 0
12  then ▷ Choose a position for a random song
13  songPosition
    ← RANDOM(b_candidateColl)
14  break
15 if songPosition <> Null
16 then ▷ Fetch the file path for the song found
17  songRecord
    ← FACTTABLELOOKUP(songPosition)
18  filePath
    ← CUBELOOKUP(songRecord.filenameID,
      "Filename")
19 return filePath

```

---

of skipped songs. In addition, the position of the seed song is passed to the operator, stating the position of the song within a bitmap corresponding to all songs in the music collection. Finally, the operator takes a parameter representing an user id indicating the music player currently interacting with the MOD framework. The id is used to identify a cached composite skip distance store.

After generation of the valid collection and the composite skip distance store, the distance store for the seed song is retrieved using the position of the seed song to perform a lookup in the distance management relation (line 5). To find the collection of candidate songs, the seed song distance store is traversed starting with the partition containing the songs most similar to the seed song. This is done while consulting the content of the corresponding partitions associated with the composite skip distance store (line 7 to 14). To ensure that only a song considered least similar to any skipped song is returned, the composite skip distance store is accumulated (line 9). Thus, restricting the partitions of the seed song distance store by the corresponding accumulated partitions of the composite skip distance store, the collection of candidate songs is obtained while only considering the songs contained in the valid collection. (line 10). In the remainder of the algorithm, the position of a selected candidate song is used to retrieve the file path of the associated audio file.



## 6 Experiments

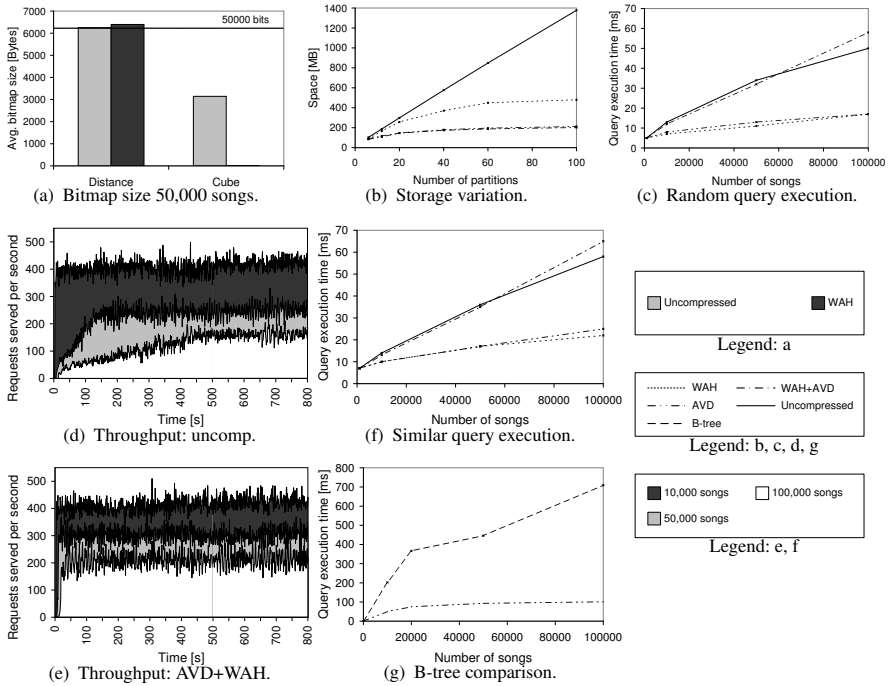
In this section the MOD framework is evaluated using various configurations. The evaluation concerns the space consumption introduced by the framework as well as the query performance when random and similar songs are retrieved. Moreover, we compare the MOD framework to an B-tree equivalent version. In the following, the tests are conducted using the MOD framework implemented with Java and MS SQL Server and are performed on a 32-bit Pentium 4 @ 3.0 GHz dual core with 3.25GB of main memory, running both the query evaluator and the SQL Server. The storage capacity is constituted by 3 x 400GB striped SATA disks. Each disk rotates at 7200rpm. The bitmaps within the databases can be configured as being uncompressed or WAH compressed. Additionally, when concerning the distance management, the bitmaps for the distance stores can be represented using either AVD or not, which gives a total of four different bitmap representations.

For test purposes, synthetic music data is generated and added to the relevant music collections. In connection to this, aspects such as artist productivity, career duration and the number of songs on albums are considered to ensure a real-life reflection of the generated collections. Moreover, upon adding synthetic data to the music collections, entire albums are added in continuation of one another, which resembles the most common way of use. When creating distance stores for synthetic data we apply random distances between the songs. The random distances are chosen such that the number of songs within each of the partitions of the distance stores gradually increases, starting from the partition representing the most similar songs only to decrease at the end. As we assume a highly diversified music collection, only few songs are located in the partitions representing the most similar songs.

### 6.1 Space Consumption

In general, a significant space reduction is obtained by applying AVD. AVD is thereby an obvious choice within the distance management. However applying only AVD, the bitmaps found within the metadata cube are then not reduced in size. Hence, the total space reduction is more optimal when applying both WAH and AVD. Moreover, it is relevant to consult the average bitmap sizes for the applied indexing. In this context, a difference is expected when considering the bitmaps of the metadata cube and the distance management in isolation. Intuitively, the bitmaps within the metadata cube contains many consecutive 0 bits and are thus subject to high compression whereas the bitmaps of the distance management are more diversified with respect to the occurrence of 0 and 1 bits, causing the compression techniques to become ineffective.

In Figure 6(a) the average bitmap sizes are presented for bitmaps within the *metadata cube* and bitmaps within the distance management, respectively. The distance management is configured to use AVD, for which reason the 12 partitions are represented using only seven bitmaps. In the figures the dashed horizontal lines indicate the threshold size needed to store 50,000 bits. The WAH compression yields a slight space overhead, i.e., no compression is possible. The average bitmap sizes for the *metadata cube* is reduced significantly when applying WAH compression. In this case, nine bytes are used for the average bitmap for 50,000 songs. This, in turn, causes the bars representing the cube



**Fig. 6.** results for storage, query execution and throughput

to become invisible in Figure 6(a). In the case of uncompressed bitmaps the average bitmap size within the cube is much below the threshold size 50,000 bits as the 0 bits in the end of the bitmaps are omitted.

Figure 6(b) shows how the number of partitions used within the distance management influences the space consumption for the four different bitmap representations. The space consumption of uncompressed bitmaps increases linearly to the number of partitions while the other three bitmap representations reach an upper bound. The growth of the WAH compressed type becomes minimal around 62 partitions. This is explained by the fact that on a 32-bit architecture, 31 consecutive bits have to be identical in order to constitute a compressible word. The WAH compression becomes effective as soon as two consecutive compressible words are found, that is as soon as chains of 62 identical consecutive bits are present. The WAH compression occupies less space in practice than compared to the theoretical worst-case calculations. The remaining two bitmap representations are very close; this indicates that WAH compression on an AVD represented distance store does not gain a notable reduction. In fact, for 6 and 12 partitions an insignificant reduction is notable whereas for 20 partitions and above an important overhead is introduced.

## 6.2 Query Performance and Throughput

We conduct the query performance experiments on the four different bitmap representations. Unless stated otherwise, all tests assume that a given music collection is restricted

to 75% of the entire collection and that the number of partitions is fixed to 12. Moreover, the number of skipped songs is as default set to 50 songs. Considering the properties of the skipping behaviour of the MOD framework, 50 songs constitutes a rather large collection of skipped songs as all songs resembling any of the skipped songs are restricted from being retrieved. The test results are based on an average of 50 requests. Between each run the cache of the SQL Server is emptied in order to ensure a fair comparison on a cold cache.

In Figures 6(c) and (d), the average query execution time is presented for random and similar songs, respectively. All average query execution times on a collection of 100,000 songs are found to be at most 65ms. In case of solely applying WAH compression we have obtained an average query execution time at 17ms and 22ms for querying random and similar songs, respectively. Comparing the two types of queries the results obtained reflect each other as the number of songs indexed increases, except that all average query execution times for a random song are a little faster than for the corresponding similar song query. The reason for this difference is due to, that a random song is retrieved within a small subset of the entire collection. In average, the bitmap representing this small subset has many omitted 0 bits in the end. Therefore, bit-wise operations perform faster. Moreover, it can be seen that the two WAH compressed representations yield faster query evaluation compared to the uncompressed representations. The reason for this is explainable by the reduced size of the bitmaps when searching for a candidate song in the border partitions of a distance store. The border partitions constitute the partitions representing the most similar and the least similar songs.

As can be seen from the figure the results are nearly linear, which reflects the expected linearity of appending skipped songs to the composite skip distance store. Independent on the chosen bitmap representation, less than 150ms is required to generate the composite skip distance store when none or a single song is skipped. When skipping 100 songs for each bitmap representation, we initially see that the WAH representation takes as long as 1.7s to construct the composite skip distance store. For the same amount of skipped songs, the two AVD representations perform faster compared to the two non-AVD representations. As we consider generation of the composite skip distance store, distance stores for all the skipped songs should be retrieved from the database. Using an AVD representation of the distance stores, fewer records should be fetched, which explains the improved query performance. However, applying both AVD and WAH compression an additional reduction is achieved. The reason for this is that the length of the bitmaps representing the music metadata is reduced, whereby less data is the be retrieved from the database.

Next, we conduct a throughput test to examine how many requests the MOD framework is able to handle over time, when a different number of songs are indexed. To conduct the tests we create multiple *request threads*, which simulates music players, including history management, restriction and handling of skipped songs. The request threads perform both random and similar requests, switching between performing 20 random requests and 20 requests of similar songs for a single seed song. The tests are conducted by instantiating 50 threads, where one half starts by requesting similar songs, and the other half random songs.

In Figure 6(e) and (f) the results obtained by execution of the throughput test are presented. The graphs indicates that, for all the test setups, no requests are served in the beginning of the conducted tests. The reason for this behavior is that the composite skip distance stores are generated during the first requests. In addition, some time elapses until the number of requests served stabilize. This is caused by the different query execution times related to the retrieval of random and similar songs. Figure 6(e) presents the results when applying neither AVD nor WAH. In this case we are able to serve around 400, 200 and 100 requests per second for indexing 10,000, 50,000 and 100,000 songs, respectively. When applying both AVD and WAH we have obtained the results presented in Figure 6(f). As expected from the previous results obtained, the performance decreases when the number of indexed songs increases. With respect to 10,000 song we see no notable increase in the number of requests served by the MOD framework. However, for 50,000 and 100,000 songs we are able to serve around 300 and 200 requests per second, respectively. Assuming an average request frequency for each listener, the number of requests per second can be turned into a the number of users that can be served simultaneously. With an average duration of three minutes per song, the average request frequency of a listener can be set to once every three minutes. Hence, converted into seconds, the frequency is  $5.56 \cdot 10^{-3}$  requests per second. Thereby, serving 200 requests per second on a database containing 100,000 songs, we are able to serve approximately 36,000 simultaneous listeners.

Finally, we now compare our framework to a “baseline” version using B-tree indexes for different size of the music collection. We chose a B-tree since other traditional indexes for high-dimensional data cannot be applied as the triangular inequality cannot be assumed to hold. The B-tree version has been indexed such as to achieve the optimal conditions for joining the tables described above.

We compare the performance of queries based on *metadata only*, i.e., without considering the similarity metric. To compare the two versions we execute 50 randomly generated restrictions in order the retrieve the filenames of the audio files associated with these restrictions. An example of such a restriction could resemble “all songs from the 70’s that are of the genre Rock”. The performed restrictions returns approximately 0.5% of the songs contained in the respective music collections. The outcome of this test is presented in Figure 6(g). For a given music collection containing 100,000 songs the B-tree version takes 709ms on average whereas the MOD framework using bitmap indexes used only 101ms. The bitmap indexes reduces the query time by a factor 7 while the space consumptions for the two approaches are *very similar*. If we should consider the similarities as well, the B-trees would have to index the distance stores for individual songs, each consisting of many songs, that must then be merged during query processing. B-trees are known not be an efficient way (neither time- nor space-wise) of doing this. We also know from the previous experiments that the time for handling both metadata and similarity in the bitmap version is not even twice of that for handling metadata alone. Since the B-tree version is much slower, even in the case where it has the best odds (metadata only), and the bitmap version can handle both metadata and similarity faster than the B-tree version can handle metadata alone, we can conclude that our bitmap approach is quite superior to using B-trees.

## 7 Conclusion and Future Work

Motivated by the proliferation of recommender systems performing similarity search in possibly non-metric similarity spaces, this paper proposes an innovative approach to flexible, yet effective, indexing for exactly such spaces. To illustrate this approach, the domain of music recommendation was chosen. Using a non-metric similarity measure, we are able to retrieve songs similar to a given seed song and avoid retrieval of songs similar to any disliked songs.

To facilitate musical similarity search, we introduced the *distance store* as an abstraction over an arbitrary similarity measure. The distance store constitutes a complete partitioning, where each partition represents a grouping of songs considered similar to a given base song. Applying bitmap indexes to represent each grouping, we are able to identify and retrieve songs similar/dissimilar to a given base song using bit-wise operations on the bitmaps associated with the individual groupings. Furthermore, in order to ensure efficient retrieval of songs based on metadata, we have constructed a metadata cube to which we applied bitmap indexing techniques. This multidimensional cube is mapped to a snowflake schema in an RDBMS, thus allowing a hierarchical representation of the music metadata.

We have thus demonstrated that bitmaps can be used to represent both metadata and non-metric distance measures. Using a single index method for the different music information, the MOD framework remains simple and highly flexible. Moreover, we have described how the framework applies bitmap compression using the Word-Aligned Hybrid compression scheme and the Attribute Value Decomposition technique. Experiments showed that the approach scaled well, both in terms of query performance, throughput, and storage requirements.

As future work, the MOD framework will be compared to other existing frameworks and indexes using various similarity measures, e.g., CompositeMap [4]. We will also address the use of bitmap operations. In case that numerous bitmaps are to be combined using regular bit-wise operations, *lazy* implementations of the Word-Aligned Hybrid compressed bitmap operations could increase the overall performance of the algorithms. Hence instead of consulting the bitmaps in a pairwise fashion, only to obtain a number of intermediate results, which again are to be consulted, the bitmaps could be stored in a special structure delaying the consultation until the result is required.

**Acknowledgments.** This work was supported by the Intelligent Sound project, founded by the Danish Research Council for Technology and Production Sciences under grant no. 26-04-0092.

## References

- [1] Silberschatz, A., Korth, H., Sudershan, S.: Database System Concepts, 4th edn. McGraw-Hill, New York (2005)
- [2] Aucouturier, J., Pachet, F.: Music Similarity Measures: What's the Use? In: Proc. of ISMIR, pp. 157–163 (2002)
- [3] Aucouturier, J.-J., Pachet, F.: Improving Timbre Similarity: How high's the sky? Journal of Negative Results in Speech and Audio Sciences 1(1) (2004)

- [4] Zhang, Q.X.B., Shen, J., Wang, Y.: CompositeMap: a Novel Framework for Music Similarity Measure. In: Proc. of SIGIR, pp. 403–410 (1999)
- [5] Chan, C.Y., Ioannidis, Y.E.: Bitmap Index Design and Evaluation. In: Proc. of SIGMOD, pp. 355–366 (1998)
- [6] Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: Proc. of VLDB, pp. 426–435 (1997)
- [7] Digout, C., Nascimento, M.A.: High-Dimensional Similarity Searches Using A Metric Pseudo-Grid. In: Proc of ICDEW, pp. 1174–1183 (2005)
- [8] Jensen, C.A., Mungure, E., Pedersen, T.B., Sørensen, K.: A Data and Query Model for Dynamic Playlist Generation. In: Proc. of MDDM (2007)
- [9] Kimball, R., Reeves, L., Thornthwaite, W., Ross, M., Thornwaite, W.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses. Wiley, Chichester (1998)
- [10] Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd edn. Wiley, Chichester (2002)
- [11] Logan, B., Salomon, A.: A Music Similarity Function based on Signal Analysis. In: Proc. of ICME, pp. 745–748 (2001)
- [12] Lübbers, D.: SoniXplorer: Combining Visualization and Auralization for Content-Based Exploration of Music Collections. In: Proc. of ISMIR, pp. 590–593 (2005)
- [13] Mandel, M., Ellis, D.: Song-Level Features and Support Vector Machines for Music Classification. In: Proc. of ISMIR, pp. 594–599 (2005)
- [14] Neumayer, R., Dittenbach, M., Rauber, A.: PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In: Proc. of ISMIR, pp. 618–623 (2005)
- [15] O’Neil, P., Graefe, G.: Multi-table Joins Through Bitmapped Join Indices. ACM SIGMOD Record 24(3), 8–11 (1995)
- [16] O’Neil, P., Quass, D.: Improved Query Performance with Variant Indexes. In: Proc. of SIGMOD, pp. 38–49 (1997)
- [17] Pampalk, E.: Speeding up Music Similarity. In: Proc. of MIREX (2005)
- [18] Pampalk, E., Flexer, A., Widmer, G.: Improvements of Audio-Based Music Similarity and Genre Classification. In: Proc. of ISMIR, pp. 628–633 (2005)
- [19] Pampalk, E., Pohle, T., Widmer, G.: Dynamic Playlist Generation Based on Skipping Behavior. In: Proc. of ISMIR, pp. 634–637 (2005)
- [20] Pedersen, T.B., Jensen, C.S.: Multidimensional Database Technology. IEEE Computer 34(12), 40–46 (2001)
- [21] Pohle, T., Pampalk, E., Widmer, G.: Generating Similarity-based Playlists Using Traveling Salesman Algorithms. In: Proc. of DAFx, pp. 220–225 (2005)
- [22] Stockinger, K., Düllmann, D., Hoschek, W., Schikuta, E.: Improving the Performance of High-Energy Physics Analysis through Bitmap Indices. In: Ibrahim, M., Küng, J., Revell, N. (eds.) DEXA 2000. LNCS, vol. 1873, pp. 835–845. Springer, Heidelberg (2000)
- [23] Thomsen, E.: OLAP Solutions: Building Multidimensional Information Systems. Wiley, Chichester (1997)
- [24] Wu, K., Otoo, E.J., Shoshani, A.: Optimizing Bitmap Indices With Efficient Compression. ACM TODS 31(1), 1–38 (2006)

# Efficient Incremental Near Duplicate Detection Based on Locality Sensitive Hashing

Marco Fisichella, Fan Deng, and Wolfgang Nejdl

Forschungszentrum L3S, Hannover 30167, Germany  
{fisichella,deng,nejdl}@L3S.de

**Abstract.** In this paper, we study the problem of detecting near duplicates for high dimensional data points in an incremental manner. For example, for an image sharing website, it would be a desirable feature if near-duplicates can be detected whenever a user uploads a new image into the website so that the user can take some action such as stopping the upload or reporting an illegal copy. Specifically, whenever a new point arrives, our goal is to find all points within an existing point set that are close to the new point based on a given distance function and a distance threshold before the new point is inserted into the data set. Based on a well-known indexing technique, Locality Sensitive Hashing, we propose a new approach which clearly speeds up the running time of LSH indexing while using only a small amount of extra space. The idea is to store a small fraction of near duplicate pairs within the existing point set which are found when they are inserted into the data set, and use them to prune LSH candidate sets for the newly arrived point. Extensive experiments based on three real-world data sets show that our method consistently outperforms the original LSH approach: to reach the same query response time, our method needs significantly less memory than the original LSH approach. Meanwhile, the LSH theoretical guarantee on the quality of the search result is preserved by our approach. Furthermore, it is easy to implement our approach based on LSH.

## 1 Introduction

Similarity search is an important research topic which finds applications in different areas. For example, finding all similar images of a query image in a large image collection based on certain similarity measures and thresholds. Feature vectors can be extracted from the images. Once this is done, the set of images can be considered as a set of high dimensional points. In general, similarity search can refer to a variety of related problems. In this paper, the problem we consider is to answer range search queries in an incremental manner. That is, whenever a new point arrives, find all similar/close points (based on a pre-specified similarity threshold) from the set of high dimensional points arrived earlier, and then insert the new point into the data set.

The motivating application of this work is online near duplicate detection for multimedia content sharing websites like Flickr<sup>1</sup> and Youtube<sup>2</sup>. Whenever a user is uploading an image or video, it would be desirable if near-duplicates that are very similar (content-wise) to the one being uploaded can be retrieved and returned to the user in real-time. In this way, the user can identify redundant copies of the object promptly and decide if he/she should continue the upload. In addition to personal users, enterprise users may also need this type of applications. For example, media companies such as broadcasters and newspapers may continuously upload their images or videos to a multimedia content repository. The copy right issue is one of their main concerns. It would be a useful feature if near-duplicate copies can be retrieved and reported to the users during the upload period so that the user can identify pirated copies promptly. If the new object is illegal, the user should immediately stop the upload process.

Compared to the traditional similarity search problem, fast response is more important for this type of applications since similarity search is only part of the online content upload process which must be completed within at most a few seconds. In addition to the online requirement, another characteristic of the motivating applications is that the similarity search operations is executed together with data point insertions. In other words, the data set is created incrementally where the near neighbors of each point are known before the point is inserted into the data set.

To speed up the searching process, in-memory indexing techniques are ideal solutions if the help of disk-based index are not necessary since a disk access is an order of magnitude slower than a memory operation. For a data set with 1 million points, an index storing all the point IDs once only needs 12MB memory assuming each ID takes 12 bytes; if each point is a 162-dimensional point and each dimension of a point takes 4 bytes, storing all the points needs 648MB, which is tolerable even for an inexpensive PC nowadays. Although processing Web-scale data set with billions of points may need clusters with tens or hundreds of distributed machines, indexing an enterprise-scale data set with tens or hundreds of millions points in main-memory is feasible using a single server with a larger memory size. Unfortunately, to give a fast query response, the index size needed for high-dimensional points is usually larger than the size we computed, and it can be even larger than the data set size. Thus, in this work we focus on reducing memory consumption of in-memory index while providing fast query response.

Although decades of research have been conducted on similarity search, the problem is still considered challenging. One important reason is the “curse of dimensionality”. It has been shown that exponential space in  $n$  (number of points in the data set) is needed to speed up the similarity search process or the searching time increases exponentially with the dimensionality [24]. It is also shown both theoretically and empirically [24] that all partitioning and clustering based indexing approaches degrade to a brute force linear scan approach when the dimensionality is sufficiently high.

---

<sup>1</sup> <http://www.flickr.com>

<sup>2</sup> <http://www.youtube.com>



To our knowledge, a state-of-the-art solution to the similarity search problem in practice, which provides fast query response time, is Locality Sensitive Hashing (LSH) [12,15] although it has been proposed for a decade. Meanwhile, LSH also provides theoretical guarantees on the quality of the solution. However, also suffering from the “curse of dimensionality”, LSH needs large amount of space to achieve fast query response.

## 1.1 Our Contributions

- We proposed a novel approach, SimPair LSH, to speed up the original LSH method; the main idea is to take advantage of a certain number of existing similar point pairs, and use them to prune LSH candidate sets relevant for a given query.
- The correctness and effectiveness of the new approach is analyzed.
- Thorough experiments conducted on 3 real-world data sets show that our method consistently outperforms LSH in terms of query time in all cases that we tried, with a small amount of extra memory cost. To achieve the same query time saving, we show that LSH need significantly more space. Meanwhile, we show that our method preserves the important theoretical guarantee on the recall of query answers.

## 2 Preliminaries and Related Work

### 2.1 Problem Statement (Incremental Range Search)

In this paper, we focus on the incremental range search problem defined as follows: given a point  $q$  and a set  $P$  with  $n$   $d$ -dimensional points, efficiently find out all points in  $P$  that are similar to  $q$  based on certain similarity/distance function and a similarity threshold  $\tau$  before  $q$  is inserted into the data set. We call the points similar to  $q$  *near neighbors* of  $q$ . In this problem, before evaluating the query  $q$ , the near neighbors of all points within the data set are retrieved when they are inserted into the data set.

**Distance measure.** We focus on Euclidean distance since it has been widely used in different applications. It is not hard to extend the technique to other distance functions such as L1 and Hamming distance, as the underlying technique, Locality Sensitive Hashing, can be applied in those cases.

**In-memory index structure.** We focus on in-memory index structure since fast real-time response is the first priority in the applications we consider. For high dimensional similarity search, the index size can be as large as or even larger than the data set size in order to give an efficient query response time. Therefore, reducing the memory cost while providing fast response is the main concern of this work.

### 2.2 Straightforward Solution

A straightforward solution to this problem is LinearScan: compute the distance between  $q$  and each point  $p$  in  $P$ ; if the similarity is above the given similarity

threshold, output this point. It is not hard to see that this approach can be very slow for large data sets, especially when the dimensionality  $d$  is large; in the case of Euclidean distance, LinearScan takes  $O(nd)$  time for each query.

### 2.3 Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing (LSH) [15,12] was proposed by Indyk and Motwani and finds applications in different areas including multimedia near duplicate detection (e.g. [9], [11], [17]). LSH was first applied in indexing high-dimensional points for Hamming distance [12], and later extended to  $L_p$  distance [10] where  $L_2$  is Euclidean distance, which we will use in this paper.

The basic idea of LSH is to use certain hash functions to map each multi-dimensional point into a scalar; the hash functions used have the property that similar points have higher probability to be mapped together than dissimilar points. When LSH is used for indexing a set of points to speed up similarity search, the procedure is as follows: first, create an index (a hash table) by hashing all points in the data set  $P$  into different buckets based on their hash values; select  $L$  hash functions uniformly at random from a LSH hash function family and create  $L$  hash tables; when the query point  $q$  arrives, use the same set of hash functions to map  $q$  into  $L$  buckets, one from each hash table; retrieve all points from the  $L$  buckets into a candidate set  $C$  and remove duplicate points in  $C$ ; for each point in  $C$  compute its distance to  $q$  and output those points similar to  $q$ .

An essential part of LSH is the hash function family  $H$ . For Euclidean distance, the hash function family can be constructed as follows [10]: map a multi-dimensional point  $p$  into a scalar by using the function  $h(p) = \lfloor \frac{a \cdot p + b}{r} \rfloor$  where  $a$  is a random vector whose coordinates are picked uniformly at random from a normal distribution, and  $b$  is a random variable uniformly distributed in the range  $[0, r]$ . In this hash function, the dot product  $a \cdot p$  is projecting each multi-dimensional point  $p$  into a random line; the line is cut into multiple intervals with length  $r$ ; the hash value shows which interval  $p$  is mapped to after a random shift of length  $b$ . Intuitively, it is clear that closer points have higher chance being mapped into the same interval than distant points under this random projection. Last, generate a new hash function  $g(p)$  to be used in constructing a hash table by concatenating  $k$   $h_i(p)$  ( $i = 1 \dots k$ ), each chosen uniformly at random from  $H$ , i.e.  $g(p) = (h_1(p), \dots, h_k(p))$ .

The nice property of the LSH is that the probability that two points  $p_1$  and  $p_2$  are hashed into the same bucket is proportional to their distance  $c$ , and this probability can be explicitly computed using the following formulas:

$$p(c) = Pr[h(p_1) = h(p_2)] = \int_0^r \left(\frac{1}{c}\right) f\left(\frac{t}{c}\right) \left(1 - \frac{t}{r}\right) dt, \quad (1)$$

where  $f(t)$  is the probability density function of the absolute value of the normal distribution. Having  $p(c)$ , we can further compute the collision probability under  $H$ :

$$P(c) = Pr[H(p_1) = H(p_2)] = 1 - (1 - p(c))^k{}^L. \quad (2)$$

## 2.4 Other LSH-Based Approaches

Since proposed, LSH has been extended in different directions. Lv et al. [20] proposed multi-probe LSH, and showed experimentally that their method significantly reduced space cost while achieving the same search quality and similar time efficiency compared with original LSH. The key idea of multi-probe LSH is that the algorithm not only searches for the near neighbors in the buckets to which the query point  $q$  is hashed, it also searches the buckets where the near neighbors have slightly less chance to appear. The benefit of multi-probe LSH is that each hash table can be better utilized since more than one bucket of a hash table is checked, which decreases the number of hash tables. However, multi-probe LSH does not provide the important search quality guarantee as LSH does. The original LSH scheme guarantees that the true results will be returned by the search algorithm with high probability, while multi-probe could not. This makes multi-probe LSH not applicable in those applications where the quality of the retrieval results are required to be guaranteed. The idea of multi-probe LSH was inspired by earlier work investigating entropy-based LSH [21]. The key idea is to guess which buckets the near neighbors of  $q$  are likely to appear in by randomly generating some “probing” near neighbors and checking their hash values. Similar to multi-probe LSH, entropy-based LSH also reduces the number of hash tables required. In practice, though, it is difficult to generate proper “probing” near neighbors in a data-independent way [20].

Another extension of LSH is LSH forest [5] where multiple hash tables with different parameter settings are constructed such that different queries can be handled with different settings. In the theory community, a near-optimal LSH [2] has been proposed; however, currently it is mostly of theoretical interest because the asymptotic running time improvement is achieved only for a very large number of input points [4]. More LSH related work can be found in a recent survey [4]. This survey also observes, that despite decades of research, current solutions still suffer from the “the curse of dimensionality”. In fact, for a large enough dimensionality, current solutions provide little improvement over LinearScan, both in theory and in practice [4]. We further note that the technique in this paper is orthogonal to the other LSH variants described above and can be applied in those scenarios.

## 2.5 Tree-Based Indexing Techniques

When the dimensionality is relatively low (e.g. 10 or 20), tree-based indexing techniques are known to be efficient. Examples include kd-trees [6], R-tree [14], SR-tree [16], cover-trees [8] and navigating-nets [19]. These methods do not scale well with the (intrinsic) dimensionality. Weber et al. [24] show that when the dimensionality exceeds 10, all space partitioning and clustering based indexing techniques degrade to LinearScan. For indexing high dimensional points, B+ tree is also used together with different techniques handling the “dimensionality curse”, such as iDistance [25] and LDC [18]. Other tree-based approaches like IQ-tree [7] and A-tree [22] use a smaller vector to represent the data points

approximately which helps to reduce the complexity of the problem. Different from the LSH based approaches where large amount of space is traded for gaining fast response time, the tree-based approaches have less concern on index space while they usually have faster but comparable query time as LinearScan.

Due to the intensive research within the past decades, there are a large body of related literature which cannot be covered here. Samet's book [23] provides a comprehensive survey on this topic.

### 3 SimPair LSH

Our approach is based on the standard LSH indexing, and takes advantage of existing similar pair information to accelerate the running time of LSH. We thus call it *SimPair LSH*. Unless noted otherwise, LSH denotes the original LSH indexing method in the rest of this paper.

#### 3.1 Key Idea

We observe that LSH retrieves all points stored in the buckets a query point  $q$  hashed to. Let the set of points returned by LSH be the candidate set  $C$ . Then  $q$  is compared with all the points in  $C$  as in LinearScan, and the near neighbors are found. To guarantee a low chance of missing a near neighbor in  $C$ , a large number of hash tables has to be created which may lead to a large  $C$  depending on the query  $q$ , and accordingly increase the running time especially when  $d$  is large.

The main idea of this paper is to take advantage of a certain number of pair-wise similar points in the data set and store them in memory; in the process of scanning through  $C$ , the search algorithm can look up the similar pair list on-the-fly whenever a distance computation between  $q$  and a point  $p$  in  $C$  is done; if a similar pair  $(p, p')$  is found in the list, it is very likely that  $p'$  will also appear in  $C$ ; based on the known distances  $d(q, p)$  and  $d(p, p')$  we can infer an upper bound to  $d(q, p')$  by using triangle inequality and may skip the distance computation between  $q$  and  $p'$ . The reason this idea works is that LSH tends to group similar objects into the candidate set  $C$ . Thus the points in  $C$  are very likely to be similar to each other. Checking one point  $p$  can avoid computing distance for the points similar to  $p$ , and therefore saving distance computations.

#### 3.2 The SimPair LSH Algorithm

Our SimPair LSH algorithm works as follows: given a set of points  $P$  and all point pairs (including their distances) whose pair-wise distances are smaller than a threshold  $\theta$  (let the set of all similar pairs be  $SP$ ). Also given the distance threshold  $\tau$  determining near neighbors, SimPair LSH then creates  $L$  indices as in LSH; whenever a query point  $q$  comes, SimPair LSH retrieves all points in the buckets to which  $q$  is hashed. Let this set of points be the candidate set  $C$ . Instead of scanning through all the points  $p$  in  $C$  one by one and compute their

distances to  $q$  as in LSH, SimPair LSH checks the pre-computed similar pair set  $SP$  whenever a distance computation  $d(q, p)$  is done. Based on the distance between  $p$  and  $q$ , SimPair LSH continues in 2 different ways:

- If  $d(q, p) \leq \tau$ , SimPair LSH searches in  $SP$  for all points  $p'$  which satisfies  $d(p, p') \leq \tau - d(q, p)$ ; check if  $p'$  in the candidate set  $C$  or not; if yes, then mark  $p'$  as a near neighbor of  $q$  without the distance computation.
- If  $d(q, p) > \tau$ , SimPair LSH searches in  $SP$  for all those points  $p'$  which satisfies  $d(p, p') < d(q, p) - \tau$ ; check if  $p'$  in the candidate set  $C$  or not; if yes, then remove  $p'$  from  $C$  without the distance computation.

The detailed description is shown in Algorithm [11](#)

---

### Algorithm 1. SimPair LSH

---

**Input:** A set  $P$  with  $n$   $d$ -dimensional points;  $L$  in-memory hash tables created by LSH; a set  $SP$  storing all similar pairs in  $P$  whose pair-wise distances are smaller than  $\theta$ ; a distance threshold  $\tau$  defining near neighbors; and a query point  $q$

**Output:** all near neighbors of  $q$  in  $P$

**begin**

check the  $L$  buckets  $q$  hashed to and retrieve all the points in those buckets as in LSH;

put all the points into a candidate set  $C$ ;

**for** each point  $p$  in  $C$  **do**

compute the distance between  $q$  and  $p$ , i.e.  $d(q, p)$ ;

**if**  $d(q, p) < \tau$  **then**

output  $p$  as a near neighbor of  $q$ ;

search in  $SP$  for all the points  $p'$  which satisfies  $d(p, p') < \tau - d(q, p)$ ;

**for** each point  $p'$  found in  $SP$  **do**

check if  $p'$  in  $C$  or not;

**if** found **then**

└ output  $p'$  as a near neighbor of  $q$  and remove it from  $C$ ;

**if**  $d(q, p) > \tau$  **then**

search in  $SP$  for all the points  $p'$  which satisfies  $d(p, p') < d(q, p) - \tau$ ;

**for** each point  $p'$  found in  $SP$  **do**

check if  $p'$  in  $C$  or not;

**if** found **then**

└ remove  $p'$  from  $C$ ;

**end**

---

The algorithm constructing the LSH indices is the original LSH algorithm. [10](#) describes how to select  $L$  and  $g_i$  to guarantee the success probability.

### 3.3 Algorithm Correctness

Since our algorithm is based on LSH, it is important that the theoretical guarantee still holds for SimPair LSH.

**Theorem 1.** *SimPair LSH has the same theoretical guarantee as LSH has in terms of the range search problem we study. That is, near neighbors will be returned by SimPair LSH with a user-specified probability by adjusting the parameters (hash functions and number of hash tables) accordingly.*

*Proof.* Since we consider points in metric space where triangle inequality holds, SimPair LSH guarantees that the points skipped are either true near neighbors or not near neighbors without distance computation.

### 3.4 Algorithm Effectiveness

The benefit of SimPair LSH compared with LSH is that points in the candidate set returned by LSH can be pruned by checking the similar pair list  $SP$  without distance computations. Therefore, it is important to analyze the number of prunes SimPair generates. Also, to obtain the benefit, SimPair LSH has to search in  $SP$  and  $C$  for the points to be pruned, which can take time although hash indices can be built to speed up each search operation to  $O(1)$  time. Next, we analyze the factors affecting the gain and cost.

**Pruning analysis.** To generate a prune from a point  $p$  in  $C$ , SimPair first has to find a “close enough” point  $p'$  of  $p$  from  $SP$ , where close enough or not depends on  $|d(q, p) - \tau|$ . If  $|d(q, p) - \tau|$  is large, SimPair LSH has a higher chance to find a  $p'$ .

Another factor that can affect the chance of finding  $p'$  from  $SP$  is the size of  $SP$ . Clearly, maintaining a large set of  $SP$  will increase the chance of finding  $p'$  of  $p$ .

Finding  $p'$  of  $p$  does not necessarily lead to a prune. The condition that a prune occurs is that  $p'$  appears in  $C$ . According to the property of LSH hash functions, points close to  $q$  have higher chance appearing in  $C$ . In other words,  $d(q, p')$  determines the chance of generating a prune. Although  $d(q, p')$  can not be known precisely, a bound of this distance can be derived from  $d(q, p)$  and the “close enough” threshold  $|d(q, p) - \tau|$ .

**Cost analysis.** To gain the pruning, SimPair LSH has to pay certain amount of costs including time and space costs. The time cost mainly comes from the searching processes: find the points “close enough” to  $p$  in  $SP$  and check those points to see if they are in  $C$  or not. By constructing hash indices for  $SP$ , searching for  $p$  in  $SP$  only takes  $O(1)$  time; constructing hash indices for  $SP$  also takes  $O(1)$  time for each object. When a candidate set  $C$  of points for the query  $q$  is retrieved, all points in the dataset belonging to  $C$  are marked both in LSH and SimPair LSH; this is possible since each point in the data set has a Boolean attribute showing if the point is in  $C$  or not. The purpose of having this attribute is to remove duplicate points when generating  $C$ . Duplicates can appear in  $C$  because one point can appear in multiple LSH hash buckets. Note that when the searching is finished, the boolean attributes need to be cleared (for both LSH and SimPair LSH) which takes  $O(|C|)$  time when all points in  $C$  are also maintained in a linked list. For the sake of pruning, another boolean attribute is needed for each point to indicate if the point has been pruned or not.

With these boolean attributes, searching for  $p'$  in  $C$  takes  $O(1)$  time. The time cost is mainly generated by searching  $p'$  in  $C$  since there can be multiple  $p'$  for each  $p$ , and therefore multiple look-ups in  $C$ .

In addition to the time cost, SimPair LSH also has some extra space cost for storing  $SP$  compared with LSH. This cost is limited by the available memory. In our approach, we always limit the size of  $SP$  based on two constraints: (i) the similarity threshold  $\theta$  (for the similar point pairs stored in  $SP$ ) is restricted to the range  $(0, \tau]$ ; (ii) the size of  $SP$  must not exceed a constant fraction of the index size (e.g. 10%).

## 4 Experiments

In this section, we demonstrate the practical performance of our approach on three real-world data sets, testing the pruning effectiveness, pruning costs, real running time together with memory saving and quality of results from SimPair LSH.

### 4.1 Data Sets

We use three real-world image data sets in our experiments: one directly downloaded from a public website and two generated by crawling commercial multimedia websites.

**Flickr images.** We sent 26 random queries to Flickr and retrieved all the images within the result set. After removing all the images with less than 150 pixels, we obtained approximately 55,000 images.

**Tiny images.** We downloaded a publicly available data set with 1 million tiny images<sup>3</sup>. The images were collected from online search tools by sending words as queries, and the first 30 returned images for each query are stored. Due to the high memory cost of LSH for large data sets, we picked 50 thousand images uniformly at random from this 1 million tiny image data set. This random selection operation also reduced the chance that similar pairs appear in the data set since the images retrieved from the result set of one query have higher chance to be similar to each other.

The reason we used this smaller data set rather than only considering the full set was that we could vary the number of hash tables within a larger range and observe the behavior of the algorithms under different number of hash tables. For example, the largest number of hash tables we used was about 1000; indexing the 1 million data points takes 12GB memory under this setting which was above the memory limit of our machine. (Note that this is an extreme case for experimental purpose and may not be necessary in practice.) If we used 10 hash tables, then the memory consumption will drop to 120MB. We also conducted experiments on the whole 1 million data set setting the number of hash tables

---

<sup>3</sup> The dataset was collected by A. Torralba and R. Fergus and W. T. Freeman at MIT in 2007; it is available at <http://dspace.mit.edu/handle/1721.1/37291>

to smaller values, so as to see how the algorithms behave with a larger data set size. Due to the space limit, the results from the 1 million data set are reported into the complete version of this paper [1].

**Video key-frames.** We sent 10 random queries to Youtube and obtained around 200 video clips from each result set, approximately 2100 short videos in total. We then extracted all the frames of the videos and their HSV histograms with dimensionality 162. After that, we extracted key frames of the videos in the following way: sequentially scan the HSV histogram of each frame in a video; if the distance between the current histogram and the previous one in the video is above 0.1, keep this histogram; otherwise skip it. We set the distance threshold to 0.1 because two images with this distance are similar but one can clearly see their difference based on our observation. In the end, we obtained 165,000 key-frame images.

For all the images data set described above, we removed duplicates and converted each data set into a  $d$ -dimensional vector ( $d = 162,512$ ) by using the standard HSV histogram methods [13]. Each entry of the vector represents the percentage of pixels in a given HSV interval.

**Pair-wise distance distribution.** Since the pair-wise distance distribution of data set may affect the result of our experiments, we plotted Figures and found that the 3 data sets had similar curves. Specifically, we cut the distance range into multiple intervals and count the number of points within each interval. Due to the space limit, we put the histograms into the complete version of this paper [1].

## 4.2 Experimental Setup

All experiments were ran on a machine with an Intel T2500 2GHz processor, 2GB memory under OS Fedora 9. The algorithms were all implemented in C compiled by gcc 4.3.0.

The data points and the LSH indices are both loaded into the main memory. Each index entry for a point takes 12 bytes memory. To test the performance of our approach, we randomly selected a certain number of objects from the data set as query objects, and measure metrics as discussed before. We took the average number of pruned points of all queries, the average percentage of pruned points, and the average number of operations spent on achieving the pruning per point (average number of cost operations per query / average size of candidate sets  $C$  of all queries).

## 4.3 Experiments Testing Pruning Effectiveness and Costs

In this set of experiments, we tested the number of distance computations saved by our approach, the time and space cost to obtain the saving. We used the Flickr data set, and the results from other data sets are also consistent in general.



From our experiments we can see that the algorithm performance is not sensitive to the number of queries, and we fixed the number of queries to 100 in the following experiments.

Since candidate set sizes  $|C|$  for some queries is quite small (e.g.  $< 50$ ), and there is no need to start the pruning process, we set a cut-off threshold  $T$  for  $|C|$ . When  $|C| > T$ , SimPlair LSH start the pruning process; otherwise, SimPlair LSH does not start the pruning process and degrades to the original LSH. From our experiments we can see that the algorithm performance is not sensitive to  $T$  in terms of both the percentage of pruned points and the average number of operations cost per point. In the rest of the experiments, we fixed  $T$  to 200.

Due to the space limit, the results of testing the number of queries and  $T$  are not shown here, but can be found in the complete version of the paper [1].

#### 4.4 Experiments Testing the Query Response Time

In this set of experiments, we report the query response time of the original LSH indexing and our approach. The LSH code were obtained from Andoni [3], and we conducted the experiments for LSH without changing the original source code.

**Hash function time costs.** Note that during query time, generating the hash values of each query also takes time where the amount depends on the number of hash functions or hash tables used. In the case that the candidate set size is relatively small and the number of hash functions is large, the hashing process can take as high as the time spent on scanning the candidate set. Since the time spent on generating the hash values are exactly the same for both SimPair LSH and the original LSH, and the percentage of this portion varies significantly with the parameter setting of  $L$  and the size of  $C$  which depends on the queries, we only report the time spent on finding near neighbors from  $C$  to see the difference between our approach and LSH better. When the size of  $C$  is relatively large, the fraction of hash function time cost is relatively small. But in the worst case where hashing queries take the same amount of time as scanning  $C$ , the time difference between two approach will be half of the numbers reported below.

**Varying  $k$  and  $L$ .** We varied the hash function parameter  $k$  and the number of hash tables  $L$  to see how these parameters affect the query time. The threshold  $\tau$  was set to 0.1, success probability  $P$  was set to 95%. Note that once  $k$  were fixed, the number of hash tables  $L$  was also fixed to guarantee the required success probability. Other parameters are the same as in the previous experiments. The results on the 3 data sets are shown in Figure 1a. Y-axis is the response time saved by SimPair LSH computed as follows:  $(\text{LSH Time} - \text{SimPair LSH Time}) / (\text{LSH time})$ . From the figure we can see that SimPair LSH consistently outperforms LSH under different settings of  $K$  and  $L$ . The extra memory consumptions of the full similar pair set  $SP$  when  $\theta = 0.1$  are  $73.7MB$ ,  $12.5MB$  and  $3.7MB$  respectively for the video key frame, Tiny images and Flickr image data sets. Recall that  $SP$  size is bound to the 10% of the index size, thus when  $L$

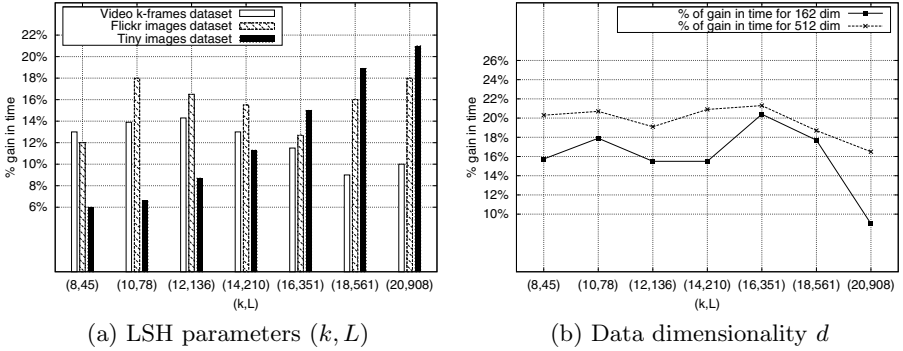


Fig. 1. LSH parameters ( $k, L$ ) and data dimensionality vs. running time

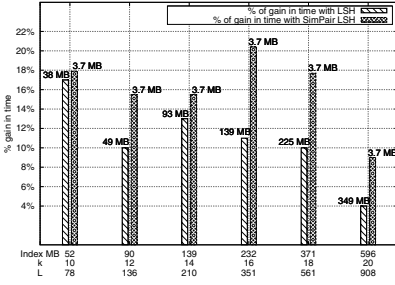
is small, we use a smaller  $\theta$ . Since LSH can also save running time by increasing the number of hash tables (varying  $K$  and  $L$ ), we tested how much additional memory LSH needs to gain the same amount of time in the next experiments.

### Extra space cost comparison showing the significance of our time gain.

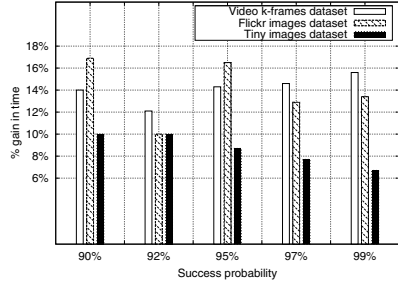
To achieve the response time gain, in addition to the memory cost of the LSH indices, the extra cost SimPair LSH takes is the memory spent on the similar pair list that we restricted at most to a constant fraction of the LSH indices (10% in our experiments). To achieve approximately the same running time gain, LSH can also increase the memory consumption by increasing  $k$  and  $L$  without resorting to our approach. Therefore, we compared the memory consumption of the two approaches to achieve roughly the same query time improvement. In fact, the memory cost of LSH can be computed from  $L$ : each hash table stores the identifiers of all  $n$  points, and each identifier takes 12 bytes as implemented by Andoni and Indyk [3]; therefore, the LSH space cost is  $12nL$ .

Hence, to see the size of extra space LSH needs, it suffices to check the value of  $L$ . Since the size of  $C$  dominates the time LSH scans through the candidate set, the running time being saved can be represented by the reduction of  $|C|$ . Recall that bigger values of  $L$  correspond to the decrease in size of  $C$ .

The Figure 2 is based on the Flickr image data set. The x-axis presents the memory consumption of hash tables needed to index all the points in the dataset for diverse settings of  $k$  and  $L$ ; such a space cost is computed from formula  $12nL$  as discussed above, where  $n = 55,000$ . Since SimPair LSH uses the indices of LSH, this memory utilization is common for both algorithms. The y-axis shows the percentage gain in time. The numbers on top of the bars show the extra memory cost required to reach the gain in time reported. It is important to note that the extra memory cost for SimPair LSH remains constant to  $3.7MB$ , the similar pair list size, for all the diverse setting of  $L$ , while LSH needs significantly more extra memory, with increasing  $L$ , to achieve similar response time as our method. We can conclude that LSH needs significantly more memory to achieve even less response time saving as SimPair LSH does. For example, to have a gain in running time of 17% when  $L = 78$ , LSH needs  $38MB$  extra memory.



**Fig. 2.** Running time saved vs. LSH memory consumption



**Fig. 3.** Success probability  $P$  vs. running time

In contrast, SimPair LSH only needs  $3.7MB$  storing the similar pair set to gain more than 17% response time; that makes our algorithm save 10 times the space cost used from LSH. Note that when the number of hash tables is increased, the time spent on computing the  $L$  hash values will also increase proportionally, which means the real running time saving is actually smaller than 17% for LSH. When  $L$  is large, even more extra memory is needed to gain the same amount of running time. For example, when the hash table size increased by around  $225MB$ , the time cost decreases only about 10%; in this case our method saves roughly 60 times the space cost used from LSH.

Comparing with the figure shown in the previous experiment, by using the same amount of extra memory ( $3.7MB$ ), SimPair LSH gains slightly more percentage for different settings of  $k$  and  $L$ . Clearly, SimPair LSH is more space efficient in terms of saving running time.

**Varying the success probability  $P$ .** We varied  $P$  from 90% to 99%, and set  $k = 12$  and  $L$  changes accordingly to see how  $P$  affects the real running time. Other parameters are the same as the previous experiment. The results are shown in Figure 3. From the figure again we can see that SimPair LSH outperforms LSH in terms of running time consistently. For different data sets,  $P$  has different impact on the saving time. However, the general trends seem to indicate that the impact is not significant.

**Varying the dimensionality  $d$ .** We ran experiments on Flickr image data set with different dimensionality  $d$ : 162 and 512, and set  $P = 95%$  to see how  $d$  affects real running time saved. Other parameters are the same as the previous experiment. The results are shown in Figure 4. The y-axis shows the percentage of running time saved as in the previous experiments. From this figure we can see that for a higher dimensionality, the percentage of real time saved is higher in general. This is because the gain in time each prune brings is relatively higher compared with the cost of each prune when the dimensionality is higher.

In addition, we also ran experiments on the full 1 million tiny image data set as mentioned earlier, and the results were consist with what we have shown. Due to the space limit, please see details in the complete version of this paper [1].

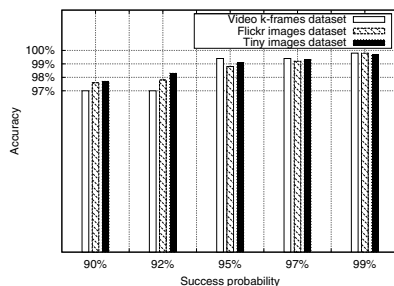


Fig. 4. Success probability  $\theta$  vs. Recall (Quality of results)

## 4.5 Quality of Results

In this set of experiments, we tested the recall or false negatives of both methods. If a user set  $P$  to 90%, it means that he/she can tolerate missing at most 10% near neighbors. The results in Fig. 4 shows that the real recall value is clearly higher than the user specified probability.

## 5 Conclusions

In this paper, we study the problem of range search in an incremental manner based on a well-known technique, Locality Sensitive Hashing. We propose a new approach to improve the running time of LSH. The idea is take advantage of certain number of existing similar point pairs, and checking this similar pair set on-the-fly during query time. Since the look-up time cost is much cheaper than the distance computation, especially when the dimensionality is high, our SimPair LSH approach consistently outperforms the original LSH method, with the cost of a small amount of extra space. To gain the same amount of running time, LSH needs significantly more space than SimPair LSH (e.g. 10 to 100 times more). The superiority of SimPair LSH over the original LSH is confirmed by our thorough experiments conducted on 3 real-world image data sets. Furthermore, SimPair LSH preserves the theoretical guarantee on the recall of the search results. Last, SimPair LSH is easy to implement based on LSH.

## References

1. Complete version of this paper can be found at, [https://www.l3s.de/web/upload/documents/1/SimSearch\\_complete.pdf](https://www.l3s.de/web/upload/documents/1/SimSearch_complete.pdf)
2. Andoni, A., Indyk, P., Patrascu, M.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: FOCS, pp. 459–468 (2006)
3. Andoni, A., Indyk, P.:  $E^2$  LSH0.1 User Manual. <http://web.mit.edu/andoni/www/LSH/manual.pdf> (2005)
4. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. CACM 51(1) (2008)

5. Bawa, M., Condie, T., Ganesan, P.: Lsh forest: self-tuning indexes for similarity search. In: WWW, pp. 651–660 (2005)
6. Bentley, J.L.: Multidimensional binary search trees used for associative searching. CACM 18(9) (1975)
7. Berchtold, S., Böhm, C., Jagadish, H.V., Kriegel, H.-P., Sander, J.: Independent quantization: An index compression technique for high-dimensional data spaces. In: ICDE, pp. 577–588 (2000)
8. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: ICML, pp. 97–104 (2006)
9. Chum, O., Philbin, J., Isard, M., Zisserman, A.: Scalable near identical image and shot detection. In: CIVR, pp. 549–556 (2007)
10. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: SCG, pp. 253–262 (2004)
11. Foo, J.J., Sinha, R., Zobel, J.: Discovery of image versions in large collections. In: Cham, T.-J., Cai, J., Dorai, C., Rajan, D., Chua, T.-S., Chia, L.-T. (eds.) MMM 2007. LNCS, vol. 4352, pp. 433–442. Springer, Heidelberg (2006)
12. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB, pp. 518–529 (1999)
13. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice Hall, Englewood Cliffs (2007)
14. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD (1984)
15. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: STOC, pp. 604–613 (1998)
16. Katayama, N., Satoh, S.: The sr-tree: An index structure for high-dimensional nearest neighbor queries. In: SIGMOD (1997)
17. Ke, Y., Sukthankar, R., Huston, L.: An efficient parts-based near-duplicate and sub-image retrieval system. In: ACM Multimedia, pp. 869–876 (2004)
18. Koudas, N., Ooi, B.C., Shen, H.T., Tung, A.K.H.: Ldc: Enabling search by partial distance in a hyper-dimensional space. In: ICDE, pp. 6–17 (2004)
19. Krauthgamer, R., Lee, J.R.: Navigating nets: simple algorithms for proximity search. In: SODA, pp. 798–807 (2004)
20. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search. In: VLDB, pp. 950–961 (2007)
21. Panigrahy, R.: Entropy based nearest neighbor search in high dimensions. In: SODA, pp. 1186–1195 (2006)
22. Sakurai, Y., Yoshikawa, M., Uemura, S., Kojima, H.: The a-tree: An index structure for high-dimensional spaces using relative approximation. In: VLDB, pp. 516–526 (2000)
23. Samet, H.: Foundations of Multidimensional and Metric Data Structures, August 8, 2006. Morgan Kaufmann, San Francisco (2006)
24. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB, pp. 194–205 (1998)
25. Yu, C., Ooi, B.C., Tan, K.-L., Jagadish, H.V.: Indexing the distance: An efficient method to knn processing. In: VLDB (2001)

# Efficient Fuzzy Top- $k$ Query Processing over Uncertain Objects<sup>\*</sup>

Chuanfei Xu, Yanqiu Wang, Shukuan Lin, Yu Gu, and Jianzhong Qiao

Northeastern University, China

{xuchuanfei,wangyanqiu,linshukuan,yuge,qiaojianzhong}@ise.neu.edu.cn

**Abstract.** Recently, many application domains, such as sensor network monitoring and Location-Based Service, raise the issue of uncertain data management. Uncertain objects, a kind of uncertain data, have some uncertain attributes whose values are ranges instead of points. In this paper, we study a new kind of top- $k$  queries, Probabilistic Fuzzy Top- $k$  queries (PF-Top $k$  queries) which can return  $k$  results from uncertain objects for fuzzy query conditions. We formally define the problem of PF-Top $k$  query and present a framework for answering this kind of queries. We propose an exact algorithm, Envelope Planes of Membership Function (EPMF) algorithm based on the upper and lower bounding functions, which answers fuzzy top- $k$  queries over uncertain objects in high-dimensional query space efficiently. We also propose an approximate algorithm which improves efficiency while ensuring high precision by setting a proper value of parameter. To reduce the search space, a pruning method is proposed to safely prune some objects before querying. The effectiveness and efficiency of our algorithms is demonstrated by the theoretical analysis and experiments with synthetic and real datasets.

## 1 Introduction

Recently, uncertain data management has gained dramatic attention due to the inherent uncertainty of data in many application domains, such as sensor network monitoring [1], moving object search [2], medical management [3], and so on. For instance, due to the imperfection of sensing devices, the data obtained are often contaminated with noises [1]. In Location-Based Service systems, it is not always possible to acquire the accurate locations of moving objects because of some measurement error [2] in positioning technologies (e.g., GPS and RFID). In biometric databases, the attribute values (e.g., blood pressure and heartbeat) of the feature vectors stored are also not exact [3] for the purpose of privacy protection. We assume that all uncertain attribute values of an uncertain object locate within a closed region, called the uncertainty region [4]. In this region, a non-zero probability density function (pdf) of the value is defined, where the integration of pdf inside the region is equal to one.

---

<sup>\*</sup> The research was partially supported by NSF of China under Grants Nos. 60873009, 60933001 and 60773220.

Top- $k$  queries over uncertain data play a very significant role in uncertain data management. Many existing works [5,6] of top- $k$  queries over uncertain objects which are modeled by pdfs in uncertainty regions. Moreover, due to some ambiguous conditions, the submitted queries are often fuzzy [7] so that the traditional top- $k$  query methods fail to query under these conditions. There are some fuzzy query methods [8] which can answer fuzzy queries but are not suitable for uncertain data. In light of the above circumstances, we extend the traditional notion of top- $k$  queries and then define a new kind of queries, Probabilistic Fuzzy Top- $k$  queries (PF-Top $k$  queries). In this kind of queries, users only desire the fixed number (e.g.,  $k$ ) results which satisfy the fuzzy query conditions and whose scores are the highest among all the uncertain objects. PF-Top $k$  queries are important to a wide range of applications, as illustrated by an example.

*Example 1:* Diagnosing H1N1 influenza is related to the feature vector, (*Body Temperature, White-cell chroma, Platelet Counts*). "*Body Temperature, White-cell chroma, Platelet Counts*" are three indicators of patients. All feature vectors obey 3-dimensional Gaussian distribution [5] in order to protect the privacy of patients. Eq.(1) shows pdf of feature vector of a patient as follows:

$$p(\mathbf{u}) = \frac{1}{(2\pi)^{3/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{u} - \mu)^t \Sigma^{-1}(\mathbf{u} - \mu)\right], \quad (1)$$

where  $\mu$  is the average of the distribution,  $\Sigma$  is a 3\*3 covariance matrix, and  $|\Sigma|$  represents its determinant. Doctors want to return the patients who have the highest  $k$  ranking probabilities having H1N1 influenza.

Example 1 illustrates the case of uncertain objects in high-dimensional uncertain space. Because there is no accurate condition which can conclude somebody surely has H1N1 influenza, the query results are unable to be obtained by the existing methods. Besides, lots of other applications such as uncertain object identification and some expert systems require this kind of queries. These applications have two common issues: one is that data uncertainty is inherent in the applications; the other is that the objects are retrieved under fuzzy conditions.

PF-Top $k$  queries defined by us aim at the above issues. To our knowledge, this is the first paper that processes fuzzy queries over uncertain objects. Our contributions are as follows.

- We formalize a novel kind of queries, Probabilistic Fuzzy Top- $k$  query (PF-Top $k$  query) which can effectively return  $k$  results from uncertain objects for fuzzy query conditions.
- We devise a query framework for answering PF-Top $k$  query, and present a pruning method to improve query efficiency by reducing the number of uncertain objects safely.
- We propose an iterative algorithm, Envelope Planes of Membership Function (EPMF) algorithm based on the upper and lower bounding functions, which can exactly answer PF-Top $k$  queries in high-dimensional space.
- We also propose an approximate algorithm, Middle Planes of Membership Function (MPMF) algorithm, which improves the efficiency while ensuring high precision by setting a proper value of parameter.

- We evaluate the efficiency and precision of our methods through the theoretical analysis and extensive experiments.

The rest of the paper is organized as follows. Section 2 gives the related work. Section 3 formally defines the problem of Probabilistic Fuzzy Top- $k$  query. Section 4 describes the process of our query. Section 5 provides the theoretical analysis. Section 6 gives the experimental results and we conclude in Section 7.

## 2 Related Work

Range query over uncertain data has received increasing attention in recent years. Tao et al. [9] propose the probabilistic range query method for the objects with imprecisely known locations; a pdf is associated with each object to represent the existence range of the object in the target space in which the location of the object is represented and a query region is specified by a rectangle. Yoshiharu et al. [10] present range query processing methods for spatial databases in which the location of the query object is imprecisely specified by a pdf based on a Gaussian distribution. Even if above work can query over uncertain objects, they do not have the top- $k$  semantics so that they cannot answer the top- $k$  query.

Nowadays, much research focuses on top- $k$  query over uncertain data. Many top- $k$  queries have been proposed according to different semantics, such as U-top $k$  [11], U- $k$ Ranks [12], and PT- $k$ /GT- $k$  [13]. These top- $k$  query methods are base on possible world model which is not suitable for modeling pdf. Besides, P $k$ NN queries [6] are a special kind of top- $k$  queries, which can return  $k$  objects which are closest to the query point among all the uncertain objects. However, both top- $k$  queries and P $k$ NN queries require the precise query conditions, so that they are difficult to answer fuzzy queries. The fuzzy theory has obtained wide attentions since it was proposed by Zadeh in 1965. Because the traditional methods cannot directly handle fuzzy queries, the solution is defining membership function according to the fuzzy query, then calculating the  $\alpha$ -intersect set [8] and acquiring the precise query conditions. However, current fuzzy query methods usually aim at certain data so that they cannot support the query over uncertain objects. This paper proposes Probabilistic Fuzzy Top- $k$  query methods, which can effectively answer fuzzy top- $k$  queries in the high-dimensional uncertain space.

## 3 Problem Formulation

In this section, we first define a function expressing the probabilities of uncertain objects satisfying a fuzzy query  $q$ , which is the score function in a PF-Top $k$  query. Assume that uncertain object set  $\mathcal{O}$  is in a  $d$ -dimensional uncertain space  $\mathcal{U}^d$ , in which each object is located within its own  $d$ -dimensional uncertainty region that is denoted by a  $d$ -dimensional rectangle called probabilistically constrained rectangle (PCR). The probabilities of the objects located in PCR are greater than the specified threshold value [9]. The fuzzy query  $q$  is defined as a membership



function  $A(\mathbf{u})$  which shows the extent of arbitrary point  $\mathbf{u}$  in  $\mathbf{U}^d$  satisfying query  $q$ . We extend  $d$ -dimensional uncertain space  $\mathbf{U}^d$  to  $(d + 1)$ -dimensional fuzzy query space  $\mathbf{Q}$  which is added in membership grade dimension ( $z$  axis) and set a point whose membership grade is one to origin point  $\mathbf{o}$ . We also assume that  $A(\mathbf{u})$  is a monotonic non-augmented function from  $\mathbf{o}$  along with arbitrary direction. In fact, there are many kinds of membership functions satisfying this assumption such as Cauchy distribution, Mountain distribution and so on. In this query space  $\mathbf{Q}$ , the score function is defined as follows.

**Definition 1.** (*Fuzzy Probability Function*) The fuzzy probability of object  $O$ ,  $Pr(O)$  expresses the probability satisfying fuzzy query  $q$  in query space  $\mathbf{Q}$ .

$$Pr(O) = \int_{PCR} A(\mathbf{u})p(\mathbf{u})d\mathbf{u}, \tag{2}$$

where  $p(\mathbf{u})$  is the pdf of  $O$ ,  $Pr(\cdot)$  denotes fuzzy probability function which is the score function in a PF-Topk query.

After defining fuzzy probability function, we formally define PF-Topk query with fuzzy probabilities.

**Definition 2.** (*Probabilistic Fuzzy Top-k Query, PF-Topk Query*) A Probabilistic Fuzzy Top-k query  $\mathbf{PFTQ}(q, k, \mathcal{O})$  returns the objects such that they have the highest  $k$  ranking fuzzy probabilities among all the uncertain objects. It is formally defined as follows:

$$\mathbf{PFTQ}(q, k, \mathcal{O}) = \{O_i | \forall i, j, O_i, O_j \in \mathcal{O}, Pr(O_i) > Pr(O_j), 1 \leq i \leq k < j \leq N\}, \tag{3}$$

where  $k$  is the number of uncertain objects to retrieve in the PF-Topk query,  $N$  is the size of uncertain object set  $\mathcal{O}$ .

**Definition 3.** (*Partitioning Uncertain Object*) Partitioning is defined that PCR of an uncertain object  $O$  is partitioned into  $r$  partitions which satisfy  $P_i \cap P_j = \emptyset$  and  $\bigcup_{t=1}^r P_t = PCR$ , where  $P_t$  is  $t$ -th partition of PCR. It is denoted by  $O = \bigcup_{t=1}^r o_t$  where  $o_t$  is  $t$ -th sub-object of  $O$  and located within  $P_t$ .

Figure 1 illustrates some uncertain objects and their sub-objects in a query space  $\mathbf{Q}$ . All the object are in uncertain space  $\mathbf{U}^d$ , where  $x_i$  and  $x_j$  are dimensions of this space.  $O_2 = \bigcup_{t=1}^4 o_t$ , where  $o_t$  is a sub-object of  $O_2$  located within  $P_t$ .  $O_2$  is partitioned based on some hyper surfaces  $S$  (e.g.  $S_t$ ) and  $P_t$  is generated by  $S_{t-1}$ ,  $S_t$  and the boundary of PCR of  $O_2$ .

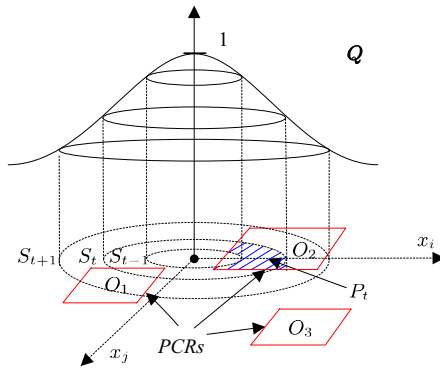
**Lemma 1.** Fuzzy probability of an uncertain object or a sub-object is equal to the expectation of membership function within its uncertainty region. Moreover, if the value of membership function in a PCR of object  $O$  is not less than that in the other PCR of  $O'$ , the fuzzy probability of  $O$  is not less than that of  $O'$ .

*Proof.* According to Definition 1, for an arbitrary uncertain object  $O$  and its sub-object  $o_i$ ,  $Pr(O) = \int_{PCR} A(\mathbf{u})p(\mathbf{u})d\mathbf{u}(\mathbf{u} \in PCR)$ ,  $Pr(o_i) = \int_{P_i} A(\mathbf{u})p(\mathbf{u})d\mathbf{u}(\mathbf{u} \in P_i)$ . The expectation of the membership function within its uncertainty region  $E(A(\mathbf{u})) = \int_{PCR} A(\mathbf{u})dp = \int_{PCR} A(\mathbf{u})p(\mathbf{u})d\mathbf{u}(\mathbf{u} \in PCR)$ ,  $E(A(\mathbf{u})) = \int_{P_i} A(\mathbf{u})dp = \int_{P_i} A(\mathbf{u})p(\mathbf{u})d\mathbf{u}(\mathbf{u} \in P_i)$ . Thus,

$$Pr(O) = E(A(\mathbf{u}))(\mathbf{u} \in PCR), \tag{4}$$

$$Pr(O_i) = E(A(\mathbf{u}))(\mathbf{u} \in P_i). \tag{5}$$

Suppose  $A = A(\mathbf{u})(\mathbf{u} \in PCR)$ ,  $A' = A(\mathbf{u})(\mathbf{u} \in PCR')$  and  $O'$  is an object located in  $PCR'$ ,  $A \geq A' \Rightarrow E(A) \geq E(A') \Rightarrow Pr(O) \geq Pr(O')$ .



**Fig. 1.** PF-Topk query space

From Lemma 1, it can be seen that the results of PF-Topk query are the highest  $k$  ranking average membership grade to satisfy the fuzzy query.

## 4 Probabilistic Fuzzy Top- $k$ Query Processing

In this section, we first present a query framework for answering PF-Topk queries. And then a pruning method is discussed. Moreover, we propose an effective query algorithm, Envelope Planes of Membership Function (EPMF) algorithm. Finally, we present a high-performance approximate algorithm for PF-Topk queries.

### 4.1 The Query Framework and Pruning Method

Figure 2 illustrates the query framework for answering PF-Topk queries. In particular, the framework is composed of two phases: filtering and querying. In the filtering phase, uncertain object set  $\mathcal{O}$  is filtered by our pruning method to reduce the costs of computation and I/O, and the unpruned objects consist of candidate set  $\mathcal{O}'$ . From Eq.(2), the fuzzy function of an object  $O$  is bounded by an interval  $[lo(O), up(O)]$ , where  $up(O)$  and  $lo(O)$  are the upper and lower bounds

of membership grade of  $O$  respectively. Actually, if there are more than  $k$  objects whose fuzzy probabilities are higher than that of  $O$ , it is certainly not the query result and should be pruned by the pruning method. Our pruning method is summarized in Theorem 1.

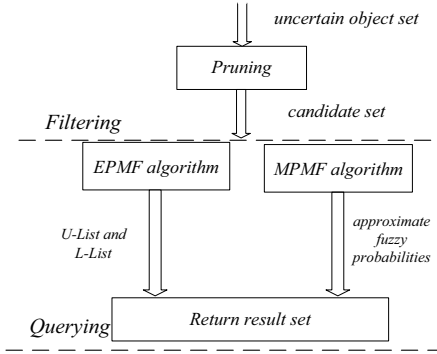


Fig. 2. The Query Framework for PF-Topk queries

**Theorem 1.** *An uncertain object can be safely pruned, if the upper bound of its membership grade is not more than the highest  $k$ -th ranking lower bound membership grade.*

*Proof.* According to the condition of Theorem 1, there are at least  $k$  objects whose lower bounds membership grades are not less than the upper bound of membership grade of an uncertain object  $O$ ,  $up(O)$ . Due to  $up(O) \geq Pr(O)$ , fuzzy probabilities of more than  $k$  objects are not less than  $Pr(O)$ . Thus,  $O$  is definitely not the query result and can be pruned.

### 4.2 Envelope Planes of Membership Function Algorithm

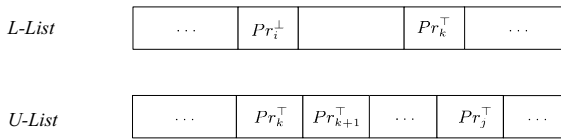
As mentioned previously, we can answer Probabilistic Fuzzy Top- $k$  queries (PF-Topk queries) by Eq.(3). Basic algorithm computes fuzzy probability for each uncertain object by Eq.(2) and compare the values in order to find the highest first  $k$  ranking fuzzy probabilities. Since Eq.(2) can be hardly calculated exactly when  $d \geq 2$ , it would be approximately calculated with Monte Carlo method which is popular with computation of high-dimensional numerical integrations. However, their costs are so high that we cannot get query results rapidly. In most cases, computing the bounds of fuzzy probability is much easier than computing the value of fuzzy probability. Thus, we propose an iterative algorithm, Envelope Planes of Membership Function (EPMF) algorithm based on the bounding functions, to answer PF-Topk queries in the querying phase.

Figure 3 illustrates an example of the upper and lower bounding lists (U-List and L-List) which store upper and lower bounds of fuzzy probabilities of all unpruned uncertain objects in the descending order respectively.  $Pr_j^+$  is the highest  $j$ -th ranking upper bound in U-List and  $Pr_i^+$  is the highest  $i$ -th ranking

lower bound in L-List. We summarize a theorem of the upper and lower bounds of fuzzy probabilities as follows.

**Theorem 2.** *If  $Pr_k^\perp \geq Pr_{k+1}^\top$ , the uncertain objects whose lower bound of fuzzy probabilities row first  $k$  in L-List are results of the PF-Topk query.*

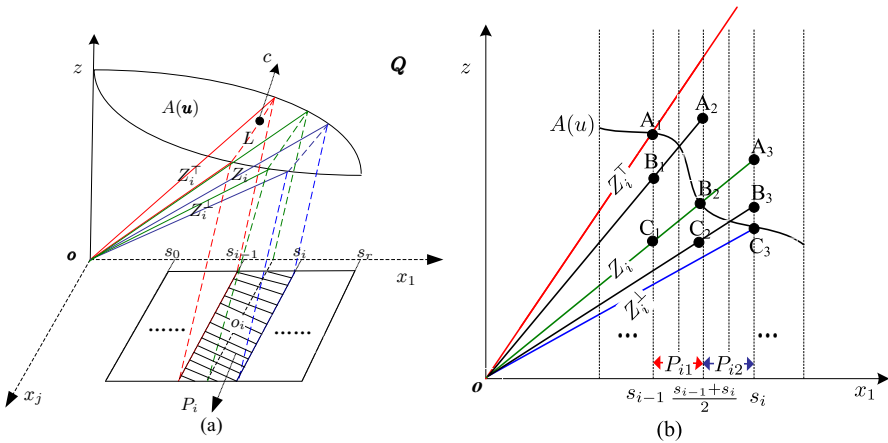
*Proof.* As shown in Figure 3, if any uncertain object  $O$  whose lower bound of fuzzy probability is at  $i$ -th position in L-List ( $\forall i \leq k$ ),  $Pr_i^\perp \geq Pr_k^\perp$ . In U-List,  $Pr_{k+1}^\top \geq Pr_j^\top$  ( $\forall j \geq k + 1$ ). According to the condition of Theorem 2,  $Pr_i^\perp \geq Pr_j^\top$ . Due to  $i \leq k$ , there are no more than  $k$  objects whose upper bounds of fuzzy probability are not less than that of  $O$ . Therefore, fuzzy probabilities of those objects (at most  $k$ ) are not less than that of  $O$ . That is to say,  $o$  is a result of PF-Topk query.



**Fig. 3.** The upper and lower bounding lists

According to Theorem 2, if we can find the proper bounds of fuzzy probabilities, the results of PF-Topk query can be obtained by U-List and L-List. Before searching the proper bounds, we need to explain the definition of envelope planes of membership function, which is summarized as follows.

**Definition 4.** (*Envelope Planes of Membership Function*) *In query space  $Q$ , if  $(d+1)$ -dimensional plane  $Z^\top$  ( $Z^\perp$ ) is not less (more) than the membership function located within a region and it includes origin point  $o$ , it is known as the upper (lower) envelope plane of membership function of this region.*



**Fig. 4.** Envelope Planes of Membership Function

Figure 4 illustrates the upper and lower envelope planes of a sub-region  $P_i$ . Uncertain object  $O = \bigcup_{i=1}^r o_i$ , and  $O$ 's PCR  $= \bigcup_{i=1}^r P_i$  based on  $(r - 1)$  surfaces  $x_1 = s_i$  ( $s_0 \leq s_{i-1} \leq s_i \leq s_{i+1} \leq s_r$  and  $1 \leq i \leq r - 1$ ). Due to our assumption for membership function in Section 3, it is monotone non-augment from origin point  $\mathbf{o}$  to other point along with the positive direction of  $x_1$  axis. Therefore, the  $(d + 1)$ -dimensional plane  $Z_i^\top$  which includes  $\mathbf{o}$  and intersection  $LS_{i-1}$  generated by  $x_1 = s_{i-1}$  and membership function is not less than membership function located in  $P_i$ . From Definition 4,  $Z_i^\top$  is the upper envelope plane of  $P_i$ . Similarly,  $Z_i^\perp$  is the lower envelope plane of  $P_i$ . According to point equation of  $(d + 1)$ -dimensional plane, the envelope plane equation can be deduced by Eq.(6).

$$\begin{vmatrix} x_1 & x_2 & \dots & x_j & \dots & x_d & z \\ s_{i-1} & c_{22} & \dots & c_{2j} & \dots & c_{2d} & c_{2(d+1)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ s_{i-1} & c_{(d+1)2} & \dots & c_{(d+1)j} & \dots & c_{(d+1)d} & c_{(d+1)(d+1)} \end{vmatrix} = 0, \tag{6}$$

where  $d$  arbitrary points  $\mathbf{c}_2 = (s_{i-1}, c_{22}, \dots, c_{2j}, \dots, c_{2d}, c_{2(d+1)}), \dots, c_{d+1} = (s_{i-1}, c_{(d+1)2}, \dots, c_{(d+1)j}, \dots, c_{(d+1)d}, c_{(d+1)(d+1)})$  (e.g.,  $\mathbf{c}$  in Figure 4(a)) are located in the corresponding intersection. In general, the costs of obtaining envelope plane equations are much less than those of computing integrations so they can be ignored. To compute the bounds of fuzzy probabilities of arbitrary object, a kind of bounding functions is defined with envelope plane equations, which is explained in Theorem 3.

**Theorem 3.** *The upper and lower envelope plane equations of a sub-region  $P_i$  is denoted by  $Z_i^\perp : \sum_{j=1}^d a_{ij}x_j$  and  $Z_i^\top : \sum_{j=1}^d b_{ij}x_j$  in query space  $\mathbf{Q}$ . The upper and lower bounding functions are equal to  $\sum_{i=1}^r a_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{i=1}^r \sum_{j=2}^d a_{ij} E(X_j)$  and  $\sum_{i=1}^r b_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{i=1}^r \sum_{j=2}^d b_{ij} E(X_j)$ , where  $X_j$  is a random variable with marginal distribution  $p_j(x_j)$  of  $O$  in dimension  $x_j$ .*

*Proof.* According to Definition 4,  $Z_i^\top$  is not less than the membership function  $A(\mathbf{u})(\mathbf{u} \in P_i)$ , so  $E(Z_i^\top) \geq E(A(\mathbf{u}))(\mathbf{u} \in P_i)$ . Due to Lemma 1,  $E(Z_i^\top) \geq Pr(o_i)$ . Since the expectation of linear function equals to its expectation function,  $E(Z_i^\top) = E(\sum_{j=1}^d a_{ij}x_j) = \sum_{j=1}^d a_{ij} \int_{P_i} x_j p(\mathbf{u}) d\mathbf{u} = a_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{j=2}^d a_{ij} E(X_j)$ . For all the other sub-objects of  $O$ ,  $\sum_{i=1}^r a_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{i=1}^r \sum_{j=2}^d a_{ij} E(X_j) \geq \sum_{i=1}^r Pr(o_i) = Pr(O)$ . Therefore, the upper bounding function of fuzzy probability is as follows:

$$Pr^\top(O) = \sum_{i=1}^r a_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{i=1}^r \sum_{j=2}^d a_{ij} E(X_j). \tag{7}$$

Likewise, the lower bounding function of fuzzy probability is as follows:

$$Pr^\perp(O) = \sum_{i=1}^r b_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{i=1}^r \sum_{j=2}^d b_{ij} E(X_j). \quad (8)$$

As indicated by Theorem 3, the bounds of fuzzy probabilities of uncertain objects can be computed by the bounding functions. And the cost of computation of some 1-dimensional integrations and expectations in Eqs. (7) and (8) is much less than that of high-dimensional numerical integration in Basic algorithm. Moreover, Eqs. (7) and (8) can be exactly calculated without Monte Carlo method. Thus, we propose Envelope Planes of Membership Function (EPMF) algorithm based on these bounding functions. This algorithm continuously increases the number  $r$  of partitioned sub-regions and iteratively calculates the bounds of fuzzy probabilities until satisfying  $Pr_k^\perp \geq Pr_{k+1}^\top$  in the bounding lists. The key steps of EPMF algorithm are illustrated in Algorithm 1.

---

**Algorithm 1.** Envelope Planes of Membership Function Algorithm

---

**Input** :  $PFTQ(q, k, \mathcal{O}')$

**Output**: U-List and L-List,  $k$  objects in result set  $\mathfrak{R}$

```

1 initialize  $r, i \leftarrow 2$ 
2 if  $M \leq k$  then
3   take all object in  $\mathcal{O}'$  into  $\mathfrak{R}$ , break
4 for each object  $O$  in  $\mathcal{O}'$  do
5   while  $i \leq r$  do
6      $s_{i-1} \leftarrow 1/r * ((i-1) * s_r + (r-i+1) * s_0)$ 
7      $s_i \leftarrow 1/r * (i * s_r + (r-i) * s_0)$ 
8      $i \leftarrow i + 1$ 
9   end
10  obtain envelope plane equations by Eq. (6)
11  compute the bounds of fuzzy probability of  $O$  by Eqs.(7) and (8)
12 end
13 take the values of upper (lower) bounds into U-List (L-List) in descending order
14 if  $Pr_k^\perp \geq Pr_{k+1}^\top$  then
15   take objects whose lower bounds row first  $k$  in L-List into  $\mathfrak{R}$ , break
16 for each object  $O$  do
17   if  $Pr_k^\perp \geq Pr^\top(O)$  then
18     delete them from  $\mathcal{O}'$ 
19 end
20  $r \leftarrow 2r, i \leftarrow 2$ , goto step 2
```

---

The EPMF algorithm first measures the size of candidate set  $\mathcal{O}'$  which must be not more than  $k$ , otherwise all the objects in  $\mathcal{O}'$  are results (lines 2-3). Then, we do  $r$  equip-partition the range  $([s_0, s_r], s_0 * s_r \geq 0)$  along with  $x_1$  axis direction for each object and evaluate  $s_{i-1}$  and  $s_i$  (lines 6-8). If an object is located at two sides of some axis (e.g.  $O_1$  in Figure 1), it is partitioned based on this axis into two partitions before evaluating. After that, if it is not content with Theorem 2,

we delete unnecessary objects from  $\mathcal{O}'$  and iteratively execute the procedure with double  $r$  (lines 16-20).

### 4.3 Middle Planes of Membership Function Algorithm

In the previous subsection, we have discussed an exact and iterative EPMF algorithm for PF-Topk queries. However, the efficiency of this algorithm decreases when the iterations become extremely large. Aiming at this problem, we propose an approximate algorithm, Middle Planes of Membership Function (MPMF) algorithm to improve the efficiency.

As shown in Figure 4, a middle plane  $Z_i$  is generated by origin point  $\mathbf{o}$  and an intersection generated by  $x_1 = (s_{i-1} + s_i)/2$  and membership function. Similar to the envelope plane equations, the middle plane equation can be obtained by Eq.(7). MPMF algorithm is proposed based on middle plane equations of all sub-regions of an uncertain object. Its main idea is that the sum of expectations of middle plane equations of all sub-regions replaces the value of fuzzy probability and then  $k$  objects whose sum values are highest ranked are returned. Assume that  $Z_i$  is middle plane of sub-region  $P_i$ , whose equation is denoted by  $Z_i : z = \sum_{j=1}^d e_{ij}x_j$ , where  $e_{ij}$  is the  $j$ -th coefficient of  $Z_i$ . The expectations sum of middle plane equations of all sub-regions

$$\sum_{i=1}^r E(Z_i) = \sum_{i=1}^r e_{i1} \int_{S_{i-1}}^{S_i} x_1 p_1(x_1) dx_1 + \sum_{i=1}^r \sum_{j=2}^d e_{ij} E(X_j). \quad (9)$$

We set the approximate value  $Pr(O)$  of fuzzy probability as the value computed by Eq.(9). The key steps of MPMF algorithm are illustrated in Algorithm 2.

---

#### Algorithm 2. Middle Planes of Membership Function Algorithm

---

**Input** :  $PFTQ(q, k, \mathcal{O}')$

**Output**:  $k$  objects in result set  $\mathfrak{R}$

- 1 initialize  $r, i \leftarrow 2$
  - 2 **if**  $M \leq k$  **then**
  - 3     take all object in  $\mathcal{O}'$  into  $\mathfrak{R}$ , break
  - 4 **for** each object  $O \in \mathcal{O}'$  **do**
  - 5     **while**  $i \leq r$  **do**
  - 6          $s_{i-1} \leftarrow 1/r * ((i-1) * s_r + (r-i+1) * s_0)$
  - 7          $s_i \leftarrow 1/r * (i * s_r + (r-i) * s_0)$
  - 8          $i \leftarrow i + 1$
  - 9     **end**
  - 10     obtain middle plane equations by Eq. (6)
  - 11     compute  $Pr(O)$  by Eq. (9)
  - 12 **end**
  - 13 take the highest values computed by step 11 first  $k$  ranking objects into  $\mathfrak{R}$
-

The procedure of MPMF algorithm is the same as that of EPMF algorithm except computing the approximate values of fuzzy probabilities (lines 10-11) instead of the bounds of fuzzy probabilities. Its efficiency is higher than that of EPMF algorithm because it needs not iterative computation. The error produced by MPMF algorithm is estimated in the next two sections.

### 5 Theoretical Analysis

Up to now, we have explained some algorithms to answering PF-Top $k$  queries. In this section, we quantitatively estimate these algorithms with time complexity and error. In  $d$ -dimensional uncertain space  $U^d$ , assume that the size of candidate set  $\mathcal{O}'$  is  $M$ , the average integration complexity of integrating marginal distribution function in a dimension is  $|C_1|$ , and that of membership function in a dimension is  $|C_2|$ . The complexities of the three algorithms proposed by us are shown in Table 1.

Obviously, the time complexity of Basic algorithm follows exponential growth with the dimensionality increasing and its complexity is much higher than others. For EPMF algorithm, the number of partitioned sub-regions is  $r$  in the first iteration, but  $2r$  in the next time. Since some objects is deleted from  $\mathcal{O}'$  (line 16 in Algorithm 1) each time, the number of rest objects in  $\mathcal{O}'$  is in a interval  $[0, M]$ . If  $m$  expresses the iterations of EPMF algorithm, the number of 1-dimensional integrations in Eq.(8) or (9) is equal to  $2^m dr$  in the  $m$ -th iteration. Therefore, the complexity of EPMF algorithm is variable and the upper bound obeys exponential growth along with the iterations  $m$  increasing. The complexity of MPMF algorithm is best and stable since it need not to iteratively calculate.

**Table 1.** Time complexities of algorithms

Algorithm	The Lower Bound	The Upper Bound
Basic algorithm	$O(M( C_1  C_2 )^d)$	$O(M( C_1  C_2 )^d)$
EPMF algorithm	$O(2Mdr C_1 )$	$O(2^{m+1}Mdr C_1 )$
MPMF algorithm	$O(Mdr C_1 )$	$O(Mdr C_1 )$

Next, we conduct error analysis for our algorithms. As mentioned earlier, Basic algorithm calculates Eq. (2) with Monte Carlo method. According to Kolmogorov’s theorem,  $|Y - Y_n|$  is bounded above by  $1.36/\sqrt{n}$  with probability 95%, where  $Y$  is the truth value of the function and  $Y_n$  is the estimated value which is calculated by  $n$  random numbers. Hence, the upper bound reduces to  $1/\sqrt{2}$  times by doubling  $n$ , which results in doubling cost. From Eq. (9), the error  $\varepsilon$  in MPMF algorithm is  $|\sum_{i=1}^r E(Z_i) - Pr(O)| = \sum_{i=1}^r \max\{E(Z_i) - A(\mathbf{u}), (\mathbf{u} \in P_i)\}$ . Therefore, the upper bounds of error can be expressed as follows:

$$\varepsilon^T = \sum_{i=1}^r \max\{E(Z_i) - A(\mathbf{u}), (\mathbf{u} \in P_i)\}. \tag{10}$$



As shown in Fig.4(b),  $\max\{E(Z_i) - A(\mathbf{u}) (\mathbf{u} \in P_i)\}$  is equal to  $\max\{A_1C_1, A_3C_3\}$ . Each sub-region is equip-partitioned into two new partitions, which results in generating two new middle planes of the two partitions. The upper bound of error is calculated in the new partitions as follows.

$$\varepsilon^\top = \sum_{i=1}^r (\max\{E(Z_{i1}) - A(\mathbf{u}), (\mathbf{u} \in P_{i1})\} + \max\{E(Z_{i2}) - A(\mathbf{u}), (\mathbf{u} \in P_{i2})\}), \quad (11)$$

$\max\{E(Z_{i1}) - A(\mathbf{u}), (\mathbf{u} \in P_{i1})\} + \max\{E(Z_{i2}) - A(\mathbf{u}), (\mathbf{u} \in P_{i2})\}$  is equal to  $\max\{A_1B_1, A_2B_2\} + \max\{B_2C_2, B_3C_3\}$  ( $A_iB_i$  is the difference of the two points), which is almost half of  $\max\{A_1C_1, A_3C_3\}$  when  $r$  is large enough. Note that, the upper bound of error  $\varepsilon^\top$  almost reduces by half with the cost doubled (due to doubling  $r$ ) in MPMF algorithm. Thus, MPMF algorithm can ensure high precision by increasing the value of  $r$ , and  $\varepsilon^\top$  decreasing ratio of MPMF algorithm is more than that of Monte Carlo method with the cost doubled.

## 6 Experiments

### 6.1 Experiment Settings

All the experiments were conducted on a PC with 2.5GHz Pentium processor and 1G main memory. We used B-tree to index bounds of membership grade of uncertain objects. We used "Los Angeles" dataset, a 2-dimensional real dataset (D4) available in [15], with 60K geographical objects described by ranges of longitudes and latitudes, and used Gaussian distribution as the objects' pdfs. For synthetic data, we generated uncertain objects in  $d$ -dimensional uncertain space  $\mathbf{U}^d$ . Specifically, we randomly produced the range for all uncertain objects whose ranges are PCRs. We varied the value of some parameters to generate three kinds of synthetic datasets (D1, D2, D3) in which uncertain objects respectively obey 3-dimensional Gaussian distribution, 3-dimensional Uniform distribution, and 4-dimensional Gaussian distribution. The membership function  $A(\mathbf{u})$  obeys Cauchy distribution and we set it as follows:

$$A(\mathbf{u}) = \frac{1}{1 + \sum_{j=1}^d \alpha_j (x_j)^{\beta_j}}, \quad (12)$$

where  $\alpha_j$  is more than zero and  $\beta_j$  is a positive even.

### 6.2 Experimental Results

In this subsection, we evaluate the performance of our proposed algorithm (Basic algorithm without pruning, Basic algorithm, EPMF algorithm, and MPMF algorithm) to answer PF-Topk queries through experiments. For two Basic algorithms, we used RANDLIB [14] for generating  $n$  random numbers obeying of pdfs uncertain objects, and computed integrations with Monte Carlo method.

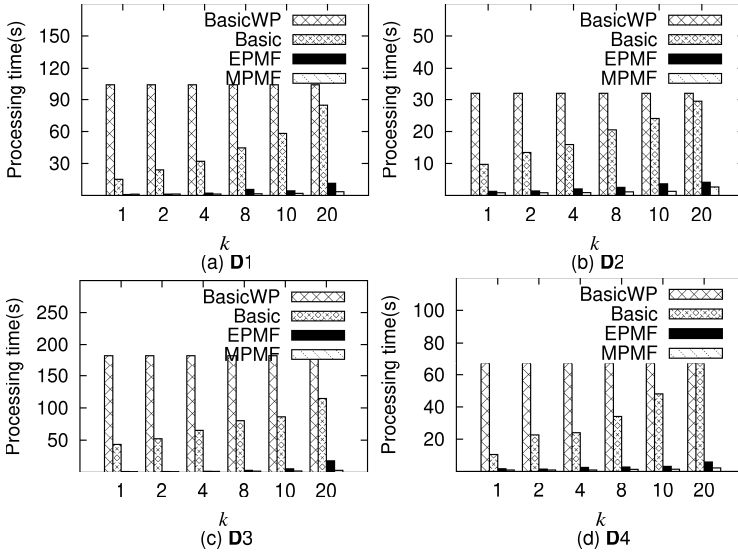
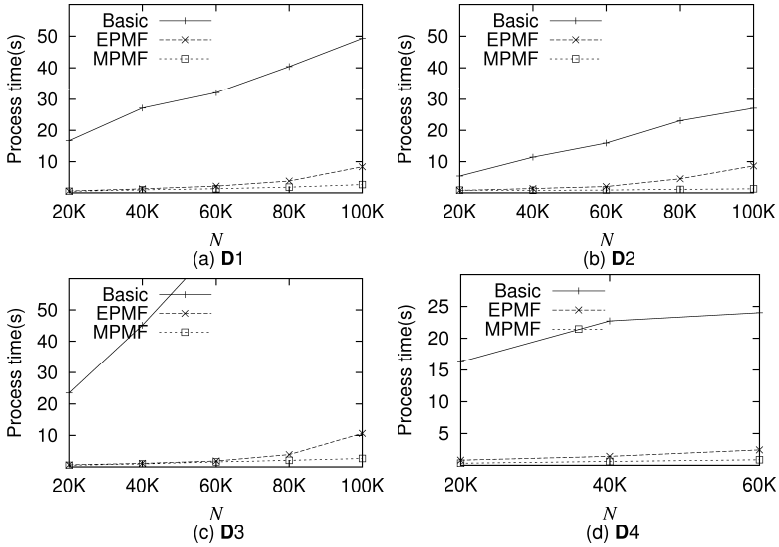


Fig. 5. Effect of the number of objects to retrieve on processing time

First, we examine the effect of objects number  $k$  to retrieve on the processing time in **D1**, **D2**, **D3** and **D4**. We set the size of dataset  $N=60K$  and vary  $k$  from 1 to 20. Figure 5 shows the experimental results. For any dataset, the processing time of Basic algorithm without pruning is the highest and constant regardless of the varied value of  $k$ . Along with  $k$  increasing, the processing time of other algorithms raise because our pruning method is sensitive to  $k$  and more effective when  $k$  is lower. Moreover, from the speed-up ratio of the processing time, we can find MPMF algorithm performs better than Basic and EPMF algorithm.

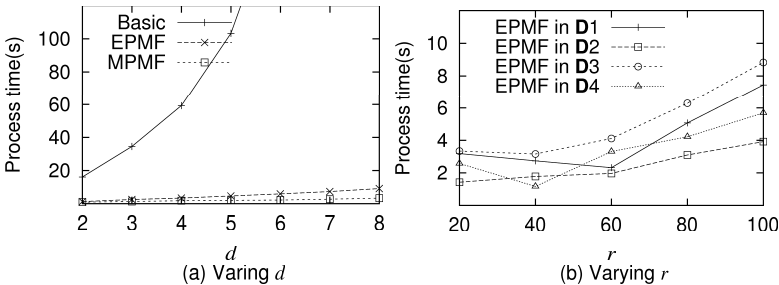
We also investigate the effect of the dataset size  $N$  on the processing time. Figure 6 illustrates the results by varying  $N$ . It can be obviously seen that the processing time of the three algorithms increases along with  $N$  increasing. Furthermore, Basic algorithm is inefficient because of considerable large computing costs. The processing time of EPMF algorithm grows rapidly when  $N$  becomes large. The reason is that the density of objects augments with increasing  $N$ , which results in fuzzy probabilities of some objects are near so it needs iteration many times. But the cost of MPMF algorithm is least and most steady. It can be explained that MPMF algorithm does not need computing iteratively and the cost of computation is quite low.

In the next experiment, we analyze the relationship between the dimensionality  $d$  of uncertain space and the processing time. We randomly generate 10000 uncertain objects with Gaussian distribution in  $d$ -dimensional space and vary  $d$  from 2 to 8. Figure 7(a) illustrates the processing time of different algorithms returning first 4 objects. The processing time of Basic algorithm is exponential growth and much higher than that of two other algorithms which is almost linear



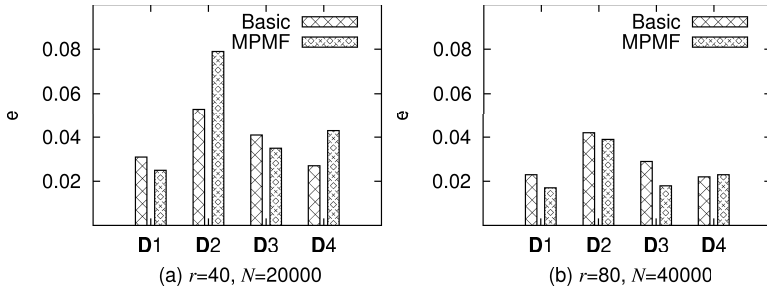
**Fig. 6.** Effect of the size of dataset on processing time of different algorithms

growth along with  $d$  increasing. Due to calculating without high-dimensional integrations, EPMF algorithm and MPMF algorithm show good performance in high-dimensional space. Then we examine the effect of the number  $r$  of partitioned sub-regions on the processing time by comparing the processing time of EPMF algorithm in different datasets. Figure 7(b) shows the results of varying  $r$  from 20 to 100. We can see that the processing time is higher when  $r$  is too larger or too smaller for any dataset. Note that, since oversized  $r$  has higher cost of each iteration and undersized  $r$  causes the increasing of iterations.



**Fig. 7.** Effect of  $d$  and  $r$  on processing time

Final, we evaluate the precision of Basic algorithm and MPMF algorithm. We test incorrect rate  $e$  of the two algorithms which is the mean ratio of incorrect results number to return ones number in 50 times of PF-Topk queries with different datasets. The results are given in Figure 8. When  $r=40$  and  $N=20000$ ,



**Fig. 8.** Incorrect rate  $e$  with different datasets

it is seen that  $e$  of MPMF algorithm is near to that of Monte Carlo method. However, it is easier to satisfy the precision of requirement than that of Monte Carlo method with the cost increasing.

## 7 Conclusions

In this paper, we have described a novel kind of queries, PF-Top $k$  queries which retrieve  $k$  objects having the highest  $k$  ranking fuzzy probabilities among all uncertain objects. The query process consists of two phases: filtering and querying. In the first phase, a pruning method has effectively pruned the objects which are definitely not query results. In the next phase, we have proposed an exact algorithm, EPMF algorithm, and an approximate algorithm, MPMF algorithm, to answer PF-Top $k$  queries in  $d$ -dimensional uncertain space efficiently. Moreover, MPMF algorithm can improve efficiency while ensuring high precision by setting a proper value of the number of partitioned sub-regions. Finally, we have identified the efficiency and precision of the proposed methods through the theoretical analysis and extensive experiments.

## References

1. Deshpande, A., Guestrin, C., Madden, S.R., Hellerstein, J.M., Hong, W.: Model-driven data acquisition in sensor networks. In: VLDB '04, pp. 588–599 (2004)
2. Chen, L., Özsu, M., Oria, V.: Robust and fast similarity search for moving object trajectories. In: SIGMOD '05, pp. 491–502 (2005)
3. Christian, B., Alexey, P., Matthias, S.: The gauss-tree: Efficient object identification in databases of probabilistic feature vectors. In: ICDE '06 (2006)
4. Cheng, R., Prabhakar, S., Kalashnikov, D.V.: Querying imprecise data in moving object environments. IEEE TKDE 16 (2002)
5. Lian, X., Chen, L.: Probabilistic ranked queries in uncertain databases. In: EDBT '08, pp. 511–522 (2008)

6. Beskales, G., Soliman, M.A., Ilyas, I.F.: Efficient search for the top-k probable nearest neighbors in uncertain databases. *VLDB Endow* 1(1), 326–339 (2008)
7. Bodenhofer, U., Küng, J., Saminger, S.: Flexible query answering using distance-based fuzzy relations. In: de Swart, H., Orłowska, E., Schmidt, G., Roubens, M. (eds.) *TARSKI 2006. LNCS (LNAI)*, vol. 4342, pp. 207–228. Springer, Heidelberg (2006)
8. Chen, S.M., Jong, W.T.: Fuzzy query translation for relational database systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 27(4), 714–721 (1997)
9. Tao, Y., Xiao, X., Cheng, R.: Range search on multidimensional uncertain data. *ACM Trans. Database Syst.* 32(3), 15 (2007)
10. Ishikawa, Y., Iijima, Y., Yu, J.X.: Spatial range querying for gaussian-based imprecise query objects. In: *ICDE '09*, pp. 676–687 (2009)
11. Soliman, M.A., Ilyas, I.F., Chang, K.C.: Top-k query processing in uncertain databases. In: *ICDE' 07*, pp. 896–905 (2007)
12. Yi, K., Li, F., Kollios, G., Srivastava, D.: Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Trans. on Knowl. and Data Eng.* 20(12), 1669–1682 (2008)
13. Hua, M., Pei, J., Zhang, W., Lin, X.: Efficiently answering probabilistic threshold top-k queries on uncertain data. In: *ICDE'08*, pp. 1403–1405 (2008)
14. Topologically integrated geographic encoding and referencing (tiger) system, <http://www.census.gov/geo/www/tiger/>
15. RANDLIB, <http://biostatistics.mdanderson.org/softwaredownload/>

# Performance and Power Evaluation of Flash-Aware Buffer Algorithms

Yi Ou, Theo Härder, and Daniel Schall

University of Kaiserslautern, Germany  
{ou,haerder,schall}@cs.uni-kl.de

**Abstract.** With flash disks being an important alternative to conventional magnetic disks, various design aspects of DBMSs, whose I/O behavior is performance-critical, and especially their I/O architecture should be reconsidered. Taking the distinguished characteristics of flash disks into account, several flash-aware buffer algorithms have been proposed with focus on flash-specific performance optimizations. We introduce several basic principles of flash-aware buffer management and evaluate performance and energy consumption of related algorithms in a DBMS environment using both flash disks and magnetic disks. Our experiments reveal the importance of those principles and the potential of flash disks both in performance improvement and in energy saving.

## 1 Introduction

In recent years, green computing gained a lot of attention and visibility by public discussion concerning energy waste and, in turn, global warming. As a consequence, thermal management is seriously addressed in IT to reduce power usage, heat transmission, and, in turn, cooling needs. Here, we consider the role DBMSs can play for green IT.

So far, flash disks were primarily considered ideal for storing permanent data in embedded devices such as personal digital assistants (PDAs) or digital cameras, because they have no mechanical parts and are energy efficient, small, light-weight, noiseless, and shock resistant. Furthermore, they provide the great advantage of zero-energy needs in idle or turned-off modes. Because of breakthroughs in bandwidth (I/Os), energy saving, reliability, and volumetric capacity [1], flash disks will also play an increasingly important role in DB servers. However, to optimize their performance and, at the same time, energy efficiency, especially their I/O architecture must be reconsidered.

The most important building blocks of flash disks are flash memory and flash translation layer (FTL). Logical block addresses are mapped by the FTL to varying locations on the physical medium. This mapping is required due to the intrinsic limitations of flash memory [2]. The implementation of FTL is device-related and supplied by the disk manufacturer. Several efforts were recently made to systematically benchmark the performance of flash disks [1,3,4]. The most important conclusions of these benchmarks are:

- For sequential read-or-write workloads, flash disks often achieve a performance comparable to high-end magnetic disks.
- For random workloads, the performance asymmetry of flash disks and their difference to magnetic disks is significant: random reads are typically two orders of magnitude faster than those on magnetic disks, while random writes on flash disks are often even slower than those on magnetic disks.

As an example, one of our middle-class flash disk achieves 12,000 IOPS for random reads and only 130 IOPS for random writes of 4 KB blocks, while high-end magnetic disks typically have more than 200 IOPS for random I/O [1]. Interestingly, due to the employment of device caches and other optimizations in the FTL, page-level writes with strong spatial locality can be served by flash disks more efficiently than write requests without locality. In particular, many benchmarks show that flash disks can handle random writes with larger *request sizes* more efficiently. For example, the bandwidth of random writes using units of 128 KB can be more than an order of magnitude higher than writing at units of 8 KB. In fact, a write request of, say 128 KB, is internally mapped to 64 *sequential* writes of 2-KB flash pages inside a flash block. Note that sequential access is an extreme case of high spatial locality.

## 1.1 Basic Principles

In DBMS buffer management, the fact “whether a page is read only or modified” is an important criterion for the replacement decision [5]. To guarantee data consistency, a modified buffer page has to be written back to disk (called *physical write* or *page flush*), before the memory area can be reused. Hence, if the replacement victim is dirty, the process or thread requesting an empty buffer frame must wait until page flush completion—potentially a performance bottleneck.

This criterion is now much more important in our context, because, for flash disks, the average cost of a page write (including block erasure) may be two orders of magnitude higher than that of a page read. At a point in time running a given workload, if a clean page  $p$  is to be re-read  $n$  times and a dirty page  $q$  to be re-modified  $m$  times ( $n \sim m$ ), the buffer manager should replace  $p$  in favor of  $q$ , because the benefit of serving  $m$  write requests directly from the buffer is much higher than the cost of  $n$  repeated flash reads.

Yet the total cost of page flushing is not linear to the number of page flushes. As introduced above, write patterns strongly impact the efficiency of flash writing; hence, they have to be addressed as well. An intuitive idea is to increase the DB *page size* so that we can write more efficiently. In most systems, the page size is the unit of data transfer between the buffer layer and the file system (or the raw device directly). It would be an attractive solution if the overall performance could be improved this way, because only a simple adjustment of a single parameter would be required. However, a naive increase of the page size generally leads to more unnecessary I/O (using the same buffer size), especially for OLTP

workloads, where random accesses dominate. Furthermore, in multi-user environments, large page sizes favor thread contentions. Hence, a more sophisticated solution is needed.

Even with flash disks, maintaining a high hit ratio—the primary goal of conventional buffer algorithms—is still important, because the bandwidth of main memory is at least an order of magnitude higher than the interface bandwidth of storage devices. Based on the flash disk characteristics and our discussion so far, we summarize the basic principles of flash-aware buffer management as follows:

- P1** Minimize the number of physical writes.
- P2** Address write patterns to improve the write efficiency.
- P3** Keep a relatively high hit ratio.

## 1.2 Contributions

Previous research on flash-aware buffer management focused on flash-specific performance optimizations. However, an important question remains open: how well do the flash-aware algorithms perform on conventional magnetic disks? This question is important because, due to the lower \$/GB cost, magnetic disks will certainly remain dominant in the near future, therefore enterprises have to deal with the situations where both kinds of devices co-exist. Another interesting aspect ignored in all previous works is the energy consumption of related algorithms, which calls more and more attention in server environments and is often critical in environments where flash devices are deployed in the first place, e.g., mobile data management. The major contributions of this paper are:

- We accomplish an extensive performance study of related algorithms in a DBMS-based environment using a variety of flash disks and magnetic disks. This device-sensitivity study is missing in all previous works, where evaluation was often performed on a single simulated flash device. Furthermore, the performance study emphasizes the basic principles introduced in Sect. 1.1.
- We examine the energy consumption of the system running these algorithms using a tailor-made energy measurement device. Our experiments reveal a strong correlation between system performance and energy consumption and demonstrated the great energy-saving potential of flash disks.

The remainder of the paper is organized as follows. Sect. 1.2 sketches related work. Sect. 1.3 discusses one of the flash-aware buffer algorithms in detail. Our experiments are presented in Sect. 1.4. The concluding remarks are given in Sect. 1.5.

## 2 Related Work

LRU and CLOCK [6] are among the most widely-used replacement policies. The latter is functionally identical to Second Chance [7]: both of them often achieve hit ratios close to those of LRU.

CFLRU [8] is a flash-aware replacement policy for operating systems based on LRU. It addresses the asymmetry of flash I/O by allowing dirty pages to stay in



the buffer longer than clean pages. The LRU list of CFLRU is divided into two regions: the *working region* at the MRU (most recently used) end of the list, and the *clean-first region* at the LRU end, where clean pages are always selected as victims over dirty pages. In this way, the buffer area for dirty pages is effectively increased—thus, the number of flash writes can be reduced.

LRU-WSR [9] is a flash-aware algorithm based on LRU and Second Chance, using only a single list as auxiliary data structure. The idea is to evict clean and cold-dirty pages and keep the hot-dirty pages in buffer as long as possible.

REF [10] is also a flash-aware replacement policy based on LRU. It maintains an LRU list and has a *victim window* at the MRU end of the list, similar to the clean-first region of CFLRU. Victim pages are only selected from the *victim blocks*, which are blocks with the largest numbers of pages in the victim window.

CFLRU and LRU-WSR do not address the problem of write patterns, while REF does not distinguish between the clean and dirty states of pages. To the best of our knowledge, CFDC [11] is the only flash-aware algorithm that applied all the three basic principles P1 to P3 introduced in Sect. 2. The study of the energy behavior of related buffer management algorithms distinguishes our work from that of [4], where the impact of the device type (flash disk or magnetic disk) on the energy efficiency of a complete database system is examined and discussed.

### 3 The CFDC Algorithm

For comprehension, we repeat the essential properties of the CFDC (Clean-First Dirty-Clustered) algorithm.

#### 3.1 The Two-Region Scheme

CFDC manages the buffer in two regions: the *working region*  $W$  for keeping *hot* pages that are frequently and recently revisited, and the *priority region*  $P$  responsible for optimizing replacement costs by assigning varying priorities to page clusters. A parameter  $\lambda$ , called *priority window*, determines the size ratio of  $P$  relative to the total buffer. Therefore, if the buffer has  $B$  pages, then  $P$  contains  $\lambda$  pages and the remaining  $(1 - \lambda) \cdot B$  pages are managed in  $W$ . Note,  $W$  does not have to be bound to a specific replacement policy. Various conventional replacement policies can be used to maintain high hit ratios in  $W$  and, therefore, prevent hot pages from entering  $P$ .

The parameter  $\lambda$  of CFDC is similar to the parameter *window size* ( $w$ ) of CFLRU. The algorithm REF has a similar configurable *victim window* as well. For simplicity, we refer to them uniformly with the name “window size”. If a page in  $P$  is hit, the page is moved (promoted) to  $W$ . If the page hit is in  $W$ , the base algorithm of  $W$  should adjust its data and structures accordingly. For example, if LRU is the base algorithm, it should move the page that was hit to the MRU end of its list structure. In case of a buffer fault, the victim is always first selected from  $P$ . Only when all pages in  $P$  are fixed<sup>1</sup>, we select the victim from  $W$ . Considering recency, the newly fetched page is first promoted to  $W$ .

<sup>1</sup> The classical pin-use-unpin (or fix-use-unfix) protocol [12] is used for page requests.

### 3.2 Priority Region

Priority region  $P$  maintains three structures: an LRU list of clean *pages*, a priority queue of *clusters* where dirty pages are accommodated, and a hash table with cluster numbers as keys for efficient cluster lookup. The cluster number is derived by dividing page numbers by a constant *cluster size*.

In our context, *spatial locality* refers to the property of contiguously accessed DB pages being physically stored close to each other. A *cluster* is a set of pages located in proximity, i.e., whose page numbers are close to each other. Though page numbers are logical addresses, because of the space allocation in most DBMSs and file systems, the pages in the same cluster have a high probability of being physically neighbored, too. The size of a cluster should correspond, but does not have to be strictly equal to the size of a flash block, thus information about exact flash block boundaries are not required.

CDFC's principles of victim selection are:

- Inside  $P$ , clean pages are always selected over dirty pages. If there is no clean page available, a *victim cluster* having the lowest priority is selected from the priority queue.
- The oldest page in the victim cluster will be evicted first, if it is not re-referenced there. Otherwise, it would have been already promoted to  $W$ .
- The priority of the victim cluster is set to minimum and will not be updated anymore, so that the next victim pages will still be evicted from this cluster, resulting in strong spatial locality of page evictions.

For a cluster  $c$  with  $n$  (in-memory) pages, its priority  $P(c)$  is computed according to Formula [1](#):

$$P(c) = \frac{\sum_{i=1}^{n-1} |p_i - p_{i-1}|}{n^2 \times (\text{globaltime} - \text{timestamp}(c))} \quad (1)$$

where  $p_0, \dots, p_{n-1}$  are the page numbers ordered by their time of entering the cluster. The algorithm tends to assign large clusters a lower priority for two reasons: 1. Flash disks are efficient in writing such clustered pages. 2. The pages in a large cluster have a higher probability of being sequentially accessed.

The sum in the dividend in Formula [1](#), called *inter-page distance* (IPD), is used to distinguish between randomly accessed clusters and sequentially accessed clusters (clusters with only one page are set to 1). We prefer to keep a randomly accessed cluster in the buffer for a longer time than a sequentially accessed cluster. For example, a cluster with pages  $\{0, 1, 2, 3\}$  has an IPD of 3, while a cluster with pages  $\{7, 5, 4, 6\}$  has an IPD of 5.

The purpose of the time component in Formula [1](#) is to prevent randomly, but rarely accessed small clusters from staying in the buffer forever. The cluster timestamp  $\text{timestamp}(c)$  is the value of  $\text{globaltime}$  at the time of its creation. Each time a dirty page is inserted into the priority queue ( $\min(W)$  is dirty),  $\text{globaltime}$  is incremented by 1.

If a page in  $P$  is hit, it will be promoted to  $W$ . A newly fetched page in  $P$  will also be promoted to  $W$ . In both cases, page  $\min(W)$  is determined by  $W$ 's

victim selection policy and demoted to  $P$ . It does not have to be unfixed, because it is just moved inside the buffer. If  $\min(W)$  is clean, it is simply appended to the LRU list of clean pages. If it is dirty, the *globaltime* is incremented and we derive its cluster number and perform a hash lookup using this cluster number. If the cluster does not exist, a new cluster containing this page is created with the current *globaltime* and inserted to the priority queue. Furthermore, it is registered in the hash table. Otherwise, the page is added to the existing cluster tail and the cluster position in the priority queue is adjusted.

After demoting  $\min(W)$ , the page to be promoted, say  $p$ , will be removed from  $P$  and inserted to  $W$ . If  $p$  is dirty and its containing cluster  $c$  is not a victim cluster, we know that  $p$  is promoted due to a buffer hit. We update the cluster IPD including the timestamp. This will generally increase the cluster priority according to Formula 1 and cause  $c$  to stay in the buffer for a longer time. This effect is desirable, because the remaining pages in the cluster will probably be revisited soon due to locality. In contrast, the cluster timestamp is not updated, when pages are added to a cluster, because they are demoted from  $W$ .

The time complexity of CFDC depends on the complexity of the base algorithm in  $W$  and the complexity of the priority queue. The latter is  $O(\log m)$ , where  $m$  is the number of clusters. This should be acceptable due to  $m \ll \lambda \cdot B$ , where  $\lambda \cdot B$  is the number of pages in  $P$ . Furthermore, with the priority queue and its priority function, both temporal and spatial locality of the dirty pages are taken into account, thus potentially high hit ratios and improved runtime performance can be expected.

The CFDC approach implies the NoForce and Steal strategies for the logging&recovery component [13], which, however, is the standard solution in most DBMSs. In practice, page flushes are normally not coupled with the victim replacement process—most of them are performed by background threads. Obviously, these threads can benefit from CFDC’s dirty queue, where the dirty pages are already collected and ordered. The two-region scheme makes it easy to integrate CFDC with conventional replacement policies in existing systems.

## 4 Experiments

### 4.1 Hardware Environment

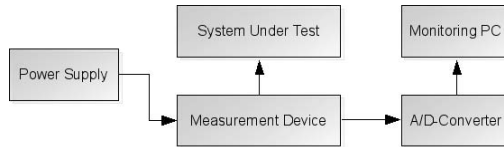
The system under test (SUT) has an Intel Core2 Duo processor and 2 GB of main memory. Both the OS (Ubuntu Linux with kernel version 2.6.31) and the DB engine are installed on an IDE magnetic disk (system disk). The test data (as a DB file) resides on a separate magnetic/flash disk (data disk). The data disks, listed in Tab. 1, are connected to the system one at a time.

A tailor-made power measurement device, consisting of ten voltage and current meters, connects the power supply and the system’s hardware components, as depicted in Fig. 1. The device does not tamper the voltages at the power lines, because the hardware components are sensitive to voltage variations. Instead, it

**Table 1.** Disk drives used in the test

name	device type	idle (W)	peak (W)	interface
HDD1	WD WD800AAJS 7200 RPM	5.3	6.3	SATA
HDD2	WD WD1500HLFS 10000 RPM	4.5	5.7	SATA
HDD3	Fujitsu MBA3147RC 15000 RPM	8.4	10.0	SAS
SSD1	SuperTalent FSD32GC35M	1.3	2.1	SATA
SSD2	MTRON MSP-SATA-7525-032	1.2	2.0	SATA
SSD3	Intel SSDSA2MH160G1GN	0.1	1.2	SATA

measures the current using current transformers with inductive measurement, and the voltage using voltage dividers on a shunt circuit. Both measurements are forwarded over a data bus to the A/D-Converter, which allows the signals being processed by a monitoring PC in real-time.

**Fig. 1.** Power measurement setup

Using this setup, we are able to precisely measure the energy consumption of the SUT’s major parts of interest: the data disk (denoted as SATA, although HDD3 is measured over the SAS power lines), the system disk (denoted as IDE), and the remaining components on the mainboard (denoted as ATX) including CPU and RAM. For a time period  $T$ , the average power  $\bar{P}$  is given by Formula 2:

$$\bar{P} = \frac{1}{T} \int_0^T (v(t) \cdot i(t)) dt \quad (2)$$

where  $v(t)$  and  $i(t)$  are the voltage and current as functions of time, and  $\int_0^T (v(t) \cdot i(t)) dt$  is the work, which is equal to the energy consumption  $E$ .

Tab. 2 lists the power profile of the major components of SUT. The idle column refers to the power values when the components are idle (0% utilization) but ready for serving requests, i.e., not in a service-unavailable state such as stand-by or hibernate, while the peak column refers to the power values when the components are under 100% utilization. An interesting observation can be made from this profile: ignoring the data disk, the idle power of the system is 57% of the peak power (25.9 W / 45.7 W). In other words, a lion’s share of the power is consumed only to keep the system in a ready-to-service state. Note this “idle share” is even larger in practice, because the peak power can only

**Table 2.** Power profile of SUT

power line	components	idle (W)	peak (W)
SATA/SAS	data disk	(see	Tab. <a href="#">II</a> )
IDE	system disk (Maxtor 6Y080P0)	6.3	12.2
ATX	CPU, RAM, and mainboard chips	19.6	33.5

be reached in some extreme conditions, where all the hardware components are fully-stressed at the same time. With the data disk, this observation still holds, because similar ratios exist between the idle and peak power of HDDs, while SSDs had too low power values to have a large impact on the idle/peak power ratio of the overall system. Furthermore, this observation is not specific to the test machine discussed here, because most state-of-the-art servers and desktop computers have similar idle/peak power ratios.

## 4.2 Software Settings

In all experiments, we use a native XML DBMS designed according to the classical *five-layer reference architecture*. For clarity and simplicity, we only focus on its bottom-most two layers, i.e., the *file manager* supporting page-oriented access to the data files, and the *buffer manager* serving page requests. Although designed for XML data management, the processing behavior of these layers is very close to that of a relational DBMS.

We deactivated the file-system prefetching and used *direct I/O* to access the DB file, so that the influences of file system and OS were minimized. The logging&recovery component is deactivated so that no extra I/Os for logging will influence our measurements. Our experiments are driven by two real-life OLTP page reference traces and only the two layers introduced were involved in the processing, thus the workload is I/O intensive. All experiments started with a *cold* DB buffer. Except for the native code responsible for direct I/O, the DB engine and the algorithms are completely implemented in Java. CFDC and competitor algorithms are fully integrated into the XML DBMS and work with other components of the DB engine.

We cross-compared five buffer algorithms, including the flash-aware algorithms CFLRU, LRU-WSR, REF, CFDC and the classical algorithms represented by LRU. The *block size* parameter of REF, which should correspond to the size of a flash block, was set to 16 pages (DB page size = 8 KB, flash block size = 128 KB). To be comparable, the *cluster size* of CFDC was set to 16 as well. The  $|VB|$  parameter of REF (the number of allowed victim blocks) was set to 4, based on the empirical studies of its authors. Furthermore, we used an improved version of CFLRU which is much more efficient at runtime yet functionally identical to the original algorithm.

### 4.3 Measuring Spatial Locality

Both CFDC and REF improves the spatial locality of page flushes. We define the metric *cluster-switch count* ( $CSC$ ) to quantify the spatial locality of page flushes. Let  $S := \{q_0, q_1, \dots, q_{m-1}\}$  be the sequence of page flushes, the metric  $CSC(S)$  reflects the spatial locality of  $S$ :

$$CSC(S) = \sum_{i=0}^{m-1} \begin{cases} 0, & \text{if } q_{i-1} \text{ exists and in the same cluster as } q_i \\ 1, & \text{otherwise} \end{cases} \quad (3)$$

The *clustered writes* of CFDC are write patterns with high spatial locality and thus minimized cluster-switch counts. Sequential writes are a special case of clustered writes, where pages are updated in a forward or reverse order according to their locations on the storage device. If  $d(S)$  is the set of distinct clusters addressed by  $S$  and  $S$  is a sequential access pattern, we have  $CSC(S) = |d(S)|$ . In the context of magnetic disks, if we set the cluster size equal to the track size, then  $CSC(S)$  approximates the number of disk seeks necessary to serve  $S$ .

Compared to CFDC, the sequence of dirty pages evicted by the algorithm REF generally has a much higher  $CSC$ , because it selects victim pages from a *set* of victim blocks and the victim blocks can be addressed in any order. However, this kind of write requests can also be efficiently handled by flash disks, if the parameter  $VB$  is properly set. Because the sequence of dirty pages evicted can be viewed as multiple sequences of clustered writes that are interleaved with one another, we call the approach of REF *semi-clustered writes*.

Let  $R := \{p_0, p_1, \dots, p_{n-1}\}$  be the sequence of page requests fed to the buffer manager, we further define the metric *cluster-switch factor* ( $CSF$ ) as:

$$CSF(R, S) = CSC(S)/CSC(R) \quad (4)$$

$CSF$  reflects the efficiency to perform clustering for the given input  $R$ . To compare identical input sequences, it is sufficient to consider the  $CSC$  metric alone.

### 4.4 The TPC-C Trace

The first OLTP trace was obtained using the PostgreSQL DBMS. Our code integrated into its buffer manager recorded the buffer reference string of a 20-minutes TPC-C workload with a scaling factor of 50 warehouses.

We ran the trace for each of the five algorithms and repeat this on each of the devices listed in Tab. 1. The recorded execution times and energy consumptions are shown in Fig. 2. Since our workload is I/O-intensive, the device performance has a strong impact on the overall system performance, e.g., SSD3 reduced the average runtime (average over the algorithms) by a factor of 21 compared with HDD1 and by a factor of 19 compared with SSD1, while the corresponding energy-saving factors are 25 and 19 respectively. The CFDC algorithm had the best performance on all of the devices, with a maximum performance gain of 22% over CFLRU on SSD1. Interestingly, even on the magnetic disks, CFDC

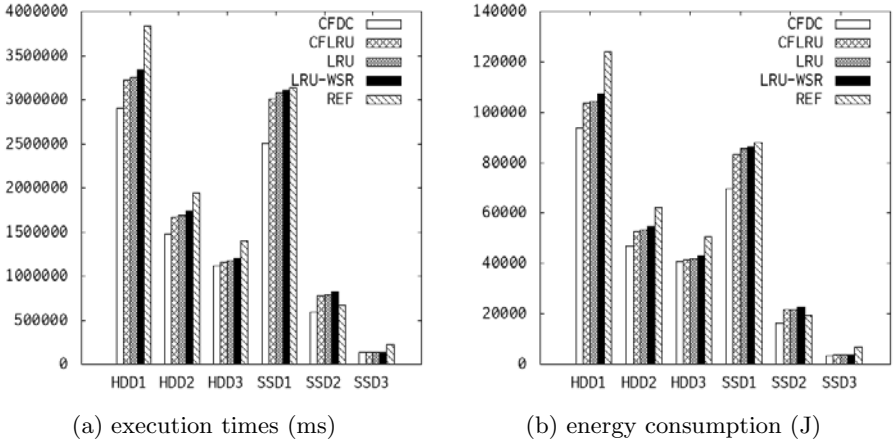
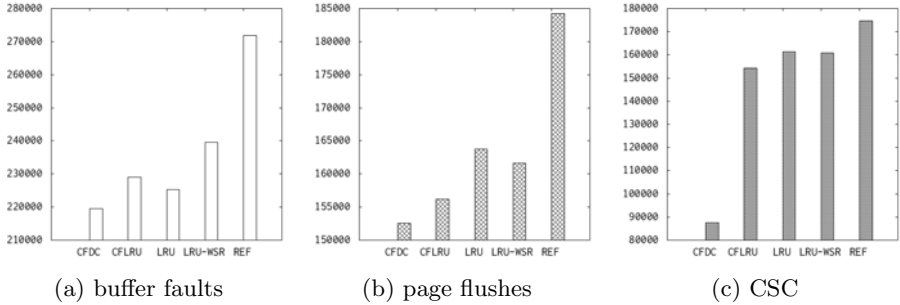


Fig. 2. Performance and energy consumption running the TPC-C trace

and CFLRU had a better performance than LRU, which, in turn, outperformed LRU-WSR even on the flash disks. In most configurations, REF had the longest execution times due to its lower hit ratio and higher number of page flushes, with the exception on SSD2, where its semi-clustered writes seems to be best accommodated by that specific device.

Similar to magnetic disks, it is common for flash disks to be equipped with a device cache. Very often, it can not be deactivated or re-sized by the user or the OS, and its size is often undocumented. Obviously, the DB buffer in our experiment should be larger than the device caches. Otherwise, the effect of the DB buffer would be hidden by them. On the other hand, if the DB buffer is too large, the difference between our algorithms would be hidden as well, since with a large-enough buffer, there would be hardly any I/O. Based on these considerations, we used a buffer size of 8000 pages (64 MB) for this experiment, because the largest known device cache size is 16 MB of HDD3. The difference between the execution times of the algorithms becomes smaller on SSD3 (see Fig. 2) due to two reasons: 1. The I/O cost on SSD3 is much smaller than on other devices, yielding the buffer layer optimization less significant; 2. This device has supposedly the largest device cache, since it is the newest product among the devices tested.

Fig. 3 shows the performance metrics that were constant across all the devices (device-independent): number of buffer faults (physical reads), number of page flushes (physical writes), and cluster-switch count (CSC). As shown in Fig. 3a and Fig. 3b, the clean-first algorithms (CFLRU and LRU-WSR) traded some of their hit ratios for a reduced page flush count, while REF suffered both from a higher number of buffer faults and a higher number of page flushes. Among all algorithms compared, the page flushes generated by CFDC had the highest spatial locality (Fig. 3c).



**Fig. 3.** Device-independent performance metrics

Comparing Fig. 2a with Fig. 2b, we can see a strong correlation between the execution times and the energy consumption. In particular, the best-performing algorithm was also the most energy-saving algorithm. For example, CFDC reduced energy consumption by 32% on HDD1 and by 71% on SSD3 compared with REF. This effect is further explained by Fig. 4, which contains a breakdown of the average working power of major hardware components of interest, compared with their idle power values. The figures shown for the configurations HDD1 and SSD1 are indicative<sup>2</sup>. Ideally, the power consumption of a component (and the system) should be determined by its utilization. But for both configurations, there is no significant power variation when the system state changes from idle to working. Furthermore, no clear difference can be observed between the various algorithms, although they have different complexities and, in fact, also generate different I/O patterns. This is due to the fact that, independent of the workload, the processor and the other units of the mainboard consume most of the power (the ATX part in the figure) and these components are not *energy-proportional*, i.e., their power is not proportional to the system utilization caused by the workload. However, due to the missing energy-proportional behavior of most system components, the elapsed time  $T$  of the workload (algorithm) almost completely determines its energy consumption  $E$  (note,  $E = P \cdot T$ ).

#### 4.5 The Bank Trace

The second trace used here is a one-hour page reference trace of the production OLTP system of a Bank. It was also used in the experiments of [14] and [15]. This trace contains 607,390 references to 8-KB pages in a DB having a size of 22 GB, addressing 51,870 distinct page numbers. About 23% of the references update the page requested, i.e., the workload is read-intensive. Even for the update references, the pages must be first present in the buffer, thus more reads are required. Moreover, this trace exhibits an extremely high access skew, e.g., 40% of the references access only 3% of the DB pages that were accessed in

<sup>2</sup> They are similar for the other device types (IDE and ATX remain constant). For this reason, we have omitted the other configurations due to space limitations.



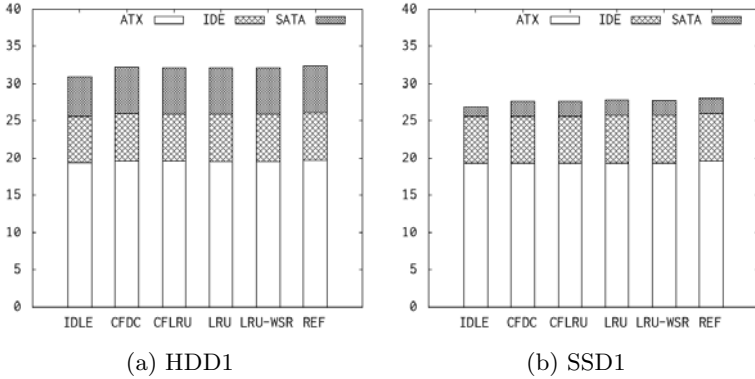


Fig. 4. Break-down of average power (W)

the trace [14]. We use this trace to examine the impact of buffer size on the performance and energy consumption of our algorithms. Hence, the buffer size varied from 500 to 16000 pages (factor 32).

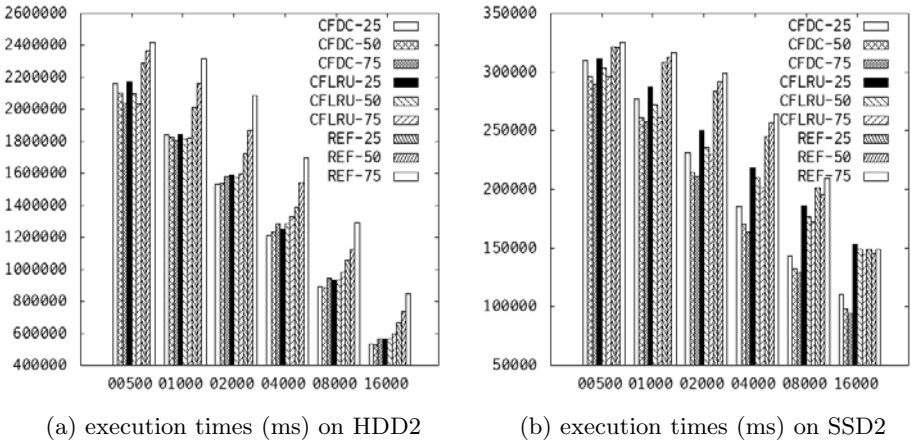
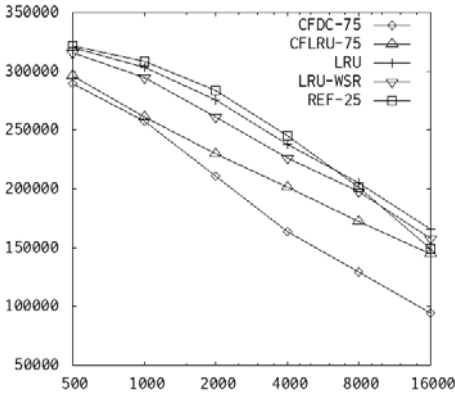
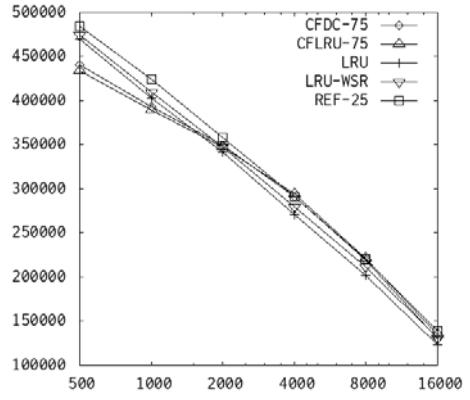


Fig. 5. Impact of window size

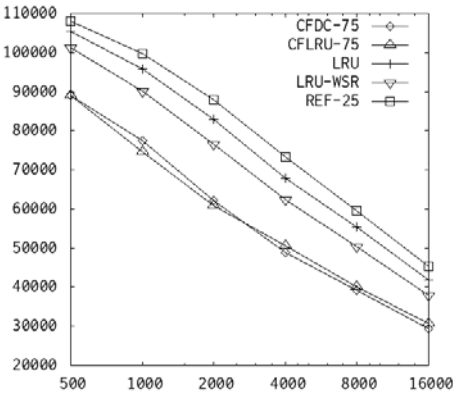
In the experiments discussed in Sect. 4.4, the parameter “window size” was *not* tuned—it was set to 0.5 for all related algorithms. To examine its impact, for each of the algorithms CFDC, CFLRU, and REF, we always ran the trace three times with the window size parameter set to 0.25, 0.50, and 0.75 respectively, denoted as REF-25, REF-50, REF-75, etc. As an indicative example, the execution times measured on the middle-class devices HDD2 and SSD2 are shown in Fig. 5. We found that the optimal window size depends on both the buffer size and the device characteristics, for all the three related algorithms.



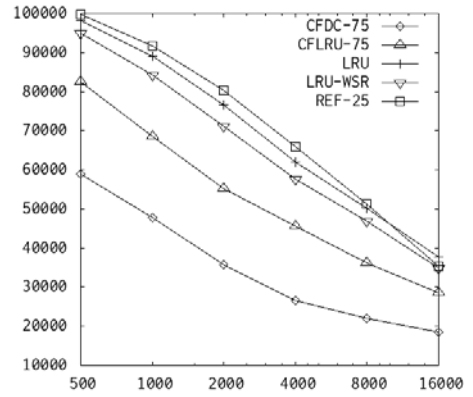
(a) execution times (ms)



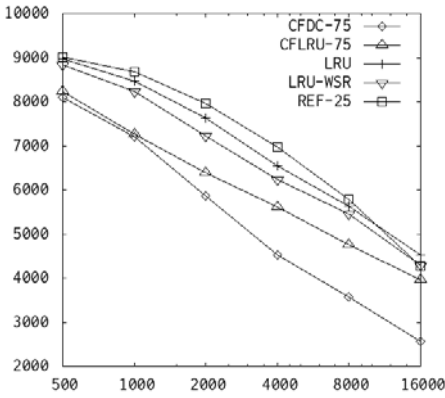
(b) page faults



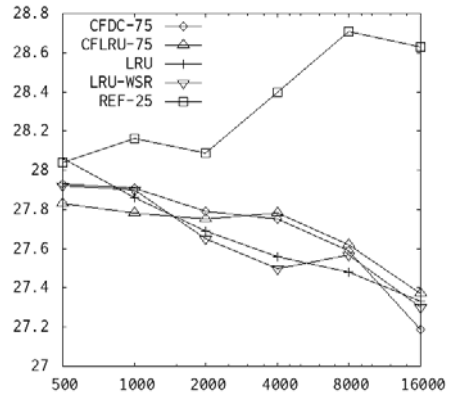
(c) page flushes



(d) CSC



(e) energy (J)



(f) power (W)

**Fig. 6.** Performance and power running the bank trace on SSD2

Due to the space limitation, we only discuss the performance and power figures measured on SSD2 in more detail. For the same reason, we choose one best-performing window-size configuration for each of the related algorithms and compare them with LRU and LRU-WSR in Fig. 6. For CFDC and CFLRU it was 0.75, and for REF it was 0.25. The performance figures shown in Fig. 6a are explained by the device-independent metrics shown in Fig. 6b to Fig. 6d. For example, CFDC and CFLRU had comparable numbers of buffer faults and page flushes, but the latter suffered from a lower spatial locality of page flushes (Fig. 6d). The access skew is reflected in Fig. 6b: the difference in number of page faults becomes insignificant beyond 4000 pages.

The linear complexity of REF resulted in a higher CPU load compared with other algorithms having constant complexity. This is captured by Fig. 6f, where the working power of the system is illustrated. The power value of REF goes up with an increasing buffer size, while the power values caused by the other algorithms slightly decrease, because the larger the buffer sizes the more physical I/Os were saved. Note, handling a logical I/O from the buffer is more energy-efficient than doing a physical I/O, because fewer CPU cycles are required and device operations are not involved.

The algorithms' complexity did have an impact on the power of the system, but the time factor had a stronger impact on the overall system energy consumption. For example, enlarging the buffer from 500 to 16000 pages augmented the power value of REF by 2.1% (from 28.04 W to 28.63 W), while the corresponding execution time (Fig. 6a) decreased by 46.3%. In fact, the effect of the increased CPU load was hidden by the "idle share" of the working power (27.08 W, not shown in the figure). Therefore, the energy consumption curves in Fig. 6e largely mirror the performance curves of Fig. 6a. In particular, at a buffer size of 16000 pages, the relative performance gain of CFDC over CFLRU is 54%, while the corresponding energy saving is 55%.

## 5 Conclusions

Our experiments reveal a great performance potential of flash disks. The use of flash-aware buffer algorithms can further significantly improve system performance on these devices. The CFDC algorithm clearly outperformed the other algorithms in most settings. According to our device sensitivity study, its flash-specific optimizations do not exclude its application in systems based on magnetic disks.

The performance gain can be translated into a significant energy-saving potential due to the strong correlation between performance and system energy consumption. Furthermore, faster I/O reduces the overall system runtime and leaves more opportunities for the system to go into deeper energy-saving states, e.g., stand-by or even off-line.

As an important future task of hardware designers and device manufacturers, all system components other than flash disks should be developed towards stronger energy-proportional behavior. Then, the speed or runtime reduction

gained by flash use for I/O-intensive applications could be directly translated into further substantial energy saving. As a consequence, energy efficiency due to flash disk use would be greatly enhanced as compared to magnetic disks.

## References

1. Gray, J., Fitzgerald, B.: Flash disk opportunity for server applications. *ACM Queue* 6(4), 18–23 (2008)
2. Woodhouse, D.: JFFS: the journalling flash file system. In: *The Ottawa Linux Symp.* (2001)
3. Bouganim, L., et al.: uFLIP: Understanding flash IO patterns. In: *CIDR* (2009)
4. Härder, T., et al.: Towards flash disk use in databases—keeping performance while saving energy? In: *BTW. LNI*, vol. 144, pp. 167–186 (2009)
5. Effelsberg, W., Härder, T.: Principles of database buffer management. *ACM TODS* 9(4), 560–595 (1984)
6. Corbato, F.J.: A paging experiment with the multics system. In: *In Honor of Philip M. Morse*, p. 217. MIT Press, Cambridge (1969)
7. Tanenbaum, A.S.: *Operating Systems, Design and Impl.* Prentice-Hall, Englewood Cliffs (1987)
8. Park, S., et al.: CFLRU: a replacement algorithm for flash memory. In: *CASES*, pp. 234–241 (2006)
9. Jung, H., et al.: LRU-WSR: integration of LRU and writes sequence reordering for flash memory. *Trans. on Cons. Electr.* 54(3), 1215–1223 (2008)
10. Seo, D., Shin, D.: Recently-evicted-first buffer replacement policy for flash storage devices. *Trans. on Cons. Electr.* 54(3), 1228–1235 (2008)
11. Ou, Y., et al.: CFDC: a flash-aware replacement policy for database buffer management. In: *DaMoN*, Providence, RI, pp. 15–20. ACM Press, New York (2009)
12. Gray, J., Reuter, A.: *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, San Francisco (1993)
13. Härder, T., Reuter, A.: Principles of transaction-oriented database recovery. *ACM Computing Surveys* 15(4), 287–317 (1983)
14. O’Neil, E.J., et al.: The LRU-K page replacement algorithm for database disk buffering. In: *SIGMOD*, pp. 297–306 (1993)
15. Megiddo, N., Modha, D.S.: ARC: A self-tuning, low overhead replacement cache. In: *FAST, USENIX* (2003)

# Towards Efficient Concurrent Scans on Flash Disks\*

Chang Xu, Lidan Shou, Gang Chen, Wei Hu, Tianlei Hu, and Ke Chen

College of Computer Science, Zhejiang University, China  
chinawraith@163.com, {should, cg, hw, ht1, chenk}@zju.edu.cn

**Abstract.** Flash disk, also known as Solid State Disk (SSD), is widely considered by the database community as a next-generation storage media which will completely or to a large extent replace magnetic disk in data-intensive applications. However, the vast differences on the I/O characteristics between SSD and magnetic disk imply that a considerable part of the existing database techniques need to be modified to gain the best efficiency on flash storage. In this paper, we study the problem of large-scale concurrent disk scans which are frequently used in the decision support systems. We demonstrate that the conventional sharing techniques of multiple concurrent scans are not suitable for SSDs as they are designed to exploit the characteristics of hard disk drivers (HDD). To leverage the fast random reads on SSD, we propose a new framework named Semi-Sharing Scan ( $S^3$ ) in this paper.  $S^3$  shares the readings between scans of similar speeds to save the bandwidth utilization. Meanwhile, it compensates the faster scans by executing random I/O requests separately, in order to reduce single scan latency. By utilizing techniques called group splitting and I/O scheduling,  $S^3$  aims at achieving good performance for concurrent scans on various workloads. We implement the  $S^3$  framework on a PostgreSQL database deployed on an enterprise SSD drive. Experiments demonstrate that  $S^3$  outperforms the conventional schemes in both bandwidth utilization and single scan latency.

## 1 Introduction

During the past decade, flash memory has become a popular medium for storage due to its fast random access, low energy consumption, shock resistance, and silent working. In recent years, the flash disk, also known as Solid State Disk (SSD), has gained momentum in its competition against the conventional magnetic hard disk drive (HDD) in the market. Many software products which previously rely on HDD storage are now being considered to adopt SSD as an alternative. Among these, the database management systems are probably within

---

\* This work was supported in part by the National Science Foundation of China (NSFC Grant No. 60803003, 60970124) and by Chang-Jiang Scholars and Innovative Research Grant (IRT0652) at Zhejiang University.

the class which requires the most attention and efforts, due to their vast complexity and numerous legacy applications. Despite the challenges, the prospect of replacing HDD with SSD in DBMS is attractive as the current database applications are desperately confined by the bottleneck of random I/O in HDDs. The most desirable feature of SSD in this regard is its elimination of seek and rotational delay, and the resultant fast random accesses. Table 1 indicates that the state-of-the-art SSD outperforms the conventional HDDs by more than two orders of magnitude in random reads.

**Table 1.** Performance Comparison of Different Read Patterns

	HDD †	SSD‡
Random Read Latency	9.69ms	0.03ms
Sequential Read Rate	114.7MB/s	up to 250MB/s

† : Western Digital 7200 rpm Digital Black

‡ : Intel X-25 Extremely

Unfortunately, previous studies have found that the performance of DBMS could not gain the expected improvements while they are deployed on SSDs straightforwardly [8] [14] [15]. The main reason for such results is that the storage engines and access methods of the conventional DBMSs are designed for the HDDs. As such, the advantages of SSDs cannot be fully exploited.

Let us consider the disk-based methods to handle *concurrent disk scans*, which are often executed as table scans in many decision support systems. The problem of concurrent table scans on HDD can be described as following: Given a limited buffer size of  $C$  and a sufficiently large table, if a running scan operation  $S_1$  *outdistances* another scan operation  $S_2$ , meaning that  $S_1$  is accessing a page which is at least  $C + 1$  blocks ahead of  $S_2$ ,  $S_2$  is unlikely to hit any useful pages in the buffer with typical buffer management policies [9] [10] [11] [12] [13]. As a result, scan  $S_2$  will not only cause disk thrashing but also compete the buffer space with  $S_1$ , leading to poor performance. In the literature, a few methods [1] [2] [4] [6] [7] have been proposed to address the above problem on HDDs. The main idea behind all these methods is that concurrent disk scans can “share” their footprints on disk pages as well as their memory buffer space. Therefore, multiple scans can share a (hopefully long) period of sequential disk-head movements, avoiding the penalty of chaotic random reads. However, these methods have the following drawbacks when employed on SSDs:

- First, there are almost always speed mismatches among multiple scans sharing the same sequential accesses. A faster scan has to be constrained by the slower ones in order to share the disk arm movement on the HDD. But on SSD this constraint is trivial as it is not mechanic-driven. Therefore, the capacity of SSD does not be fully exploited if the overall disk bandwidth is not saturated.

- Second, the existing methods focus on optimizing the disk accesses only because the queries are I/O-bounded. As SSD provides much larger bandwidth than HDD, some complicated queries are bounded by CPU power other than I/O bandwidth. Therefore, the strategy of the disk accesses needs to be re-considered to exploit the system capability.

In this work, we leverage the advantage of SSD to speed up concurrent scans. First we introduce a cost model for concurrent scans on SSD, and then we propose a novel framework named *Semi-Sharing Scans* ( $S^3$ ).  $S^3$  consists of two main components, namely the *scan scheduler* and the low-level *I/O scheduler*. (1) The *scan scheduler* clusters multiple scans into a set of groups by the similarity between their speeds. The scans in a same group share a stream of sequential block read operations which we refer to as the *unified I/Os*. In addition, faster scans which are not fully feed are allowed to perform *complementary I/Os* at other disk addresses. (2) The *I/O scheduler* dispatches the I/O requests from the scans into two separate queues, namely the *unified* and *complementary* I/O queue. During I/O scheduling, the unified queue enjoys higher priority compared to the complementary one.

Compared to the existing methods,  $S^3$  is more suitable for concurrent scans on SSD as it optimizes both the bandwidth utilization and the average latency of each single scan under various workloads. Additionally, the framework is easy to implement, as only limited modifications are needed in a RDBMS. The contributions of our work include:

- We present a new cost model of the concurrent scans on SSD and demonstrate that the state-of-the-art scan scheduling policies do not fully exploit the I/O capability of SSD in various workloads.
- We propose a multiple-scan scheduling framework  $S^3$  based on the cost analysis. This framework improves the bandwidth utilization via grouping and I/O scheduling. Compared to the existing methods,  $S^3$  reduces the latency of a faster scan by compensating it with a separate stream of disk footprints.
- We implement the  $S^3$  framework in the PostgreSQL DBMS. Our comprehensive experiments based on the *TPCH* benchmark indicate that the proposed approach is very effective for various workload patterns.

The rest of the paper is organized as follows. In Section 2 we define the problem and analyze the existing strategies on SSD. Our  $S^3$  framework is described in Section 3. The experiments in Section 4 demonstrate the efficiency of  $S^3$  and we conclude our work in Section 5.

## 2 Problem Definition and Cost Analysis

### 2.1 Problem Definition

A formal definition to concurrent scans problem is as follows. Assume that a table contains  $N$  pages stored in the secondary storage and a small buffer with capacity  $C$  ( $C$  is much smaller than  $N$ ) in the main memory. If there are  $k$  scans

denoted by  $S_1, S_2, \dots, S_k$ , whose speeds are  $v_1, v_2, \dots, v_k$  respectively, all beginning at arbitrary times and having to access all the  $N$  pages of the table. The problem is how to coordinate the scans to gain the optimal performance.

The performance optimization of concurrent scans relies on the definitions of a few terms as following. The speed  $v_i$  of a scan  $S_i$  is defined as the number of pages that can be consumed per second when  $S_i$  is executed alone. Meanwhile, the response time of  $S_i$ , denoted by  $rt(S_i)$ , is the elapsed time between its start and end. Given sufficiently large disk bandwidth and a specific CPU processing power, the minimum response time of  $S_i$ , denoted by  $rt_{min}(S_i)$ , is the response time when  $S_i$  is executed alone. Therefore we have  $rt_{min}(S_i) = N/v_i$ . As the computational costs between queries may differ significantly, the speeds of different scans may also be very different. For example, in the *TPCH* benchmark,  $Q_1$  generates a slow scan on the *Lineitem* table as it needs many arithmetic computations, while  $Q_6$  is much faster as it only contains an aggregation.

A set of concurrent scans  $S_i$  ( $i = 1, 2, \dots, n$ ) may share disk I/Os if their I/O requests are scheduled properly. In such case, the overall number of pages being transferred per second is defined as the *bandwidth consumption*. However, in real world the bandwidth consumption is always constrained by the physical device bandwidth capacity  $V$ . Therefore, we refer to the unconstrained, ideal bandwidth consumption as the *bandwidth demand* or  $B_{demand}$ , while the constrained, actual bandwidth consumption seen on the physical disk interface as the *actual bandwidth consumption* or  $B_{actual}$ . We denote by  $B_{demand}(system)$  the total bandwidth demand of the  $n$  scans in the system, and by  $B_{demand}(s_i)$  the demand of scan  $s_i$ .  $B_{actual}(system)$  and  $B_{actual}(s_i)$  are defined similarly. We note that  $B_{actual}(system)$  must always be no more than  $B_{demand}(system)$ . The latter must also be no more than the sum of all the speeds of the scans. Therefore, we have

$$B_{actual} \leq B_{demand} \leq \sum v_i.$$

The target of our optimization is to minimize the average response time

$$\frac{1}{n} \sum_{i=1}^n rt(S_i) \quad (1)$$

while  $B_{actual}$  is constrained by the device bandwidth capacity. It is important to note that the  $B_{actual}(system)$  is either  $V$  or  $B_{demand}(system)$  whichever is smaller.

## 2.2 Existing Scheduling Schemes

In this subsection, we shall look at a few existing schemes for handling concurrent scans. A naive scheme to handle concurrent scans is to simply rely on the LRU buffer replacement policy. The naive scheme does not allow any page reusing when the distance between the footprints of two concurrent scans is greater than  $C$  pages. For clearness we name it as “no share”. In such case, each scan needs to read from the secondary storage for every I/O request. This may lead



to arm “thrashing” on HDDs, causing serious performance degradation. Using the MRU replacement policy might be slightly better than the LRU because it could reuse some initial pages in the buffer. However, the gain is very limited as  $C$  is usually much smaller than  $N$ .

A better scheme used by some DBMSs [1] [2] [3] [4] is to save I/O by sharing a stream of reads among multiple scans. When a new scan arrives, the system tries to attach it to an existing scan, which shares its reads of the rest pages with the new one. After a shared scan reaches the end of the table, it resumes from the beginning of the table until arriving at the original starting location. The main problem of this *shared* policy is that it cannot handle the speed mismatch between the scans. Once a scan outdistances another for more than  $C$  pages, the situation degenerates to the *no share* scheme. A common solution, namely “*strict share*”, is to stall the fast scans, thus guaranteeing the sequential movement of the disk arm. In this case, the  $rt(S_n)$  is between  $[N/v_{slowest}, N/v_n]$ . To relieve the performance loss of the fast scans, DB2 [6] [7] proposes an improved “*group shared*” scheme in which the scans with similar speeds are grouped together and sharing happens within each group. However, the faster scans still need to wait for the slower ones in a same group.

A novel approach presented in [5] suggests to schedule the buffer by a suite of relevance-intensive functions. However, the computation of the relevance requires subtle dynamic statistics of the table and each scan. This would burden the CPU when the table is large. Another drawback is that it roughens the granularity of the buffer management, which makes it incoincident with accesses of other pattern.

### 2.3 Cost of Concurrent Scans on SSD

As SSD is non-mechanic driven and offers significantly higher random access speed compared to the HDD, the seek time and rotational delay are both eliminated. Therefore, when the DBMS is deployed on SSDs, the cost of multiple scans must be computed differently. In this subsection, we evaluate the scheduling schemes described in subsection 2.2 using the cost model for SSD. Then we demonstrate that none of the existing scheduling schemes for concurrent scans can adequately exploit the capacity of SSD.

Some assumptions are necessary for introducing the cost model of SSD. (1) We assume that we are equipped with sufficiently large number of CPUs regarding the I/O capacity so that the scans only sleep on I/O requests. (2) We observe that the bandwidth of a SSD device can linearly scale-up before its actual bandwidth consumption reaches the device capacity. Before that, the actual bandwidth being consumed should be equal to the sum of the demands of all scans. That means  $B_{actual}(system) = B_{demand}(system) = \sum v_i$  when  $B_{demand}(system) < V$ . (3) When the bandwidth demand of the whole system  $B_{demand}(system)$  is greater than the physical disk capacity  $V$ , indicating that the I/O subsystem is overloaded, some requests have to wait for others to complete. Such waiting causes performance degradation to a scan in terms of speed (bandwidth) loss. Without loss of generality, we assume that such speed loss is uniformly distributed among all scans.

To minimize the average response time proposed in Equation 11, we look at the response time of each scan  $s_k$  ( $k = 1, \dots, n$ ) under different scheduling schemes. Based on our second assumption of SSD, we have

$$B_{actual}(system) = \min(V, B_{demand}(system)).$$

For the *no share* scheme,  $B_{demand}(system)$  is  $\Sigma v_i$ . If  $B_{demand}(system)$  is less than  $V$ , each scan could achieve  $rt_{min}$  as no I/O request would be waiting. Otherwise, the actual bandwidth of  $s_k$ ,  $B_{actual}(s_k)$ , is only  $V * v_k / \Sigma v_i$ . Therefore we have  $rt(s_k) = N/V * v_k / \Sigma v_i$ . For the *strict shared* policy,  $B_{demand}(system)$  is equal to the speed of the slowest scan, denoted by  $v^s$ . Therefore the response time of each scan is  $N/v^s$  if  $v^s < V$ , and  $N/V$  if  $v^s > V$ . For the *group shared* scheme, the  $B_{demand}$  of each group  $g$ , denoted by  $B_{demand}(g)$ , is the slowest speed within group  $g$  (we denote it by  $v^{sg}$ ) and  $B_{demand}(system)$  is the sum of all groups in the system, namely  $\Sigma v_i^{sg}$ . If  $\Sigma v_i^{sg} < V$ , then the response time of  $s_k$  is given by the slowest scan in its group, namely  $rt(s_k) = N/v_k^{sg}$ . Otherwise,  $rt(s_k) = N/V * v_k^{sg} / \Sigma v_i^{sg}$ .

**Table 2.** Response times of scans on SSD under existing schemes.  $v^s$  means the slowest speed among all scans, while  $v^{sg}$  means the slowest speed in its group.

	$rt(s_k)$	$B_{actual}(system)$
no share	$N/\min(v_k, V * v_k / \Sigma v_i)$	$\min(V, \Sigma v_i)$
strict share	$N/\min(v^s, V)$	$\min(v^s, V)$
group share	$N/\min(v_k^{sg}, V * v_k^{sg} / \Sigma v_i^{sg})$	$\min(V, \Sigma v_i^{sg})$

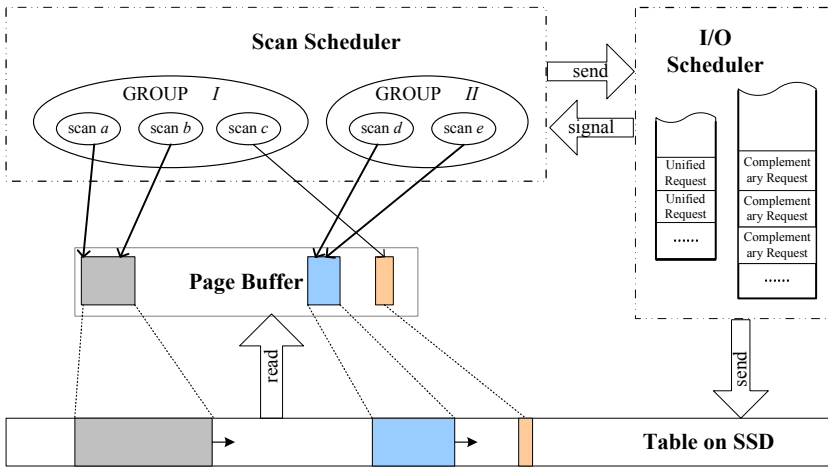
Table 2 implies that all existing schemes have their limitations on the SSDs. The *no share* scheme could achieve the minimal average response time when  $B_{demand}(system) < V$ . However, this benefit comes at the expense of large bandwidth consumption when the number of the scans increases. The performance of the *no share* scheme is likely to plunge when  $B_{demand}(system)$  exceeds  $V$ . In contrast, the *strict shared* scheme aims at minimizing  $B_{demand}(system)$ . However, faster scans in the system have to be confined by the slower ones. When the disk has available bandwidth, the *strict shared* scheme is unable to fully exploit it to expedite the fast scans. The *group share* scheme compromises between these two schemes. But it still does not make adequate use of the available bandwidths to expedite the faster scans within each group.

According to the analysis above, we design a new framework  $S^3$  which could fully exploit the capacity of the SSD by combining the sharing I/O and separated I/O.  $S^3$  shares part of the I/O reading among similar scans for reducing the bandwidth consumption, while it compensates the faster ones in random I/Os to improve the response time. Compare to the conventional sharing policies,  $S^3$  could gain in both query latencies and bandwidth consumption.

### 3 The Semi-Sharing Scans Framework

#### 3.1 Overview

The  $S^3$  framework consists of two components, depicted in Figure 1. The *Scan Scheduler* coordinates the multiple scans, possibly in groups, and the *I/O Scheduler* dispatches the low-level I/O requests. Each scan consumes the pages of the table via a common buffer. The buffer is located in the main memory and caches part of the data pages from secondary storage. Once a new page is read in, the buffer manager evicts a victim page according to its priority mechanism, such as *LRU*, *MRU*, *Clock Sweep* etc.



**Fig. 1.** An Overview of  $S^3$ . Two scan groups  $I$  and  $II$  are in progress. In group  $I$ , scan  $c$  acquires for a page via a *complementary* I/O request, while the group  $I$  moves forward via a *unified* I/O request.

The *Scan Scheduler* manages a number of scan groups, each of which contains one or several scans. To control the memory footprint, each group is allocated a fix-sized window in the buffer, which contains a number of contiguous data pages of the table. The scans in a same group share the reads of the pages in the window, which moves forward on the table via the cooperation from all the scans in the group. Specifically, the fastest scan reads the next data page and pins it in the memory until the slowest one consumes it. This read is translated to a *unified* I/O request in the low-level. We note that the *scan scheduler* can be easily implemented on the buffer module of any existing DBMSs.

If the fastest scan outdistances the slowest one by the size of the window, it cannot proceed sequentially to the next disk address. In this case we compensate it by allowing complementary I/Os outside the shared stream of disk addresses, by sending *complementary* I/O requests. For example in Figure 1, scan  $c$  is

restricted by the slowest scan  $a$ , so  $c$  can make a complementary I/O request, which deviates from its conventional sequence.

All I/O requests in  $S^3$  are scheduled by the *I/O Scheduler*. The *unified* I/O requests and the *complementary* I/O requests are stored and manipulated in separate queues. The unified requests are given higher priority. Therefore, if the unified request queue is not empty, no complementary requests will be processed. Once a request is completed, the *I/O Scheduler* evokes the scans which are waiting on it.

In the rest of this section, we shall look at the detailed techniques of scan scheduling and I/O scheduling.

### 3.2 Scheduling the Scans

Like the typical table scan processes in most existing DBMSs, each scan in the  $S^3$  framework consists of three main steps, namely (1) *BeginScan*, (2) *FetchNext*, (3) *EndScan*. In the *BeginScan* step, a new scan is initialized. Then the scan traverses the table from the first page to the end by invoking the *FetchNext* step iteratively. After all the pages are processed, the *EndScan* step terminates the scan. We shall now elaborate these steps in the following.

#### Algorithm 1. BeginScan( $S$ )

```

Open( $S$ );
 $S.loc_{begin} \leftarrow 0$ ;
if there exists no groups then
  create a new group  $g$ ;
  add  $S$  to  $g$ 
else
   $g \leftarrow ChooseGroup()$ ;
   $S.loc_{begin} \leftarrow MAX$ ;
  for each  $S_i$  in  $g$  do
    if  $S_i.loc_{cur} < S.loc_{begin}$ 
    then
       $S.loc_{begin} \leftarrow S_i.loc_{cur}$ 
  add  $S$  to  $g$ ;
  if CheckSplit( $g$ ) then
    Split  $g$ ;
   $S.loc_{cur} \leftarrow S.loc_{begin} + 1$ 

```

#### Algorithm 2. EndScan( $S$ )

```

if  $S$  is the only scan in  $g$  then
  delete  $g$ ;
  for all groups  $G_N$  do
     $g_i \leftarrow PickSplitGroup(G_N)$ 
    Split  $g_i$ ;
else
  remove  $S$  from  $g$ ;
Close( $S$ );

```

#### Algorithm 3. FetchNext( $S$ )

```

if  $S.loc_{cur} = S.loc_{begin}$  then
  return NULL
while true do
   $loc_{acq} \leftarrow S.loc_{cur}$ ;
   $type \leftarrow unified$ ;
   $g \leftarrow S$ 's group;
  for each  $S_i$  in  $g$  do
    if  $S.loc_{cur} - S_i.loc_{cur} > |g.Window|$  then
       $loc_{acq} \leftarrow S.loc_{begin}$ ;
       $type \leftarrow complementary$ 
  if  $type = unified$  then
    if  $loc_{acq}$  is in buffer then
       $S.loc_{cur} ++$ ;
      return  $loc_{acq}$ ;
    else
      SendRequest( $loc_{acq}, S$ );
      Signal(I/O Scheduler);
  else
    if  $loc_{acq}$  is in buffer then
       $S.loc_{begin} --$ ;
      return  $loc_{acq}$ ;
    else
      SendRequest( $loc_{acq}, S$ );
      Signal(I/O Scheduler);
Wait();

```

The process of *BeginScan* in  $S^3$  is described in Algorithm 1. When a new scan  $S$  begins, the scan scheduler creates a new group if there is no group at all. Otherwise,  $S$  is added to an existing group according to its speed, which is estimated by the query optimizer module. Then the scan scheduler checks whether a splitting is necessary. The detail of choosing and splitting the group will be described in the next subsection. Once a group is chosen, the beginning location of  $S$  is determined. For simplicity, we choose the page being processed by the slowest scan in the group as the beginning location of  $S$ .

The *FetchNext* step returns one page of the table each time. The process of *FetchNext* is described in Algorithm 3. To synchronize with other scans in the group, A scan  $S$  firstly checks its current location. If  $S$  has outdistanced the slowest one in its group for more than the window size, it will acquire a complementary page, starting from the beginning location of itself and extending backward. Otherwise it moves to the next page in the forward direction. If the page acquired is not in buffer,  $S$  has to send either a *complementary* or a *unified* I/O request to the I/O scheduler and wait for it. More details about the circular location computation, priority updating, and page latching etc. are omitted in our paper due to space limit.

Once a scan  $S$  has traversed all the pages in the table, we need to use *EndScan* step to complete the scan. A fast scan is ended when its current location and its beginning location meet. If  $S$  is the last scan in its group, the scan scheduler removes this group and checks whether the removal causes splitting of any other groups.

### 3.3 Grouping the Scans

Analogous to the discussion in section 2.3, the bandwidth demand of scans in  $S^3$  can be calculated as following. As  $B_{demand}$  consists of two parts: the *unified bandwidth* ( $B_u$ ) which is occupied by the *unified* I/O requests and the *complementary bandwidth* ( $B_c$ ) produced by the *complementary* I/O requests. Generally in each group  $g$ , we have

$$B_{u_{demand}}(g) = v_{slowest}, \forall v_i \in g;$$

and

$$B_{c_{demand}}(g) = \sum v_i - (|g|) * v_{slowest}, \forall v_i \in g.$$

Adding  $B_{u_{demand}}(g)$  to  $B_{c_{demand}}(g)$  gives  $B_{demand}(g)$ . Given a set of groups  $G_N = \{g_1, g_2, \dots, g_m\}$ ,  $B_{demand}(system)$  can be obtained as  $\sum_{g \in G_N} B_{demand}(g)$ .

When  $B_{demand}(system)$  is smaller than  $V$ , the response time of each scan  $S_i$  is  $rt_{min}$  as each can acquire full bandwidth. We minimize the  $B_{demand}(system)$  by grouping the scans, and splitting the groups when necessary. The central idea of the grouping and splitting is that the  $B_{c_{demand}}(system)$  could be reduced significantly if scans with similar speeds are clustered in a same group. As a result,  $B_{demand}(system)$  can also be reduced. We illustrate this optimization by giving an example of splitting a group.

**Example 1.** A group  $g$  containing two scans  $S_1(v_1 = 1000 \text{ p/s})$  and  $S_2(v_2 = 3000 \text{ p/s})$  is joined by a new scan  $S_3(v_3 = 4000 \text{ p/s})$ . This would cause splitting of group  $g$ , because the bandwidth consumption after the splitting can decrease by 1000 p/s, as indicated in Table 3.

**Table 3.** An Example of Splitting

	Scan Groups	$Bu_{demand}$	$Bc_{demand}$	$B_{demand}$
<b>Before Splitting</b>	$\langle S_1, S_2, S_3 \rangle$	1000	5000	6000
<b>After Splitting</b>	$\langle S_1 \rangle, \langle S_2, S_3 \rangle$	4000	1000	5000

On the other hand, when  $B_{demand}(system) > V$ , some I/O requests have to wait for the others' completion. To improve the average response times, the *unified* I/O requests are always given higher priority by the *I/O scheduler*. Therefore, too many splittings would impair the performance when the system is heavily loaded in I/O, as splitting always increases  $Bu_{demand}(system)$ . In Example 1,  $B_u(system)$  increases by 3000 after splitting. To control the number of the groups, we define two constraints in  $S^3$ . First, if the  $Bu_{demand}(system)$  is greater than a threshold  $T$  ( $T$  is usually a bit smaller than  $V$ ), we do not allow any splitting. Second, when the scan scheduler considers to attach a new scan to a group  $g$ , the remain life span of  $g$  must be larger than a threshold  $L_{max}$ . The reason is that if  $g$  will terminate in a short while, it might release its own bandwidth very soon. Therefore, there is no much benefit to attach the new scan to  $g$ .

We adopt a number of greedy algorithms for maintaining the groups. Once a scan arrives, the scan scheduler inserts it into the group which, if the new one becomes attached to it, produces the smallest  $B_{demand}(system)$ . After that, the scheduler checks if the group needs splitting via a routine called *CheckSplit*.

The basic idea of the *CheckSplit* algorithm is to find a splitting plan which minimizes  $B_{demand}(system)$  without causing  $Bu_{demand}(system) > V$ . If this optimal splitting plan causes reduction to the current  $B_{demand}(system)$ , splitting is beneficial and therefore will be truly undertaken. It can be proven that if there are  $n$  scans in a group  $g$ , sorted by their speeds in ascending order as  $g = (S_1, S_2, \dots, S_n)$ , then choosing the optimal splitting will be equivalent to finding an optimal index  $i \in [1, n - 1]$  such that vector  $g$  is split into two new vectors  $(S_1, \dots, S_i)$  and  $(S_{i+1}, \dots, S_n)$ . Therefore, the computational complexity of them is  $O(n)$ .

When a group is eliminated, its unified bandwidth can be released. The scan scheduler then searches among all existing groups for one which, if being split, causes the largest reduction of  $B_{demand}(system)$ . This search algorithm is implemented in the *PickSplitGroup* algorithm. The *PickSplitGroup* checks each existing

group using the *CheckSplit* routine and chooses the one which could benefit the system most. Again, groups whose remaining life spans are less than  $L_{max}$  are not considered. The chosen group is then split according to the respective optimal plan found by the *CheckSplit* algorithm.

It is worthwhile to mention that a group splitting causes changes to the window sizes of all groups. We adopt a simple method which redistributes the available buffer to each group uniformly.

### 3.4 Scheduling the I/O Requests

The I/O scheduler in  $S^3$  manages all I/O requests in two separate queues, namely  $Q_{unified}$  and  $Q_{comple}$ . The meanings of the queue names are self-explanatory. Once an I/O request, either a unified or complementary one, for page  $p$  is produced by  $S_i$ , it is appended to the tail of the respective queue in the form  $\langle p, S_i \rangle$ . The I/O scheduler dispatches the I/O requests as described in Algorithm 4.

#### Algorithm 4. I/O Scheduler

```

while true do
  if  $Q_{unified}$  is not empty then
     $\langle p, S \rangle \leftarrow \text{GetHeader}(Q_{unified})$ ;
    Read  $p$  from SSD into buffer;
    for each  $r_i$  in  $Q_{comple}$  do
      if  $r_i.S$  is in  $S$ 's group then
        delete  $r_i$ ;
      for each  $S_i$  in  $S$ 's group do
         $\text{Signal}(S_i)$ ;
    else if  $Q_{comple}$  is not empty then
       $\langle p, S \rangle \leftarrow \text{GetHeader}(Q_{comple})$ ;
      Read  $p$  from SSD into buffer;
       $\text{Signal}(S)$ ;
    else
       $\text{Wait}()$ ;

```

The I/O scheduler process is evoked whenever a scan needs to access a physical page on the SSD. It first manages the requests on the header of  $Q_{unified}$ . Once a *unified* request is completed, it eliminates all the *complementary* requests from the same group and evokes all the scans in the group. The requests in the  $Q_{comple}$  are only scheduled if  $Q_{unified}$  is empty. If there are no requests in both queues, the I/O scheduler sleeps until a new request arrives.

In case of heavy workload so that  $B_{demand}(system) > V$ , the *unified* I/O requests are still prioritized. The faster scans could succeed in performing *complementary* I/Os only when there are no pending *unified* I/O requests. When there are too many scans being executed concurrently, then  $S^3$  degenerates to the *group share* scheme as described in section 2.3.

## 4 Experiments

We implement the  $S^3$  framework on the PostgreSQL DBMS and conduct the performance study on a 20-scale *TPCH* database. The machine that we use is an Intel Xeon PC equipped with 8 cores and an Intel X-25E SSD. Our hardware can achieve a maximum random I/O bandwidth of 85 MB/s on Debian with kernel 2.6. Among the *TPCH* queries, we focus on  $Q_1$  and  $Q_6$ . The former is a typical CPU-intensive query, while the latter is an I/O-intensive one. Both queries scan the *lineitem* table, which consumes around 18GB space on the SSD. The default buffer size of the DBMS is set to 256MB. For more different query speeds, we also add two new queries, named  $Q'_1$  and  $Q'_6$ , which are slightly modified from  $Q_1$  and  $Q_6$ . The respective speeds of all four queries have the following relation:  $v(Q'_1) < v(Q_1) < v(Q'_6) < v(Q_6)$ . Therefore, it can be inferred that  $Q'_1$  is the most CPU-intensive query among the four, while  $Q_6$  is the most I/O-intensive one.

### 4.1 Results on Two Concurrent Query Streams

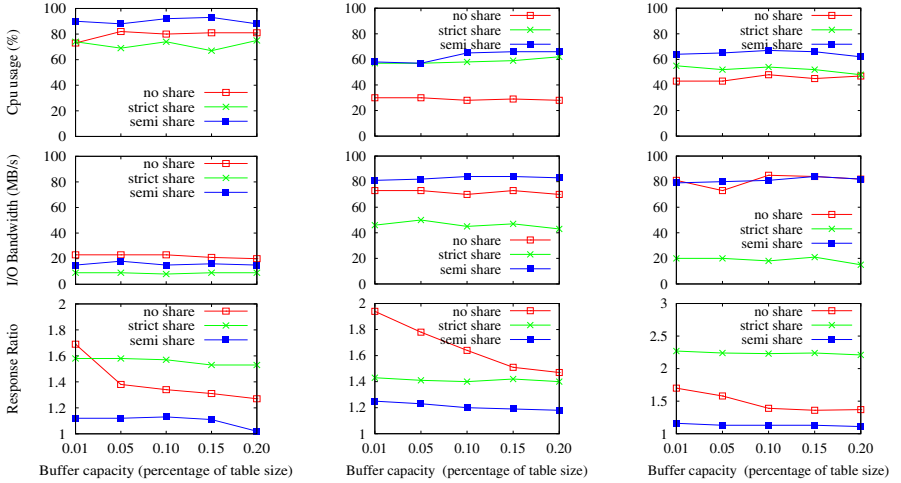
First we conduct a set of experiments on two concurrent streams of queries, where each stream issues the same query repetitively. For comparison, we also implement the *strict share* as described in Section 2.2. We do not compare with *group share*, as it apparently degenerates to the *strict share* or *no share* in this set of experiments.

**Table 4.** Statistics of different schemes when processing two concurrent query streams. Each cell under “response time” contains the two average response time values of the respective query streams.

	no share			strict share			semi share ( $S^3$ )		
	response time (s)	I/O throughput (MB/s)	CPU usage	response time (s)	I/O throughput (MB/s)	CPU usage	response time (s)	I/O throughput (MB/s)	CPU usage
$Q_1$ vs $Q'_1$	1261 2393.3	23.7	76% 88%	2090.1 2108.6	9.1	37% 100%	1096.7 2228.6	18.6	75% 100%
$Q_6$ vs $Q_1$	332.3 1230.1	63.9	20% 43%	906.6 915.3	20.5	4% 100%	225 910.9	80	30% 100%
$Q_6$ vs $Q'_6$	439.1 534.8	73.3	15% 44%	357.3 366.8	50.7	14% 99%	229.3 330.2	82.1	32% 82%

Table 4 shows the statistics of different schemes on various query streams. It can be seen that, in most cases,  $S^3$  provides considerably shorter average response time for each query. Under the *no share* scheme, the CPU-intensive workload ( $Q_1$  vs.  $Q'_1$ ) is completed quickly as they could fully exploit the CPU capacity and the I/O bandwidth. However, for  $Q_6$  vs.  $Q'_6$ , the average response times of both streams are poor as the two queries compete on I/Os. The *strict share* scheme performs well when two I/O-intensive queries  $Q_6$  and  $Q'_6$  are executed concurrently. However, it confines the faster scan to the slower one when they run concurrently, as the results of  $Q_1$  vs.  $Q_6$  indicate.



(a) CPU-intensive workloads ( $Q_1$  vs  $Q'_1$ )(b) I/O-intensive workloads ( $Q_6$  vs  $Q'_6$ )(c) Mixed workloads ( $Q_1$  vs  $Q_6$ )

**Fig. 2.** Statistics of different strategies on varying buffer size, workloads. The *average latency* of the query is normalized.

Figure 2 illustrates the results of varying the buffer size from 1% to 20% of the full table size. We use a smaller dataset of 5-scale data, in which the *lineitem* table is around 4.5GB. The *response ratio* of a scan  $S_i$  is defined as  $rt(S_i)/rt_{min}(S_i)$ . In Figure 2 we can see several interesting results. (1)  $S^3$  always has the highest CPU utilization compared to the other scheduling schemes. The reasons are two-fold. The first reason is that the faster query will never be blocked if the whole I/O bandwidth is not saturated. The second reason is that the slower query can always benefit from the faster one, as the pages loaded from SSD by the latter can be reused by the former. (2) Only the *no share* scheme is sensitive to the buffer size. The *strict share* and  $S^3$  do not appear to vary much by the buffer size as they intentionally share the buffer reading themselves. (3) The response ratio of  $S^3$  is always better than the other two. This confirms the *semi share* as an efficient scheme for concurrent scans on SSDs.

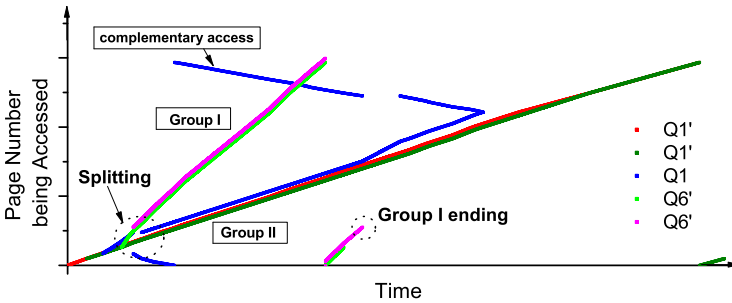
## 4.2 Results on Many Concurrent Queries

In this subsection we conduct more than two queries concurrently to study the performance and behavior of  $S^3$ . Specifically we execute 5 streams of queries, consisting of two streams of  $Q'_1$ , one stream of  $Q_1$ , and two streams of  $Q'_6$ . Table 5 presents the results of different scheduling schemes. The average response time of *no share* is slow because the bandwidth demand exceeds  $V$ . The *strict share* and *group share* could save the bandwidth efficiently. However, they delay the faster scans like  $Q_1$  or  $Q'_6$  to share the whole scan process. Our  $S^3$  scheme outperforms the other three in average response time because it exploits the I/O bandwidth of SSD more efficiently.

**Table 5.** Statistics of different strategies when processing five concurrent query streams

	no share			strict share			group share			semi share ( $S^3$ )		
	response time (s)	I/O thr. (MB/s)	CPU usage	response time (s)	I/O thr. (MB/s)	CPU usage	response time (s)	I/O thr. (MB/s)	CPU usage	response time (s)	I/O thr. (MB/s)	CPU usage
$Q'_1$	2459.4		80%	2153.2		83%	2196.7		82%	2132.4		94%
$Q'_1$	2499.4		80%	2144.6		83%	2160.5		83%	2202.8		83%
$Q_1$	1275.6	82.5	64%	2109	7	45%	2079.1	33.5	42%	1064.4	53	65%
$Q'_6$	651.8		31%	2094.2		9%	629		35%	545.2		40%
$Q_6$	647		26%	2099.1		9%	625.2		35%	589.2		35%

Finally, we conduct a more microscopic experiment, in which five queries are started in the order of  $Q'_1, Q'_1, Q_1, Q'_6, Q_6$ , each one is started 50 seconds later than its predecessor. Figure 3 plots all disk page IDs being accessed by time. It can be seen that after the first query  $Q'_1$  is started, new queries  $Q'_1, Q_1$ , and  $Q'_6$  are started subsequently. All these four are attached to the same group. When the final  $Q_6$  joins, the group is split into two, namely *Group I* including two  $Q_6$ 's, and *Group II* including two  $Q'_1$ 's and  $Q_1$ . Since  $Q_1$  is faster than  $Q'_1$ , it performs complementary I/Os to compensate the speed mismatch. We notice that when *Group I* ends,  $Q_1$  stops the complementary I/O for a while, as indicated by the gap in the top-most blue curve. This is because the window size of *Group II* is enlarged due to the elimination of *Group I*. Therefore  $Q_1$  can move forward for a certain period on the unified I/O stream, until reaching the end of the window.

**Fig. 3.** The process of 5 queries starting at an interval and running concurrently

To summarize, on both CPU-intensive and I/O-intensive workloads,  $S^3$  outperforms the other conventional schemes. Generally,  $S^3$  could improve the query efficiency for about 20% to 100%.

## 5 Conclusion

In this paper we propose a new framework, namely *Semi-Sharing Scan*, for processing concurrent scans on SSD efficiently.  $S^3$  groups the scans and compensates

the faster ones by random I/Os, if the hardware bandwidth is not saturated. Via I/O scheduling,  $S^3$  also improves the bandwidth utilization on I/O-intensive workloads. We implement  $S^3$  on the PostgreSQL DBMS. Experiments based on TPCB benchmark confirm that  $S^3$  is an efficient scheme for concurrent scans on SSD.

## References

1. Colby, L.S., et al.: Redbrick vista: Aggregate computation and management. In: Proc. ICDE (1998)
2. Cook., C., et al.: SQL Server Architecture: The Storage Engine. Microsoft Corp., <http://msdn.microsoft.com/library>
3. Jeff Davis Laika, Inc.: Synchronized Sequential Scan in PostgreSQL: <http://j-davis.com/postgresql/syncscan/syncscan.pdf>
4. NCR Corp. Teradata Multi-Value Compression V2R5.0 (2002)
5. Zukowski, M., Héman, S., Nes, N., Boncz, P.A.: Cooperative Scans: Dynamic Bandwidth Sharing in a DBMS. In: VLDB (2007)
6. Lang, C.A., Bhattacharjee, B., Malkemus, T., Padmanabhan, S., Wong, K.: Increasing buffer-locality for multiple relational table scans through grouping and throttling. In: ICDE (2007)
7. Lang, C.A., Bhattacharjee, B., Malkemus, T., Wong, K.: Increasing Buffer-Localities for Multiple Index Based Scans through Intelligent Placement and Index Scan Speed Control. In: VLDB (2007)
8. Lee, S.-W., Moon, B., Park, C., Kim, J.-M., Kim, S.-W.: A Case for Flash Memory SSD in Enterprise Database Applications. In: Sigmod (2008)
9. O'Neil, E.J., O'Neil, P.E., Weikum, G.: The LRU-K Page Replacement Algorithm For Database Disk Buffering. In: SIGMOD Conference (1993)
10. Johnson, T., Shasha, D.: 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm. In: VLDB (2004)
11. Nyherg, Chris: Disk Scheduling and Cache Replacement for a Database Machine, Master Report, UC Berkeley (July 1984)
12. Robinson, J., Devarakonda, M.: Data cache management using frequency-based replacement. In: Proc. ACM SIGMETRICS Conf. (1990)
13. Lee, D., Choi, J., Kim, J.-H., Noh, S.H., Min, S.L., Cho, Y., Kim, C.-S.: LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies. IEEE Trans. Computers (2001)
14. Lee, S.W., Moon, B.: Design of flash-based dbms: an in-page logging approach. In: SIGMOD Conference, pp. 55–66 (2007)
15. Tsirogiannis, D., Harizopoulos, S., Shah, M.A., Wiener, J.L., Graefe, G.: Query processing techniques for solid state drives. In: Sigmod (2009)

# Enhanced Foundry Production Control

Javier Nieves<sup>1</sup>, Igor Santos<sup>1</sup>, Yoseba K. Peña<sup>2</sup>,  
Felix Brezo<sup>1</sup>, and Pablo G. Bringas<sup>1</sup>

<sup>1</sup> S<sup>3</sup>Lab

<sup>2</sup> eNergy Lab

University of Deusto

Bilbao, Basque Country

{javier.nieves,isantos,yoseba.pena,  
felix.brezo,pablo.garcia.bringas}@deusto.es

**Abstract.** Mechanical properties are the attributes that measure the faculty of a metal to withstand several loads and tensions. Specifically, *ultimate tensile strength* is the force a material can resist until it breaks and, thus, it is one of the variables to control in the foundry process. The only way to examine this feature is the use of destructive inspections that renders the casting invalid with the subsequent cost increment. Nevertheless, the foundry process can be modelled as an expert knowledge cloud upon which we may apply several machine learnings techniques that allow foreseeing the probability for a certain value of a variable to happen. In this paper, we extend previous research on foundry production control by adapting and testing support vector machines and decision trees for the prediction in beforehand of the mechanical properties of castings. Finally, we compare the obtained results and show that *decision trees* are more suitable than the rest of the counterparts for the prediction of ultimate tensile strength.

**Keywords:** fault prediction, machine-learning, industrial processes optimization.

## 1 Introduction

Foundry is one of the axis of current economy: thousands of castings are created in foundries around the world to be part of more complex systems, say for instance, brake of a car, propeller of a boat, wing of an aircraft or the trigger in a weapon. As one may think, the tiniest error may have fatal consequences and, therefore, if one of the pieces is found faulty, this fact can be detrimental to both individuals and for businesses activities.

Moreover, current trends encourage the production of smaller and more accurate components. It is really easy to produce castings and suddenly discover that every single one is faulty. Unfortunately, although there are many standards and methods to check the obtained parts, these are carried out once the production has been completed. In this way, the most used techniques for the assurance of failure-free foundry processes, are exhaustive production control and diverse simulation techniques [1] but they are extremely expensive and only achieve good

results in an *a posteriori* fashion. Hence, providing effective *ex-ante* methods can help to increase the quality standards and to save resources in the process (i.e. saving money).

In this paper, we focus on the so-called *ultimate tensile strength* that is the force which a casting can withstand until it breaks, or in other words, it is the maximum stress any material can withstand when subjected to tension. Therefore, manufactured iron castings have to reach a certain value (threshold) of ultimate tensile strength in order to pass the strict quality tests.

As shown in [2,3], a machine-learning-based tool could help avoid these problems. In both approaches we presented a prediction system based on Bayesian networks. After a training period, the Bayesian network learnt the behaviour of the model and, thereafter, it was able to foresee its outcome illustrating how computer science can improve foundry production techniques.

Still, similar *machine-learning* classifiers have been applied in domains alike with outstanding results, for instance, neural networks [4] or the K-nearest neighbour algorithm [5]. In this way, successful applications of artificial neural networks include for instance spam filtering [6] or industrial fault diagnosis [7]. Similarly, K-nearest neighbour algorithm, despite its simplicity, has been applied for instance to automated transporter prediction [8] or malware detection [9]. These good results boosted us to test other machine learning models. Carrying out these experiments [10,11], we discovered that for each defect or property, the most accurate classifier was not always the same and, therefore, we decided to find out which classifier suited better to each domain.

Finally, some other machine learning models (as support vector machines [12] and decision trees [13]) have been used in less similar domains, such as, identification of gas turbine faults [14], fault diagnosis [15] and prediction [16].

Against this background, this paper advances the state of the art in two main ways. First, we address here a methodology to adapt machine learning classifiers, specifically support vector machines and decision trees, to the prediction of ultimate tensile strength and we describe the method for training them. Second, we evaluate the classifiers with an historical dataset from a real foundry process in order to compare the accuracy and suitability of each method.

The remainder of this paper is organised as follows. Section 2 details mechanical properties of iron castings, focusing on the ultimate tensile strength and how the foundry processes can affect them. Section 3 describes the experiments performed and section 4 examines the obtained results and explains feasible enhancements. Finally, section 5 concludes and outlines the avenues of future work.

## 2 Foundry Processes and Mechanical Properties

Several factors, for instance the extreme conditions in which it is carried out, contribute to render the foundry process very complex. Thereby, starting from the raw material to the final piece, this process has to go through numerous

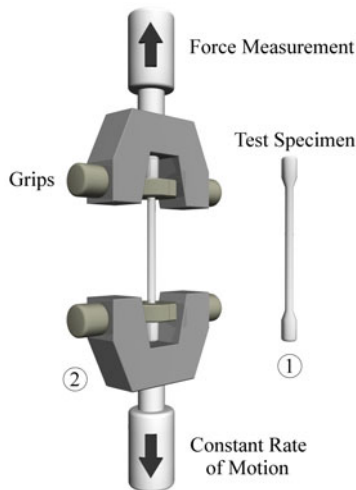
phases, some of which may be performed in parallel way. More accurately, when it refers to iron ductile castings, a simplification of this process presents the following phases.

First, the *melting and pouring phase* in which the raw metals are melt, mixed and poured onto the sand shapes . Second, the *moulding phase* in which the moulding machine forms and prepares the sand moulds. And last but not the least, the *cooling phase* where the solidification of the castings is controlled in the cooling lines until this process is finished.

When these aforementioned phases are accomplished, foundry materials are subject to forces (loads). Engineers calculate these forces and how the material deforms or breaks as a function of applied load, time or other conditions. Therefore, it is important to recognise how mechanical properties influence iron castings [17]. Specifically, the most important mechanical properties of foundry materials [18] such us strength (there are many kinds of strength as the ultimate tensile strength), hardness, resilience and elasticity.

Furthermore, there are common or standard procedures (i.e. ASTM standards [19]) for testing the value of mechanical properties of the materials in a laboratory. Unfortunately, in order to learn about these properties, scientists have to employ destructive inspections as the only possible method. Moreover, the process requires suitable devices, specialised staff and quite a long time to analyse the materials.

Regarding the ultimate tensile strength, on which we focus here on, its checking method is performed as follows. First, a scientist prepares a testing specimen from the original casting (see (1) in Figure 1). Second, the specimen is placed on the tensile testing machine (2). And finally, it pulls the sample from both ends and measures the force required to pull the specimen apart and how much the sample stretches before breaking.



**Fig. 1.** Ultimate Tensile Strength Test

Moreover, the main variables to control in order to predict the mechanical properties of metals are the composition [20], the size of the casting, the cooling speed and thermal treatment [17,21]. In this way, the system should take into account all these variables in order to issue a prediction on those mechanical properties. Hence, the machine-learning models used in our experiments are composed of about 25 variables.

### 3 Experiments

We have collected data from a foundry specialised in safety and precision components for the automotive industry, principally in disk-brake support with a production over 45000 tons a year. These experiments are focused exclusively in the ultimate tensile strength prediction.

Moreover, the acceptance/rejection criterion of the studied models resembles the one applied by the final requirements of the customer. Pieces flawed with an invalid ultimate tensile strength must be rejected due to the very restrictive quality standards (which is an imposed practice by the automotive industry). To this extent, we have defined two risk levels: Risk 0 (more than 370 MPa) and Risk 1 (less than 370 MPa).

In these experiments, the machine-learning models have been built with the aforementioned 25 variables. We have worked with 11 different references (i.e. type of pieces) and, in order to test the accuracy of the predictions we have used as input data the results of the destructive inspection from 889 castings (note that each reference may involve several castings or pieces) performed in beforehand.

Specifically, we have conducted the next methodology in order to evaluate properly the machine-learning classifiers:

- **Cross validation:** We have performed a *k-fold cross validation* [22] with  $k = 10$ . In this way, our dataset is 10 times split into 10 different sets of learning.
- **Learning the model:** We have made the learning phase of each algorithm with each training dataset, applying different parameters or learning algorithms depending on the model. More accurately, we have use this three different models:
  - *Support Vector Machines:* In order to train Support Vector Machines we have used different kernels: a polynomial kernel, a normalised polynomial kernel, a Pearson VII function-based universal kernel and a radial basis function (RBF) based kernel.
  - *Decision Trees:* We have performed experiments with *random forest*, an ensemble of randomly constructed decision trees using different amount of trees ( $n$ ):  $n = 10$ ,  $n = 50$ ,  $n = 100$ ,  $n = 150$ ,  $n = 200$ ,  $n = 250$  and  $n = 300$ . And we have also used *J48*.
  - *Artificial neural networks:* We have used a three-layer Multilayer Perceptron (MLP) learnt with *backpropagation* algorithm. We include this

model for comparison purposes because, as it is showed in previous work [10], it appears to be the best machine-learning model to foresee the ultimate tensile strength.

- **Testing the model:** For each classifier, we evaluated the percent of correctly classified instances and the area under the Receiver Operating Characteristic (ROC) that establishes the relation between false negatives and false positives [23].

## 4 Results

As we mentioned before, we have evaluated the classifiers in terms of prediction accuracy and the area under the ROC curve. In this way, Table 1 illustrates the obtained results in terms of prediction accuracy. Using the full original dataset of 889 evidences we can achieve an 86.84% of accuracy level. *Random forest with 250 trees* outperformed the rest of classifiers. On one hand, each of the random forest are better classifiers than the J48. Although both of them are based in decision trees, the first one is the best classifier and the second one is nearly the worst classifier. On the other hand, the deviation between all random forest is really small, but we can consider that the random forest with 250 trees like a local maximum.

**Table 1.** Results in terms of accuracy

Machine-learning Model	Accuracy (%)
Decision Tree: RandomForest with 250 trees	86.84
Decision Tree: RandomForest with 200 trees	86.77
Decision Tree: RandomForest with 300 trees	86.76
Decision Tree: RandomForest with 150 trees	86.68
Decision Tree: RandomForest with 100 trees	86.55
Decision Tree: RandomForest with 50 trees	86.53
Decision Tree: RandomForest with 10 trees	85.40
Artificial Neural Network: Multilayer Perceptron	84.23
SVM with Normalised Polynomial Kernel	83.78
SVM with Polynomial Kernel	82.07
SVM with Radial Basis Function Kernel	81.71
Decision Tree: J48	81.66
SVM with Pearson VII universal kernel	80.75

Notwithstanding, despite random forests have achieved better accuracy levels than the ANN, SVM-based classifiers could not overcome the ANN. Surprisingly, SVMs did not achieve as good results as we thought in beforehand because of their impressive results in information retrieval [24]. Hence, we can leave aside the SVM and J48 because they do not bring any improvement in the prediction of the ultimate tensile strength.

Furthermore, Table 2 shows the area under the ROC curve (AUC). In this way, the obtained results in terms of AUC are similar to the ones of prediction



**Table 2.** Results in terms of area under the ROC curve

Machine-Learning Model	Area under ROC curve
Decision Tree: RandomForest with 250 trees	0.9206
Decision Tree: RandomForest with 300 trees	0.9206
Decision Tree: RandomForest with 200 trees	0.9202
Decision Tree: RandomForest with 150 trees	0.9197
Decision Tree: RandomForest with 100 trees	0.9182
Decision Tree: RandomForest with 50 trees	0.9155
Decision Tree: RandomForest with 10 trees	0.8936
Artificial Neural Network: Multilayer Perceptron	0.8594
Decision Tree: J48	0.7626
SVM with Normalised Polynomial Kernel	0.7524
SVM with Polynomial Kernel	0.7445
SVM with Radial Basis Function Kernel	0.7151
SVM with Pearson VII universal kernel	0.6570

accuracy and the *random forest with 250 trees* also outperformed the rest of algorithms. Although all of them accomplish acceptable values (they exceed the line of no-discrimination), random forests outshine the other classifiers.

Actually, even the system has not achieved a 100% accuracy level, it has interesting results for being used in a high-precision foundry (more than 86%). In this way, we reduce in a significant manner the cost and the duration of the actual testing methods. Remarkably, the outstanding results achieved by the random forest with 250 trees show that it can be used in a similar way as we have used the Bayesian networks or artificial neural networks in previous works.

In addition, using this kind of predictive tool, the behaviour of the foundry workers can be the following one: when the system detects that the apparition's probability of an inadequate value of the ultimate tensile strength is very high, the operator may change the factors to produce this casting within the reference or change the whole reference (skipping the cost of having to re-manufacture it one more time). Also, the foundry workers and engineers can test the new configuration of a casting before they make it at foundry.

## 5 Conclusions

The ultimate tensile strength is the capacity of a metal to resist deformation when subject to a certain load. When a manufactured piece does not resist a certain threshold, it must be discarded in order to avoid breaking afterwards. Foreseeing the value of ultimate tensile strength renders as one of the hardest issues in foundry production, due to many different circumstances and variables that are involved in the casting process.

Our previous research [2][1] pioneers the application of Artificial Intelligence to the prediction of microshrinkages. Here, we have extended that model to the prediction of mechanical properties [3]. Later, we have focused on comparing *machine-learning* classifiers used for the prediction of ultimate tensile strength

[10]. Specifically in this research, we have included and adapted to our particular problem domain two classifiers that have been used widely in similar issues. All of them behave well, but random forests outperform the rest of the classifiers. Still, the ability of Bayesian theory and specifically, the sensitivity module (developed in [3]) cannot be ignored since it is an effective method that adds a decision support system for the operators in the foundry plant.

In addition, as we noticed in previous works [3,10,11], there are some irregularities in the data that may alter the outcome rendering it not as effective as it should. More accurately, these inconsistencies appear because the data acquisition is performed in a manual fashion.

Accordingly, the future development of this predictive tool will be oriented in five main directions. First, we plan to extend our analysis to the prediction of other defects in order to develop a global system of incident analysis. Second, we will compare more supervised and semi-supervised machine learning algorithms in order to prove their effectiveness to predict foundry defects. Third, we plan to integrate the best classifiers in a meta-classifier which will work as a black box combining all partial results to predict any defect. Fourth, we plan to employ some techniques (e.g. Bayesian compression) to give more relevance to the newer evidences than to the older ones. And, finally, we plan to test a preprocessing step to reduce the irregularities in the data.

## References

1. Sertucha, J., Loizaga, A., Suárez, R.: Improvement opportunities for simulation tools. In: Proceedings of the 16th European Conference and Exhibition on Digital Simulation for Virtual Engineering (2006) (invited talk)
2. Peña, Y., García Bringas, P., Zabala, A.: Advanced fault prediction in high-precision foundry production. In: Proceedings of the 6th IEEE International Conference on Industrial Informatics, pp. 1673–1677 (2008)
3. Nieves, J., Santos, I., Peña, Y.K., Rojas, S., Salazar, M., Bringas, P.G.: Mechanical properties prediction in high-precision foundry production. In: Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN'09), pp. 31–36 (2009)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
5. Fix, E., Hodges, J.L.: Discriminatory analysis: Nonparametric discrimination: Small sample performance. Technical Report Project 21-49-004, Report Number 11 (1952)
6. Elfayoumy, S., Yang, Y., Ahuja, S.: Anti-spam filtering using neural networks. In: Proceedings of the International Conference on Artificial Intelligence, ICAI, vol. 4, pp. 984–989
7. Simani, S., Fantuzzi, C.: Neural networks for fault diagnosis and identification of industrial processes. In: Proceedings of 10th European Symposium on Artificial Neural Networks (ESANN), pp. 489–494 (2002)
8. Li, H., Dai, X., Zhao, X.: A nearest neighbor approach for automated transporter prediction and categorization from protein sequences. *Bioinformatics* 24(9), 1129 (2008)

9. Santos, I., Peña, Y.K., Devesa, J., Bringas, P.G.: N-grams-based file signatures for malware detection. In: Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS), vol. AIDSS, pp. 317–320 (2009)
10. Santos, I., Nieves, J., Peña, Y.K., Bringas, P.G.: Machine-learning-based mechanical properties prediction in foundry production. In: Proceedings of ICROS-SICE International Joint Conference (ICCAS-SICE), pp. 4536–4541 (2009)
11. Santos, I., Nieves, J., Peña, Y.K., Bringas, P.G.: Optimising machine-learning-based fault prediction in foundry production. In: Omatu, S., Rocha, M.P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J.M. (eds.) IWANN 2009. LNCS, vol. 5518, pp. 553–560. Springer, Heidelberg (2009)
12. Vapnik, V.: The nature of statistical learning theory. Springer, Heidelberg (2000)
13. Quinlan, J.: Induction of decision trees. *Machine learning* 1(1), 81–106 (1986)
14. Maragoudakis, M., Loukis, E., Pantelides, P.P.: Random forests identification of gas turbine faults. In: ICSENG '08: Proceedings of the 2008 19th International Conference on Systems Engineering, Washington, DC, USA, pp. 127–132. IEEE Computer Society, Los Alamitos (2008)
15. Yang, J., Zhanga, Y., Zhu, Y.: Intelligent fault diagnosis of rolling element bearing based on SVMs and fractal dimension. *Mechanical Systems and Signal Processing* 1, 2012–2024 (2007)
16. Thissena, U., van Brakela, R., de Weijerb, A.P., Melssena, W.J., Buydens, L.M.C.: Using support vector machines for time series prediction. *Chemometrics and Intelligent Laboratory Systems* 69, 35–49 (2003)
17. Gonzaga-Cinco, R., Fernández-Carrasquilla, J.: Mecanical properties dependency on chemical composition of spheroidal graphite cast iron. *Revista de Metalurgia* 42, 91–102 (2006)
18. Lung, C.W., March, H., N.: *Mechanical Properties of Metals: Atomistic and Fractal Continuum Approaches*. World Scientific Pub. Co. Inc., Singapore (July 1992)
19. For Testing, A.S., *Materials: ASTM D1062 - Standard Test Method for Cleavage Strength of Metal-to-Metal Adhesive Bonds* (2008)
20. Carrasquilla, J.F., Ríos, R.: A fracture mechanics study of nodular iron. *Revista de Metalurgia* 35(5), 279–291 (1999)
21. Hecht, M., Condet, F.: Shape of graphite and usual tensile properties of sg cast iron: Part 1. *Fonderie, Fondeur d'aujourd'hui* 212, 14–28 (2002)
22. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence, vol. 14, pp. 1137–1145 (1995)
23. Singh, Y., Kaur, A., Malhotra, R.: Comparative analysis of regression and machine learning methods for predicting fault proneness models. *International Journal of Computer Applications in Technology* 35(2), 183–193 (2009)
24. Peng, T., Zuo, W., He, F.: SVM based adaptive learning method for text classification from positive and unlabeled documents. *Knowledge and Information Systems* 16(3), 281–301 (2008)

# The Profile Distance Method: Towards More Intuitive Multiple Criteria Decision Making in Information Systems

Edward Bernroider<sup>1</sup>, Nikolaus Obwegeser<sup>2</sup>, and Volker Stix<sup>2</sup>

<sup>1</sup> Royal Holloway, University of London, School of Management,  
Egham TW20 0EX, United Kingdom  
edward.bernroider@rhul.ac.uk

<sup>2</sup> Vienna University of Business and Economics, Institute for Information Business,  
Augasse 2-6, 1090 Wien, Austria  
{nikolaus.obwegeser,volker.stix}@wu.ac.at

**Abstract.** This paper seeks to improve usability and semantics of complex decision support based on multiple criteria and data envelopment analysis using the profile distance method. We recognize the need of decision making practice for more intuitive and understandable decision support in complex and comprehensive settings by proposing three areas for improvement. We suggest a more meaningful indicator of organizational fit, an advanced and dynamic multi-dimensional visualization, and embed support for weight estimation with pairwise comparisons into the method. The methodological advancements are shortly illustrated for an Information System selection problem.

## 1 Introduction

This paper proposes an augmentation of the profile distance method (PDM) originally proposed by Bernroider and Stix [1] to allow a more intuitive application of the method to increase its appeal to decision makers in practice. While decision making and investment appraisal techniques have received a lot of attention in the last decade, research reports an inability of management to holistically evaluate the implications of adopting a new technology in particular referring to Information Systems [2]. The absence of intuitive approaches and suitable implementations thereof to help the decision maker (DM) fully understand the underlying decision problem in all dimensions remains an open problem. Decision support systems (DSS) are often constructed and therefore understood as black box machines that calculate an output (e.g. a ranking or an efficiency score) based on some input information given by the DM. The methods and techniques used are often mathematically challenging in particular in selection problems based on multiple criteria. Thus, “the average DM” is not anywhere near fully understanding or comprehending the methods applied to adequately interpret results. This results in common mistakes and an underutilisation of potential benefits in applying methods. The Profile Distance Method (PDM)

itself constitutes one of the possible optimization based methods that a DSS can adopt. In contrary to many other optimization models the PDM actively encourages the DM to acquire a deeper understanding of the multi-dimensional structure of the decision problem, comparing the optimal individual profiles of each considered alternative (calculated by means of DEA based optimization) with a given desired weight profile (gained by using a utility model). The DM is thereby given a way not only to engage comparative analysis of alternatives but also to better understand the organizational fit considering all given dimensions. As the original authors mention, one essential issue is how well the method is embedded in a decision support system and what kind of access level in particular in terms of visualization is offered to the DM [1].

This paper intends to advance the PDM and its application in three ways, by (i) introducing a more intuitive understanding of the profile distance indicator, (ii) enabling the DM to graphically design the decision problem instead of formulating it on a text-based entry only and (iii) complementing left out parts of the decision process with a pairwise-comparison based desired weight profile (DWP) definition process.

Referring to (i), in order to implement a more intuitive indicator reflecting the information held as profile distance, we take a detailed look at the composition of the profile distance value and its interpretation. By focusing on user-oriented aspects, we translate the profile distance from a mathematical term to a meaningful indicator for the DM. This is achieved by defining an upper boundary for the profile distance value (a worst performing alternative) and normalizing all other alternatives thereby. We compare the possible alternatives, demonstrate the improved model using showcases and validate the increase in intuitivity and usability.

With regard to (ii), a second major improvement in practicability is proposed by implementing an interactive graphical interface. Based on a prototype implementation [3], a more advanced concept lets the user model the decision problem interactively with support of the tool. We argue that thereby the user can get into more direct contact with the problem and can test different scenarios to explore organizational fit of the alternatives. Additionally, the DM gets a deeper understanding of the indicators provided by the PDM to support the decision making process.

Thirdly (iii), we augment the existing PDM process by adding support for the definition of the desired weight profile (DWP) by making use of pairwise comparison concepts from the Analytical Hierarchy Process [4]. We thereby follow a three staged process including identification, analysis and implementation. Attention of many existing models and methods was placed on the analysis stage in particular regarding the PDM. We add functionality by using a pairwise technique for the definition of the desired state. In addition, the net of transitive dependencies created by the aforementioned technique is integrated in the analysis part, where the PDM model is augmented to react properly on input changes in advanced phases of the process.

The remainder of this article is outlined as follows. In the course of the next section we briefly present the Profile Distance Method and its foundations. Subsequently, we elaborate on the meaning of the profile distance and introduce the new indicator. In Sect. 4 an interactive graphical tool for decision modeling is introduced. Section 5 covers the pairwise-comparison process for tool-supported weight profile definition. Section 6 discusses and summarizes the findings and concludes with further research issues to be addressed.

## 2 The Profile Distance Method

The profile distance method (PDM) is a multiple attributive decision making method proposed by Bernroider and Stix [1] based on a linear optimization model. While the method was successfully implemented and tested for Information Systems (IS) selection problems, usability and visualization were identified as main targets for further development [3]. The linear optimization model of the PDM is grounded on the original Data Envelopment Model (DEA) model referred to as CCR-model proposed by Charnes, Cooper and Rhodes [5] mapping a fractional linear measure of efficiency into a linear programming (LP) format. It reflects a non-parametric approach optimizing a linear programme per decision making unit (DMU) yielding weights for both the chosen input and outputs, and a relative efficiency rating given by the sum of its weighted output levels to the sum of its weighted input levels. For a complete introduction into DEA see e.g. [6].

A review paper due to the recent 30 year anniversary of the popular method by Cook and Seiford [7] suggests connections into general multiple criteria decision models (MCDM) as an area for future DEA developments [7], which is in essence what the PDM seeks to achieve. The PDM links into a general MCDM framework termed utility ranking models (URMs), i.e., into an additive value model, which is concerned with selecting the best alternative among a finite set of possible choices based on multiple attributes reflected by partial utility functions [8]. Viewed in isolation, both, the DEA and URM approaches, have a number of problems in decision making practice. For example, the pure DEA approach achieves no clear cut ranking naturally evident in settings with many attributes and a few alternatives. On the other side, the URM creates super utility values based on biased estimations of weights that cannot easily be interpreted or justified. The idea of the PDM is to mitigate those and other problems by combining merits of both approaches. The mathematical representation of the PDM is an optimisation model and can be given as follows:

$$h_k = \max_{u,v} \sum_{r=1}^s u_r y_{rk} - \alpha \sum_{i=1}^s d_i (u_i - w_i u_1) \tag{1}$$

subject to:

$$\sum_{i=1}^m v_i x_{ij} - \sum_{r=1}^s u_r y_{rj} \geq 0 \quad \text{for } j = 1, \dots, n$$

$$\begin{aligned} \sum_{i=1}^m v_i x_{ik} &= 1 \\ \alpha d_i(u_i - w_i u_1) &\geq 0 \quad \text{for } i = 1, \dots, s \\ u_i, v_j &\geq 0 \quad \text{for all } i, j \end{aligned}$$

According to DEA, we have  $n$  alternatives each with  $m$  benefit attributes represented through the  $m \times n$  matrix  $X$  and  $s$  cost attributes stored in the  $s \times n$  matrix  $Y$ . The vectors  $v$  and  $u$  are the DEA multipliers or weight vectors for benefit- and cost-attributes, respectively. We have for each DMU a different LP which can lead to a different optimal solution. The parameter  $k$  selects the alternative for which the optimization should be performed. In contrast to DEA, the objective function (II) of PDM includes a penalty function, which measures the distance from a given desired weight profile given by the URM. It therefore accepts that the given weight profile is just an approximation of the true and ideal profile but seeks to penalise system alternatives depending on the distance of their DEA multipliers to the approximation. The function  $f$  measures the absolute distances between the weight vector  $u$  and the desired profile  $w$ . The fade-factor  $\alpha$  controls the tradeoff level of DEA ( $\alpha = 0$ ) and URM ( $\alpha \rightarrow \infty$ ), which allows the user to fade between both techniques, thereby exploring the organizational fit of the current alternative under evaluation. For more details on the metric used to measure distances and its implementation we refer to the original publication [1].

### 3 Improving the Profile Distance Indicator

The profile distance is one of the main benefits of the PDM and allows insights into the structure of the decision problem. It helps comparing the alternatives with each other and the desired profile. In the course of our research, we expand the benefits of the profile distance by normalizing its value within a lower and upper bound. We can thereby raise the intuitive understanding of the indicator and the PDM as a whole.

#### 3.1 Mathematical Interpretation

As follows from Sect. 2, the profile distance  $f$  represents the sum of all distances between the relative weights of a given profile (desired weight vector) and the relative optimized weight vector of an alternative, calculated by means of DEA-like optimization. Using the fade factor  $\alpha$  to force the optimized weights of an alternative closer to the desired profile, the profile distance decreases monotone with increasing  $\alpha$ . The fade factor  $\alpha$  is bound by the DEA optimization of each alternative in the case  $\alpha = 0$ . Its upper boundary is dependent on the underlying alternative, being reached at the point the profile distance  $f$  reaches zero. Moreover, the profile distance  $f$  boundaries are defined by zero as the lower bound and a data dependent upper bound given by DEA optimization.

### 3.2 Semantic Improvements

From the decision makers perspective the profile distance may be hard to interpret in its original composure. Furthermore, while the profile distance reflects suitability for a single option only, the DM is likely to be interested in relative performances between competing alternatives. We are therefore augmenting the model with problem specific boundaries for the profile distance, enhancing the meaning with an overlap rate of the alternatives' profile with the desired profile. This is achieved by defining an upper boundary for the profile distance, viz. a worst competing alternative. In the following we will present a case that demonstrates a way of creating such a boundary, using the worst competing alternative as upper bound.

We are using empirical data drawn from an enterprise resource planning selection problem comprising 3 possible alternatives (DMUs) with 8 output criteria each. The desired weight profile (DWP) has been assessed by the responsible decision makers and domain experts. Table 1 shows weights per alternative for each of the eight criteria resulting from a PDM optimization with an initial setting of  $\alpha = 0$ , thereby gaining full DEA compliance. The table contains the optimized weights for each DMU (rows) for each criterion (columns). The leftmost column (depicting the values for criterion 1) is always 1 since all values are normalized with the first value according to the recommendations of the PDM. The rightmost column shows the numerical value of the profile distance indicator.

**Table 1.** Relative weight vectors and profile distance

	Crit. 1	Crit. 2	Crit. 3	Crit. 4	Crit. 5	Crit. 6	Crit. 7	Crit. 8	Profile distance
DWP	1	1,619	0,523	0,619	0,714	1,047	1,523	1	0
DMU 1	1	1	10	1	1	1	1	1	0,384
DMU 2	1	1	1	1	1	1	1	10	0,356
DMU 3	1	1	7,551	1	1	8,967	10	1	0,464

In order to define an upper bound for a normalization of the profile distance value, the worst competing real DMU is taken into account. That is, the alternative with the highest profile distance value when optimizing under pure DEA conditions ( $\alpha = 0$ ). As Table 1 shows, the highest profile distance is reached by DMU3 with a value of 0.464. Due to the monotone decrease of the profile distance with increasing  $\alpha$ , this value can be considered as the largest under all possible circumstances. We argue that it is more intuitive for the DM to be presented an overlap rate rather than a distance measure, so we are transforming the profile distance values according to the following procedure. The division by the largest existent number (0.464) results in relative deviations, denoting DMU3 as the worst performer with a value of 1. Flipping the results by calculating (1 - deviation) allows us to interpret the outcome as overlap rate with the desired profile, thus stating that DMU3 has a relative overlap rate of 0 per cent, whereas DMU2 has a rate of 23 per cent and so forth. The results of this transformation are shown in Table 2.



**Table 2.** Transforming the profile distance into a structural overlap measure

	Profile distance	Deviation	Overlap rate
DMU 1	0,384	0,827	0,173
DMU 2	0,356	0,767	0,233
DMU 3	0,464	1	0

## 4 Interactive Visualizations with Radar Plots

Multiple research efforts have shown that visual approaches are particularly helpful in the area of MCDM, to communicate a clear picture of the likely complex decision problem [9,10]. With regard to PDM a previous implementation and empirical test of the method showed that satisfaction and acceptability can be substantially improved when providing graphical representations of the decision problem [3]. Although only static visualizations were used in this test, a significant number of respondents valued those features “very high” in terms of usefulness and practical convenience. Consequently, we were aiming at further improving visual support by adding an active visual decision modelling component to a PDM software tool and explored different techniques therefore. Choosing an effective graphical depiction for a decision problem is, however, not an easy task. That is particularly true in the area of MCDM, where multiple interdependent criteria and weights form a complex net of influences.

In a previous attempt to visualize the decision process using the PDM a bar chart was used to outline the weights of the desired profile in comparison with all DMUs. While this type of diagram gives a good overview of the weights for the alternatives for a specific attribute, it lacks the ability to integrate both the profile distance indicator as well as overall efficiency into the visualization. Consequently, it is not possible to include insights into the structural overlap with the desired profile. For this reason we decided to use a radar plot diagram to visualize the performance of the alternatives, additionally allowing the decision maker to interact with the graphical representation.

The radar plot chart is an integrated visualization of the criteria of the decision problem, showing not only weights and structural composition but also depicting the efficiency by means of surface area. The DM can easily change the desired profile by interacting with the radar plot in a drag and drop manner. The connected attribute weights are calculated accordingly in time so the user can actually get in touch with the decision problem more intuitively.

In the following we present an example scenario for the usage of radar plot within the PDM. Figure 1 displays settings drawn from the same data used in Sect. 3, comprising 3 DMUs with 8 output variables each. The desired weight profile is drawn in a bold water mark style, enabling the decision maker to easily compare the structure of the alternatives against it. While both DMU 2 and DMU 3 comply with the weight structure defined by the DM (full structural overlap), the larger surface area of DMU 2 indicates its better performance in overall efficiency. When forcing both affected alternatives into this condition, DMU 2 objectively outperforms DMU 3.

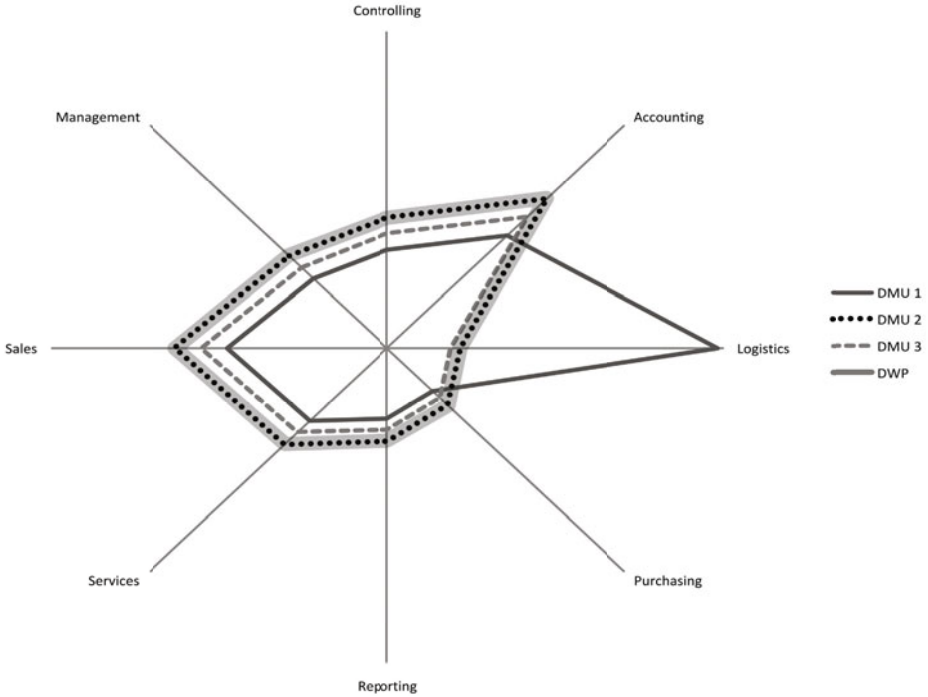


Fig. 1. Radar plot visualization ( $\alpha = 0.42$ )

## 5 Incorporation of Pairwise Comparisons

Using pairwise comparisons as an instrument of comparative judgement has been addressed by scientific research in various fields in particular by psychology and decision making theory. Pairwise comparisons have been applied to multitudinous scientific fields (e.g. voting systems, social choice or MCDM [11]). The underlying problem of estimating a weight vector has been an active area of research for a long time (e.g. [12]). A prominent application in decision support embodying pairwise comparisons and the eigenvector has been developed by T. Saaty, called the Analytic Hierarchical Process (AHP) [13]. Pairwise comparisons ultimately lead to a structure of the problem situation and a clear ranking of the decision makers preferences [14].

Referring back to phase one of PDM, the DM has to define a desired weight profile (DWP) that is to be compared to the performance of the alternatives. Since the issue of having a sound desired weight profile is of critical importance to the whole decision making process we are addressing this problem by pre-concatenating a pairwise comparison methodology to the PDM. In many utility ranking based decision making approaches, it is left to the decision makers' discretion how to estimate the weighting vector needed for value aggregation. We suggest to estimate the desired weights from pairwise comparison matrices and

refer to the standard approach in the AHP based on the eigenvector method. The DM is requested to define  $n$  influencing criteria in the beginning of the process to allow the construction of a  $n \times n$  square matrix  $A$ .  $A$  is defined to be a positive reciprocal matrix and has to follow the rules of transitivity (2) and reciprocity (3) to be consistent, for  $i, j$  and  $k$  being alternatives of the matrix:

$$a_{ij} = a_{ik} \cdot a_{kj} \quad (2)$$

$$a_{ij} = \frac{1}{a_{ji}} \quad (3)$$

As discussed in previous research the generation of a completely consistent pairwise comparison matrix is only realistic in very small matrices and does not necessarily reflect the real choice of the DM [15]. Many weight derivation methods account for this problem using one of the concepts of perturbation theory or distance minimization. Saaty proposes the principal eigenvector (EV) to be used for weight derivation to allow for slight inconsistencies in matrices to be reflected in the weight vector (see [16] for detailed information).

Here after establishing the  $n \times n$  square matrix  $A$  the principal diagonal holds the comparisons for the criteria with itself (all elements equal 1). All entries under the principal diagonal are subject to reciprocity. A fully consistent matrix allows  $n - 1$  elements to be chosen by the DM, whereas all others are calculated by means of transitivity. Since we do not force the DM into having a fully consistent matrix we allow comparisons for each pair of criteria, simultaneously filling the rest of the matrix according to transitivity rules. The DM can change this elements at any time, possibly generating inconsistencies. The matrix is needed to have a consistency rate lower than 0.1 in order to function properly with the eigenvector [13]. Using a multiplicative matrix with a 1 to 9 range value results very likely in an inconsistent matrix.

The EV is then derived using a heuristic approach, namely squaring the matrix with itself and then calculating and normalizing row sums. This procedure is repeated until the difference between the calculations no longer exceeds a given criterion. The result is the EV, which represents the input for the PDM to be used as a desired weight profile.

## 6 Conclusions and Future Work

This research paper extends existing research into multiple criteria decision making and data envelopment analysis referring to the profile distance method [1] and to its initial implementation [3]. We recognized the problem of decision making practice with meaningful applications of rather complex decision making methods and suggest three potential improvements of the PDM which were shortly illustrated for an Information System selection problem.

With the ambition in mind to provide a more complete approach to MCDM we augmented the original PDM methodology by giving support for estimating a sound desired weight profile. The idea of pairwise comparisons was extensively

exploited in AHP based evaluations not only in the area of Information Systems. For practicability, our implementation of PDM with computer software accounts for the quality of comparison matrices to support the decision maker in creating comparison matrices with inconsistencies that remain in certain boundaries.

Our second area of improvement demonstrated an interactive graphical interface based on a radar chart, which can include an additional dimension reflecting distances or overlap measures. We argue that thereby the users conceive additional information and get in more direct contact with the underlying model and its optimization results. A main feature is that the user can explore the problem and test different scenarios to explore organizational fit of the alternatives intuitively and directly. Thereby, the DM is expected to get a deeper understanding of the indicators provided by the PDM to support the decision making process.

We also proposed to replace or complement the main indicator of distance with a new overlap measure, that seems to be more intuitive and meaningful in its semantic representation. As possible drawback we see that a real worst upper bound to normalize the profile distance value could tempt the DM to prematurely dismiss this DMU from further analysis. To abate this setting a virtual worst DMU can be used instead. We can thus retain the benefits of setting the profile distance indicator of the alternatives into relation with each other, thereby making the measure better understandable while at the same time mitigating pre-mature exclusion of alternatives. The concept of adding artificial DMUs to a set of real DMUs is a common technique in the area of DEA research. Various different approaches make use of artificial DMUs, mostly due to the lack of discrimination power in the original DEA approach [17]. The investigation on how to generate a virtual DMU that represents the greatest profile distance for the PDM is a matter of ongoing research.

## References

1. Bernroider, E., Stix, V.: Profile distance method: a multi-attribute decision making approach for information system investments. *Decis. Support Syst.* 42(2), 988–998 (2006)
2. Gunasekaran, A., Ngai, E., McGaughey, R.: Information technology and systems justification: A review for research and applications. *European Journal of Operational Research* (173), 957–983 (2006)
3. Bernroider, E., Obwegeser, N., Stix, V.: Introducing complex decision models to the decision maker with computer software - the profile distance method. In: *Proceedings of the Knowledge Generation, Communication and Management (KGCM 2009)*, pp. 68–72 (2009)
4. Saaty, T., Air Force Office Of Scientific Researchboling AFB DC: *Optimization by the Analytic Hierarchy Process* (1979)
5. Charnes, A., Cooper, W.W., Rhodes, E.: Measuring the efficiency of decision making units. *European Journal of Operational Research* 2, 429–444 (1978)
6. Cooper, W., Seiford, L., Tone, K.: *Data envelopment analysis*. Springer, Heidelberg (2000)
7. Cook, W.D., Seiford, L.M.: Data envelopment analysis (dea)—thirty years on. *European Journal of Operational Research* 192, 1–17 (2008)

8. Yoon, K.P., Hwang, C.L.: Multiple Attribute Decision Making: An Introduction. Sage University Paper series on Quantitative Applications in, CA. the Social Sciences. Sage Publications, Thousand Oaks (1995)
9. Belton, V., Vickers, S.P.: Demystifying DEA—a visual interactive approach based on multiple criteria analysis. *Journal of the Operational research Society*, 883–896 (1993)
10. Korhonen, P., Laakso, J.: A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research* 24(2), 277–287 (1986)
11. Deng, H.: Multicriteria analysis with fuzzy pairwise comparison. *International Journal of Approximate Reasoning* 21(3), 215–231 (1999)
12. Barzilai, J.: Deriving weights from pairwise comparison matrices. *Journal of the operational research society* 48(12), 1226–1232 (1997)
13. Saaty, T.: Decision making with the analytic hierarchy process. *International Journal of Services Sciences* 1(1), 83–98 (2008)
14. Saaty, T., Bennett, J.: A theory of analytical hierarchies applied to political candidacy. *Behavioral Science* 22(4) (1977)
15. Barzilai, J., Cook, W., Golany, B.: Consistent weights for judgements matrices of the relative importance of alternatives. *Operations Research Letters* 6(3), 131–134 (1987)
16. Saaty, T.: Decision-making with the AHP: Why is the principal eigenvector necessary. *European Journal of Operational Research* 145(1), 85–91 (2003)
17. Allen, R., Athanassopoulos, A., Dyson, R., Thanassoulis, E.: Weights restrictions and value judgements in data envelopment analysis: evolution, development and future directions. *Annals of Operations Research* 73, 13–34 (1997)

# Hybrid In-Memory and On-Disk Tables for Speeding-Up Table Accesses

Joan Guisado-Gómez<sup>1</sup>, Antoni Wolski<sup>2</sup>, Calisto Zuzarte<sup>3</sup>,  
Josep-Lluís Larriba-Pey<sup>1</sup>, and Victor Muntés-Mulero<sup>1</sup>

<sup>1</sup> DAMA-UPC, Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, Campus Nord-UPC, 08034 Barcelona

{joan,larri,vmunes}@ac.upc.edu

<sup>2</sup> IBM Helsinki Laboratory, Finland

antoni.wolski@fi.ibm.com

<sup>3</sup> IBM Toronto Laboratory, Markham, Ontario, Canada L6G 1C7

calisto@ca.ibm.com

**Abstract.** Main memory database management systems have become essential for response-time-bounded applications, such as those in telecommunications systems or Internet, where users frequently access a table in order to get information or check whether an element exists, and require the response to be as fast as possible. Continuous data growth is making it unaffordable to keep entire relations in memory and some commercial applications provide two different engines to handle data in-memory and on-disk separately. However, these systems assign each table to one of these engines, forcing large relations to be kept on secondary storage.

In this paper we present TwinS — a hybrid database management system that allows managing hybrid tables, i.e. tables partially managed by both engines. We show that we can reduce response time when accessing a large table in the database. All our experiments have been run on a dual-engine DBMS: IBM<sup>®</sup>SolidDB<sup>®</sup>.

**Keywords:** Hybrid tables, Main memory databases, DBMS performance.

## 1 Introduction

As a result of cyberinfrastructure advances, the vast amount of data collected on human beings and organizations and the need for fast massive access by a large amount of users to these data pose a serious performance challenge for database management systems. For example, large *Domain Name Systems* (DNSs), that are used for translating domain names to IPs, may be constantly queried by a large amount of users per second that require the response to be as fast as possible even in peak situations where the amount of users increases. Efficiently detecting whether a domain exists or not must meet real-time requirements since this does not only speed up this query answer, but it also reduces the load of the system, accelerating other concurrent queries.

Queries in this type of applications are usually characterized for accessing data depending on a certain value of one of its attributes which are typically unique, such as the string containing the domain name in our examples. Also, it is very usual to find queries on data which do not exist in the database. For instance, domains that are not associated to any existent IP are very common. Looking for data which are not present in the database affects the overall capacity of the system to respond as fast as possible. In this situation, fast database management solutions such as main-memory database management systems (MMDBMSs) become essential.

Although MMDBMSs are efficient in terms of accessing or modifying data, they limit the total amount of data to the available memory. Some commercial MMDBMSs like IBM<sup>®</sup>SolidDB<sup>®</sup> or Altibase<sup>™</sup> resort to a hybrid solution implementing a second storage based on disk. This requirement causes large tables to be necessarily classified as on-disk, not allowing them to benefit from main memory techniques. This restriction directly collides with the fact that the size of large databases is in the petabytes nowadays and, therefore, it is very common to find massive tables that do not fit in memory entirely. This situation demands for a coupled solution where tables that do not entirely fit in memory may partially benefit from MMDBMS advantages.

It is important to take into account that classifying a table as on-disk does not strictly mean that the whole table is on disk, since buffer pool techniques may be used to keep the most frequently accessed information in memory. However, using efficient memory structures as those in MMDBMSs is preferable to the use of buffer pool techniques in disk-resident DBMSs, as discussed in [1], since the latter might not take full advantage of the memory.

In this paper, we propose TwinS — a hybrid in-memory/on-disk database management system that allows managing *hybrid tables*, i.e. the same table is split into two parts kept on disk and in memory, respectively. This poses a challenge when it comes to decide the existence and location of a specific row, when performing an access by using the value of a unique attribute. Our main contributions are: (i) proposing a new architecture that allows for hybrid tables, (ii) providing an efficient data structure for avoiding useless accesses to both memory and disk, specially when the queried information is not in the database, (iii) comparing our approach to a real commercial MMDBMS, IBM<sup>®</sup>SolidDB<sup>®</sup>, that provides both an in-memory and a disk-resident engine. Our results show that, thanks to our approach, we are able to speed up the overall execution when accessing rows in the table intensively. This provides us with experimental evidence that allows us to state that using in-memory data structures outperforms classical buffer pool techniques for on-disk DBMS.

The remainder of this document is organized as follows. Section 2 describes the architecture proposed for our hybrid database and our proposal in order to reduce unnecessary accesses to both memory and disk. In Section 3, we present our results. Section 4 presents some related work. Finally, in Section 5, we draw the conclusions and present some future work.

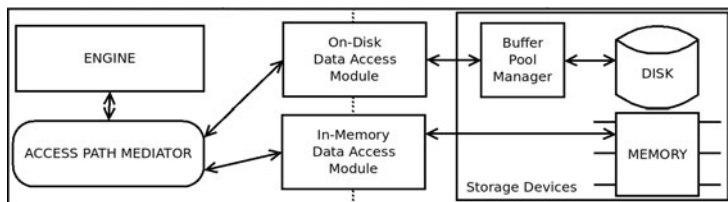


Fig. 1. TwinS architecture

## 2 TwinS

TwinS is a hybrid database management system that aims at improving the performance of disk resident databases by enabling hybrid tables which store part of their rows in memory. This architecture naturally requires the coexistence of two modules that separately manage the accesses to each of the storages. Let  $R = \{r_1, r_2, \dots, r_n\}$  be a hybrid table of  $n$  rows. In TwinS, we assume that a hybrid table  $R$  is divided into two parts,  $R_M$  and  $R_D$ , such that  $R = R_M \cup R_D$  and  $R_M \cap R_D = \emptyset$ , where  $R_M$  is the part of the table managed by the in-memory engine, and  $R_D$  is the disk-resident part.

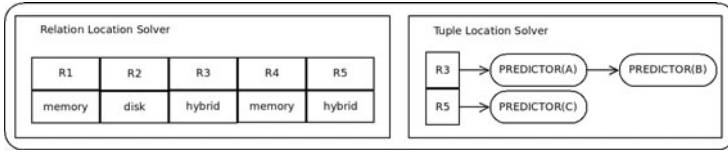
Figure 1 shows the architecture of our system with the modules involved in the execution of a query in TwinS, which is basically divided into two parts: the *engine* and the *access path mediator* (APM). The APM is in charge of solving data access paths and it takes into account that the location of data in TwinS may be distributed among two storage devices. Depending on the data location information in the APM, it redirects the query to the in-memory or on-disk data access modules used to access the rows stored in main memory or on disk, respectively. One of the main contributions of this paper is to design the APM such that it keeps information in a compact way and maximizes the efficiency of the system by avoiding unnecessary accesses to both memory and disk.

The APM has to have information of data location at row level. We do not discuss in this paper about sequential accesses because using hybrid tables instead of in-memory or on-disk tables does not add any complexity in this operation, since it would imply traversing data in both memory and disk sequentially and, therefore, the APM would not be strictly necessary.

### 2.1 A First Naive Approach: Latency Priority (LP)

In order to start the discussion, we present a first approach that takes the latency of each storage device as the only information to decide the storage to be accessed, prioritizing the one with the lowest latency, i.e. memory. We call this approach *Latency Priority* (LP). Thus, a naive solution to solve value-based queries on a hybrid table  $R$  would be trying to find data in the in-memory storage first and, if not found, trying to find it on disk. This solution forces us to access memory at least once for each value and, if the value is not found, it also requires accessing disk.





**Fig. 2.** Access Path Mediator design

The LP APM has a data structure called Relation Location Solver (RLS), which classifies each table in the system as in-memory, on-disk or hybrid.

## 2.2 Prediction Based Approach (PB)

Now, we propose a refined design of the APM which is depicted in Fig. 2. This version has two components: the Relation Location Solver (RLS)(left side of Fig. 2), and the Tuple Location Solver (TLS)(right side of Fig. 2), which is used to avoid useless accesses.

The main difference between LP and PB is that the latter incorporates a new type of structure that allows the system to know whether a row is in a specific storage or not. We call this new data structure *predictor*. The TLS is a structure in the form of a vector where each position points to a list of predictors. Predictors allow to know whether a row can be in memory or on disk depending on the values of its attributes. Thus, each hybrid table may have as many predictors as attributes in the table. Note that in Fig. 2, R3, identified as a hybrid table by the RLS, has one predictor for attribute *A* and another one for attribute *B*, and R5, also identified as a hybrid table, has only one predictor for attribute *C*. This way, for each queried value on a given attribute, the APM checks in the corresponding predictor (if the attribute has one associated) the location of the matching row.

Algorithm 1 describes the procedure to access a unique attribute. First PB gets the category of table *R* through the RLS (lines 2-3). If *R* is classified as a disk-resident or an in-memory table the system scans the table in order to find a certain value (line 2). If *R* is a hybrid table, the system checks in the TLS if there are rows of *R* fulfilling the condition related to that *value* in  $R_M$  (line 4),  $R_D$  (line 6) or in both of them (line 7), and then proceeds scanning the corresponding part. Note that *scan()* is not a sequential operation, but it consists in looking up the data through in-memory or on-disk indexes.

### Using Count Filters for Prediction

This section discusses on the implementation of the predictors depicted in Fig. 2. From previous sections, we may infer several aspects that are crucial for predictors:

The access time to the predictor has to be as fast as possible. Therefore, a predictor has to be in memory. This also implies that it has to be as compact as possible in order not to reduce significantly the amount of memory needed to execute the access operation. However, the accuracy of the predictor is crucial

**Algorithm 1.** Access by value

```

Input : Value 'value' of the attribute 'Attribute', Relation 'R'
Output: L::List of rows of R that  $R.Atr = val$ 
1  switch  $APM.checkRLS(R)$  do
2    case  $DISK$  or  $MEMORY$ :  $L \leftarrow Scan(R, Attribute, value)$ ;
3    case  $HYBRID$ :
4      switch  $APM.checkTLS(R, Attribute, value)$  do
5        case  $MEMORY$ :  $L \leftarrow Scan(R_M, Attribute, value)$ ;
6        case  $DISK$ :  $L \leftarrow Scan(R_D, Attribute, value)$ ;
7        case  $BOTH$ :  $L \leftarrow Scan(R_M \cup R_D, Attribute, value)$ ;
8      endsw
9    endsw
10 endsw

```

in order to be able to reduce the number of accesses to both memory and disk. Predictors may return false hits, i.e. predict that a row is in a certain storage when it is not true. This will result in an unnecessary access, but will preserve the correctness of results. However, they can never return a false negative [2], since that would imply not accessing the storage containing a row with the requested value. Although it is out of the scope of this paper to analyze the performance aspects regarding insertions, deletions and updates in the database, we need this structure to be ready to be updated if this happens.

The first two requirements are contradictory, since the first requires to keep predictors as small as possible, while the second needs keeping as much information as possible in order to make accurate predictions about the location of rows. A way to reduce the number of useless accesses is by implementing the predictor using two presence bitmaps in order to mark which values have been inserted in memory and on disk, respectively. A bit set to 1 in one of the bitmaps indicates that the value represented by that unique bit exists in the associated storage device. Using two bitmaps with as many positions as the size of the domain of the attribute that is linked to that predictor would allow us to have an exact predictor. With this structure we guarantee the four conditions stated above. However, in the presence of non-contiguous values or when the databases are updated very frequently, they would be very inefficient. The use of Bloom Filters [2] instead of bitmaps saves memory by means of applying hash functions at the cost of losing exact answers. The precision of a Bloom Filter depends on two different variables: the number of hash functions that are applied for each key and the size of the Bloom Filter. However, we would still have a problem when removing a value: we could not be sure that setting its corresponding bit to zero is correct, since other values could be mapped to the same bit. This is solved by using count filters. In order to implement the predictor we use Partitioned Dynamic Count Filters (PDCF) [3], a compact data structure that is able to keep approximate counters for large sets of values. As we see later, this structure requires little memory since it dynamically allocates and deallocates memory for the counters that need it. In our case, each predictor is composed by two PDCF, one for memory and another one for disk. Counters in the PDCF are assigned to values using a hash function. Therefore, each time a value is inserted in memory the corresponding counter (or counters) in the PDCF assigned to

memory is increased. When a value is removed the counter is decreased. Note that only when the counter is equal to zero, we are sure that the corresponding value is not in memory. Of course, the PDCF might return a false positive when more than one value is mapped on the same counter, but it will never return a false negative. It is also possible to allocate a single PDCF. For instance, if we wanted to avoid useless disk accesses, but we were not worried about memory accesses because it was the case that memory is very much faster than disk, we could consider creating a single PDCF. In general, it is worth to create both, since the space needed by these structures is not significant compared to the size of the database.

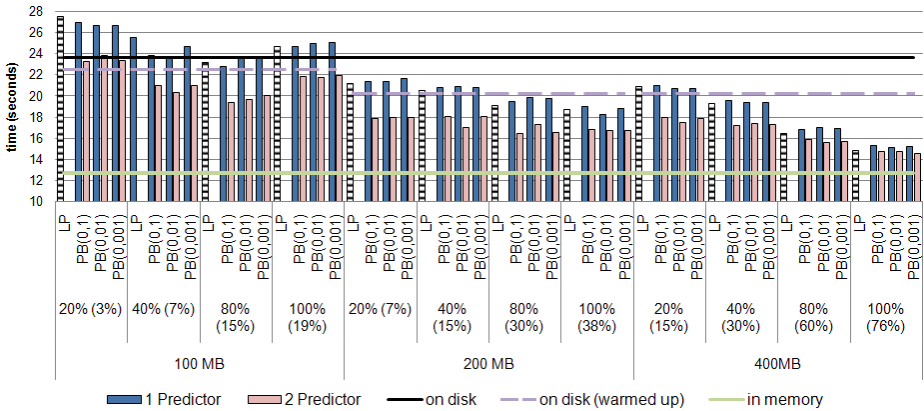
### 3 Experimental Results

This section analyses the effect of using hybrid tables. First, we start with the performance achieved by TwinS when accessing a table randomly by its primary key, using LP and PB with one or two predictors. Second, we analyze the impact of querying values which are not in the database.

Data have been generated with *DBGEN*, the TPC-H database generator. We run the experiments on the TPC-H table *orders*. The size of the table after loading  $10^6$  rows into memory using IBM SolidDB is 550 MB. We test TwinS varying the amount of available memory, the percentage of the memory used for the in-memory engine and the TLS probability of false hit. Unless explicitly stated, we consider for all the experiments three different values for the amount of available memory: 100, 200 and 400 MB. For each of these three cases, we vary the portion of memory used by the in-memory engine (20%, 40%, 80% and 100%). Note that the remainder of the memory (80%, 60%, 20% and 0%) is used for the buffer pool. This will allow us to understand whether it is more useful to use memory to keep part of a table using in-memory data structures, or it is better to use it as a classical buffer pool. Also, we test several configurations of the TLS, with a probability of yielding a false hit of 0.001, 0.01 and 0.1.

Each experiment is run three times, all the values shown in this section are the average of the results of these executions. When stated, the buffer pool is warmed up. The experiments are run using an Intel<sup>®</sup> Core<sup>™</sup> 2 Duo at 3.16GHz. The memory available by the computer is 3.5 GB. However, taking into account that during the execution of a query the scan operation may coexist with other operations in the QEP or even with concurrent queries from other users, we assign a maximum of 400 MB to table orders.

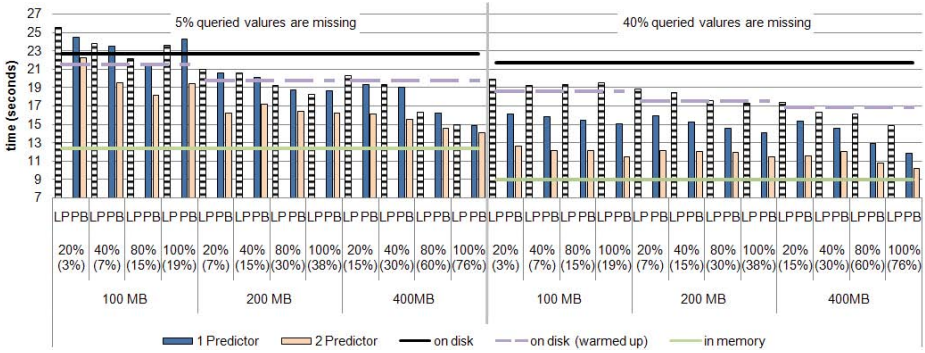
We test our architecture when accessing data at random using a unique attribute. Figure 3 shows the access time in seconds when using LP and PB with both one predictor and two predictors. With this experiment, we first aim at understanding the effect of the accuracy of predictors on the overall access time. Because of this, we have reserved an extra space of memory for predictors which is not competing with the memory pool reserved for keeping data. The percentage of tuples that fit in the in-memory engine appears in brackets next to the percentage of memory used for the in-memory engine. It also shows the access



**Fig. 3.** Access time when random access by primary key is done

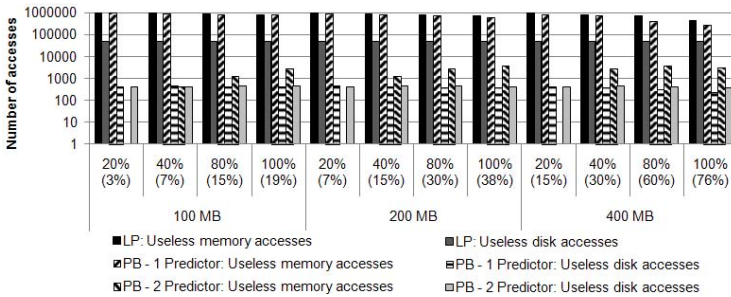
time when the system has very little memory and almost the whole table is on disk, the time when the table is on disk, but we have previously warmed-up the buffer pool, and the access time when the whole table fits in memory.

As we can see from these results, LP is able to achieve better results, when the amount of memory available is large and allows to keep a large percentage of the table in memory, as opposed to the case where the table is completely on disk and even to the case where the table is on disk but we have previously warmed up the buffer pool. Note that warming up the buffer pool might not be realistic in several scenarios, but we have included it to stress our proposals and show that even in this situation, we can benefit from hybrid tables. However, as expected, when the amount of memory is limited, and a large part of the table is on disk, since LP requires accessing memory for any access by value, performance degrades, making it better to use an on-disk DBMS. Regarding PB, we can observe that using one predictor will not save time with respect to LP, since LP saves accessing disk whenever the value is on memory and, therefore, the predictor is useless. Taking this into account, LP is desirable to PB with one predictor. On the contrary, with two predictors, we save unnecessary accesses to disk when the value we are looking for is on disk, and the overall time is reduced with respect to all the other approaches. Another important result is that it is more beneficial to keep data in the in-memory data structures, than using classical buffer pool techniques, agreeing with the ideas presented in [1]. This can be seen clearly in the 200 MB scenario of Fig. 3: when 80% of the memory available is used by the in-memory engine, any configuration with two predictors is able to reduce significantly the access time compared to the reduction that the buffer pool is able to achieve. Finally, we can see that the accuracy of the predictor results in very similar time responses. The proper configuration of the TLS which, as it is shown in Fig. 3, consists in using two predictor with probability of false hit under 0.1, only requires 6-7 MB, which represents 1.3% of the total amount of memory needed to store table (550 MB).



**Fig. 4.** Left: access time when 5% of the queried values are missing. Right: access time when when 40% of values are missing.

Following, we repeat the same experiments adding a certain number of queries to values that are missing in the database. We run the experiments taking into account two different situations: a) 5% of the queried values are missing (leftmost side in Fig. 4) and b) 40% of the queried values are missing (rightmost side in Fig. 4). Results focus exclusively on the case where predictors have a probability of false hit equal to 0.1 which has been shown to be the best case taking into account both response time and memory requirements.



**Fig. 5.** Analysis of useless disk accesses

Again, we show three baseline scenarios: the whole table fits in memory, the whole table is stored on disk, and the table is on disk and the buffer pool has been warmed up. LP is in general better than having the whole table on disk, except when the amount of memory available is not very large. If the buffer pool is warmed up, then buffer pool based tables obtain similar results to those obtained by LP, except if almost the whole table fits in memory in which LP achieves better response times. Regarding PB, it achieves response times that are in general shorter than those obtained by on-disk DBMSs. This difference is increased as the number of queried values which are not in the database increases. The difference between using one or two PDCFs is again significant,

and saving useless accesses to memory reduces the response time, as expected. The reasons for this are explained in Fig. 5, where we show the number of unnecessary accesses to both memory and disk, in logarithmic scale, when the percentage of queried values which are not in the database is 5%. Thanks to the first PDCF, we save most of the unnecessary disk accesses made by LP. The clear benefit of using the second PDCF is that the number of accesses to memory is reduced.

## 4 Related Work

Since the 1980s, with the availability of very large, relatively inexpensive, main memory, it became possible to keep large databases resident in main memory [4], and concepts such as main memory database management systems (MMDBMS) became usual. As described by Hector Garcia-Molina in [1], MMDBMSs store their data in main physical memory providing very high speed access. MMDBMSs use optimizations to structure and organize data. Because of the volatile characteristics of the main memory, reliability on MMDBMS has been one of the main concerns in the area and a lot of work has been done concerning the recovery, logging and checkpointing of such systems [5]. The interest on MMDBMS has increased [6] and many commercial MMDBMSs have been developed. MonetDB [7], Altibase [8], IBM-SolidDB [9] or Oracle-TimesTen [10] are just some examples. SolidDB and Altibase allow MMDBMS and conventional DBMS to coexist although they do not allow for hybrid tables in the way that we propose.

The use of structures such as PDCF is usual for several purposes such as analyzing data streams [11], summarizing the content of peer-to-peer systems [12], reducing file name lookups in scale distributed systems [13], etc. In databases, these structures have been used to answer queries regarding the multiplicity of individual items such as in [14].

## 5 Conclusions and Future Work

The results in this work show that using hybrid tables is a good solution when the whole table does not fit in memory. To our knowledge this is the first approach towards a hybrid in-memory and on-disk table allowing to reduce reading time by splitting the table into two parts. The use of predictors is essential in order to reduce useless accesses to both memory and disk. In the case of accessing values which are not in the database, foreseeing their non-presence and completely avoiding the access to data improves the overall performance of the system, making our proposal important for real-time applications, where a large number of users might be querying the same table.

The experiments have been done following a random pattern for accessing rows. However, it is still necessary to test the benefits of a hybrid table when the access pattern is skewed and a few rows are accessed very frequently, while the remaining rows are seldom accessed. Taking this into account, future work includes, among other possibilities, considering reallocation of data between devices in order to favor the system response time reduction.

## Acknowledgments

The authors would like to thank the Center for Advanced Studies (IBM Toronto Labs) for supporting this research. We also thank Generalitat de Catalunya for its support through grant GRC-1087 and Ministerio de Ciencia e Innovación of Spain for its support through grant TIN2009-14560-C03-03.

**Trademarks.** IBM is a registered trademark of International Business Machines Corporation in the United States, other countries, or both.

## References

1. Garcia-Molina, H., Salem, K.: Main memory database systems: An overview. *IEEE Transactions on knowledge and data engineering* 4(6), 509–516 (1992)
2. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13(7), 426 (1970)
3. Aguilar-Saborit, J., Trancoso, P., Muntés-Mulero, V., Larriba-Pey, J.-L.: Dynamic adaptive data structures for monitoring data streams. *Data Knowl. Eng.* 66(1), 92–115 (2008)
4. DeWitt, D., Katz, R., Olken, F., Shapiro, L., Stonebraker, M., Wood, D.: Implementation techniques for main memory database systems. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pp. 1–8. ACM, New York (1984)
5. Liedes, A., Wolski, A.: SIREN: a memory-conserving, snapshot-consistent checkpoint algorithm for in-memory databases. In: *ICDE'06. Proceedings of the 22nd International Conference on Data Engineering*, p. 99 (2006)
6. Stonebraker, M., Madden, S., Abadi, D., Harizopoulos, S., Hachem, N., Helland, P.: The end of an architectural era it's time for a complete rewrite). In: *Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB Endowment*, pp. 1150–1160 (2007)
7. Boncz, P., Kersten, M., Manegold, S.: Breaking the memory wall in MonetDB
8. Jung, K., Lee, K., Bae, H.: Implementation of storage manager in main memory database system altibase TM. In: Chen, J., Hong, S. (eds.) *RTCSA 2003. LNCS*, vol. 2968, pp. 499–512. Springer, Heidelberg (2004)
9. soliddb, <http://www-01.ibm.com/software/data/soliddb>
10. Team, C.: In-memory data management for consumer transactions the timesten approach. *ACM SIGMOD Record* 28(2), 528–529 (1999)
11. Wang, S., Hao, X., Xu, H., Hu, Y.: Finding frequent items in data streams using ESBF. In: Washio, T., Zhou, Z.-H., Huang, J.Z., Hu, X., Li, J., Xie, C., He, J., Zou, D., Li, K.-C., Freire, M.M. (eds.) *PAKDD 2007. LNCS (LNAI)*, vol. 4819, pp. 244–255. Springer, Heidelberg (2007)
12. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey. *Internet Mathematics* 1(4), 485–509 (2004)
13. Ledlie, J., Serban, L., Toncheva, D.: Scaling filename queries in a large-scale distributed file system. *Tech. rep., Citeseer* (2002)
14. Manku, G., Motwani, R.: Approximate frequency counts over data streams. In: *Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment*, p. 357 (2002)

# Low Power Management of OLTP Applications Considering Disk Drive Power Saving Function

Norifumi Nishikawa, Miyuki Nakano, and Masaru Kitsuregawa

Institute of Industrial Science, The University of Tokyo,  
4-6-1 Komaba Meguro-ku, Tokyo 153-8505, Japan  
{norifumi,miyuki,kitsure}@tkl.iis.u-tokyo.ac.jp  
<http://www.tkl.iis.u-tokyo.ac.jp/top/>

**Abstract.** A power-saving management for OLTP applications has become an important task for user budgets and datacenter operations. This paper presents a novel power-saving method for multiple disk drives based on knowledge of OLTP application behaviors. We report detail analysis of power consumption of disk drives and I/O characteristics of OLTP application. We then show experimental and simulation results of our power-saving methods. Our method provides substantially lower power consumption of disk drives compared to that of a conventional OLTP environment.

**Keywords:** Database, Online Transaction Processing (OLTP), Disk Drive, Power-Saving.

## 1 Introduction

Server and storage aggregation at datacenters has increased datacenters' power consumption. The power consumption of servers and datacenters in the United States is expected to double during 2006-2011 [1]. Storage is a high power consuming unit at large datacenters from a database-application workload perspective. Consequently, disk storage power-savings have become a major problem for database systems at datacenters [2,3].

A Database Management System (DBMS) is reportedly a major storage application [4]. A storage capacity shipment for DBMS is more than 60% of the total shipment of high-end class storage installations, and shipments for online transaction processing (OLTP) such as ERP and CRM constitute more than half of the shipments of storage installations for DBMS. Therefore, storage for OLTP is expected to be major power consumption unit at datacenters. Reducing power consumption of storage devices for OLTP is an important task that must be undertaken to decrease power consumption of datacenters.

Regarding power consumption of a storage unit such as RAID, power consumption of disk drives occupies about 70% of the total storage power consumption [5]. Today's disk drives have a power-saving function such as stopping the spindle motor or parking a head. These functions are useful for power-saving of disk drives; however, several hundred joules of energy and more than 10 s are



required to spin up the disk drives [6]. An inappropriate usage of the power-saving function therefore the increment of power consumption of disk drives and slows down applications. Consequently, it is important to select appropriate opportunities for using power-saving functions to reduce the disk-drive power consumption.

In the past few years, several studies have addressed these problems. The features of these studies are estimation of I/O-issued timing by analyzing application behavior. The length of the estimated period of time is limited by the length (latency) of the transaction. These approaches are, therefore, suitable for long-term transactions and not for short-term transactions. Consequently, it is difficult to apply these approaches to OLTP applications for which the transaction latency time is less than a few seconds.

Workloads of applications at a large datacenters are mainly short-term transactions such as banking or stock-market applications: OLTP applications. Power-saving methods using characteristics of I/O behavior of OLTP, however, have never been examined. In this paper, we focus on the most challenging problem of power saving of disk drives under the unfavorable OLTP environment that all disk drives are likely to be accessed constantly and equally.

Our contribution is to propose of a new power-saving method without OLTP performance degradation by considering OLTP I/O behaviors. The feature of our approach is to extend idle periods of disk drives using the comprehensive behavior of OLTP DBMS. Our approach uses I/O behavior knowledge of OLTP applications and background processes of DBMS. The other contribution is that we measured actual power consumption of OLTP applications on multiple disk drives using a power meter in detail. Few reports describe actual measurements of power consumption.

Our power-saving method enables reduction of disk drive power consumption of more than 38% in the best case in our experimental results. We also intend to use our proposed method to apply large RAID storage systems in future works.

## 2 Related Works

In this section, we describe related works of storage power-saving methods. These methods are classified into disk drive rotation control methods, I/O interval control methods, data placement control methods.

Disk drive control method controls a disk drive rotation speed or power status. Proposed approaches of disk drive control are categorized into two groups: i) changing the length of wait time to change the status of disk drive to standby or sleep [7,8]; ii) rotating a disk drive at multiple speeds [9,10]. These approaches typically use a long period of idle time such as 30 s. Moreover, application-aware power saving methods are also proposed to control disk drives precisely by using knowledge of applications [18]. However, the OLTP application executes multiple transactions in a few seconds and its idle lengths of I/O becomes less than one second, so it is difficult to apply these approaches independently.

I/O interval control method controls the I/O timing of an application to increase the chance that a disk drive is in a power-saving mode. A feature of this

approach is to increase the idle period using hierarchical memory architecture such as cache memory [11,12,13], and changing the application codes in order to control I/O timing [19,20]. These approaches are useful for applications with a small data footprint size or low I/O frequency, or a long-term transaction. Therefore, it is difficult to apply these methods to OLTP directly.

Data placement control method is intended to reduce the power consumption of disk drives by controlling data placement on disk drives. The idea of this approach is to concentrate frequently accessed data into a few disk drives, then to move the status of other disk drives to standby or sleep [14,15,16,17]. These approaches are also useful with applications for which the I/O frequency is low. It is not easy to apply these approaches to OLTP applications since there are many short transactions issued within few seconds. Consequently, it is necessary to combine other approaches that find less frequently accessed data at block level.

### 3 Characteristics of Disk Drive Power Consumption

This section explains characteristics of power consumption of disk drives based on actual measurement results.

#### 3.1 Measurement Environment

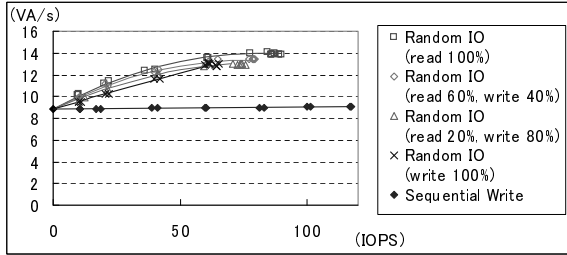
A load-generating PC provides power to a measured disk drive using 4-pin power cables. We connected a digital power meter (WT1600; Yokogawa Electric Corp.) to the power cables in order to measure the electric current and voltages of the disk drive. The load-generating PC CPUs are two Athlon 64 FX-74 3 GHz, 1MB cache, 4-core processors (Advanced Micro Devices, Inc.). Main memory sizes of the load generating PCs are 8 GB. Measured disk drive is Barracuda ES ST3750640NS (750 GB, 7200 rpm; Seagate Technology LLC). The disk drive write caches are turned off to protect reliability of the database because DBMS uses no write cache.

#### 3.2 Power Consumption at Active/Idle States

Fig. 1 depicts a relation between the disk drive power consumption and I/Os per second (IOPS). The I/O size is 16 KB, which indicates that the power consumption of random I/O increases in accordance with an increase of IOPS, but saturates the increase of power where IOPS is larger than 70-80 IOPS. The power consumption of a sequential write is much less than that of random I/O because the disk drive head movements are far fewer than those of random I/O.

#### 3.3 Power Consumption of Standby Status and Break Even Time

We also measured the power consumption of a standby state disk drive, along with the transition from active/idle status to standby status, migration from standby status to active/idle status, and break-even time of the disk drive.



**Fig. 1.** Relationship between the disk drive power consumption and I/Os per second (IOPS). The power consumption of random I/O increases in accordance with an increase of IOPS. The power consumption of a sequential write is much less than that of random I/O.

The power consumption of a standby state was 1.5 VA. The transition status from a standby to an active/idle, however, requires 8 s and more than an average of 23.3 VA. The transition status from an active/idle to a standby requires 3.5 VA with 0.2 s. The break-even time calculated from these values is 15.8 s. Therefore, idle time of more than 24 s (15.8 s + 8.0 s) is necessary to use this disk drive power-saving function. Hereafter, we call this idle time as "required idle time".

## 4 I/O Behavior of OLTP Application

For investigating a power-saving method using characteristics of I/O behavior of OLTP application, we measured I/O behavior of `tpcc-mysql` [22] on our test bed environment. Here, `tpcc-mysql` is a simple implementation of the TPC-C benchmark.

### 4.1 Experimental Environment

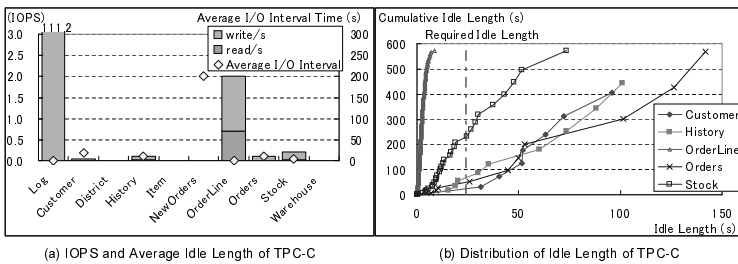
The hardware is the same configuration described in 3.1. The software configuration is the following: the OS is 32-bit version of CentOS; the DBMS is MySQL Communication Server 5.1.40 for Linux; and the OLTP application is `tpcc-mysql`. The file system cache and the disk drive are disabled. The size of the DBMS buffer is 2 GBytes. Our first target is high transaction throughput OLTP applications served at large datacenters. Therefore, we configured the size of DBMS buffer as larger than the size of the database.

The database is approximately 1 GByte (number of Warehouse is 10), in which the Log data size is not included. We partition a disk into 10 volumes and format these volumes using the Ext2 file system. Log data and each file of the tables and indexes are placed separately into each volume. This file placement eases the receipt of I/O performance data of each database data: we simply measure the OS level performance.

### 4.2 Behavior Characteristics of TPC-C

Fig. 2(a) shows the number of reads and writes per second and the average I/O intervals of each datum. As presented there, the characteristics of TPC-C I/Os were the following: i) write I/O to Log data were dominant, ii) I/Os to tables and indexes were fewer than two I/Os per second, and iii) more than half these I/Os were writes. No I/O was measured to District, Item, or Warehouse data. The I/O intervals of these data (Log, tables and indexes) were shorter than the required idle length (24 s) except for NewOrder data.

Fig. 2(b) portrays the intervals of I/Os of data. This figure reveals that the idle lengths of OrderLine, Orders, and Stock data were skewed; some idle period was longer than the required idle time. Therefore, a power-saving method using I/O behavior characteristics enables stoppage of disk drives for a long time.



**Fig. 2.** IOPS, average idle length, and distribution of idle length of TPC-C. The average I/O intervals of data is shorter than the required idle length, some idle period was, however, longer than the required idle time.

## 5 Power-Saving Method Using I/O Behavior Characteristics of OLTP Application

We propose a power-saving method using characteristics of the I/O behaviors of TPC-C application. The features of the proposed method are: i) to generate non-busy disk drives by gathering data of a few I/Os, ii) to delay writing I/Os to database data until the database data are read on the same disk drives.

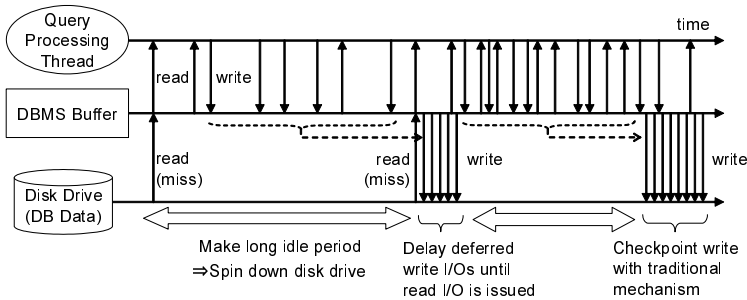
### 5.1 Power Saving Method Using Data Placement

This method gathers frequently accessed data into a few disk drives, and generate chance to spin down other disk drives which store infrequently accessed data. As shown in Fig. 2(b), we can observe long I/O intervals for Orders, History, Customer, and Stock data. We expect that this long I/O interval will enable us to use of the power-saving function of disk drives aggressively. This prediction of long I/O intervals cannot be achieved solely considering the I/O disk drive frequency of storage-level knowledge.

## 5.2 Power-Saving Method Using Delayed Write I/Os

We propose a delayed write I/O method to use long I/O intervals for power-saving of disk drives. This method is based on the DBMS behavior of writing operations.

Fig. 3 presents our proposed method. The main idea of our approach is to produce a long idle period by delaying deferred write I/O of database data until the DBMS reads database data on the same disk drive or a checkpoint. As shown there, our approach causes no delay of query processing threads. We can spin down the disk drive at this long idle period without degradation of OLTP throughput. The delay period should be defined based on the dirty page rate of DBMS buffer, the number of pages updated per second, and the interval length of a DBMS checkpoint. This subject warrants future study.



**Fig. 3.** Mechanism of delayed write I/O. Write I/Os to a disk drive are delayed until the DBMS reads database data on the same disk or a checkpoint.

## 6 Evaluation

### 6.1 Evaluation of Data Placement Method Based on Access Frequency

For evaluation of two disk drives, we add a SATA disk drive to the configuration described section 3 and 4 and put database data and Log data into two SATA disk drives. The added disk drive is the same model as those described in sections 3 and 4. We connected a digital power meter to the added drive and measured the disk drive power consumption. The configurations of DBMS and DB are as described in section 3 and 4. Disk drives are configured to transition to standby status when the idle period is longer than 5 s, and move to active state when an I/O arrives.

*Data Placement Variation.* We evaluated four data placements of two disk drives listed in Table 1. The disk drives are of two types: active and inactive. In this approach, disk drive #1 is active and disk drive #2 is inactive.

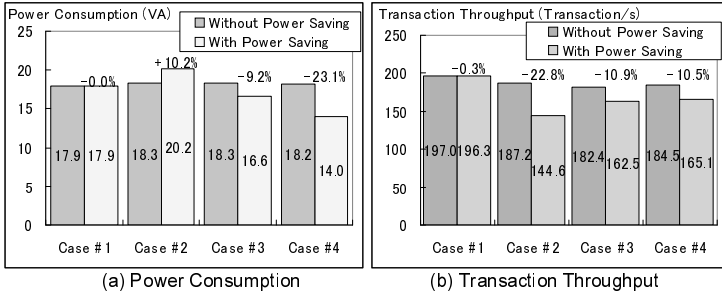
**Table 1.** Data Placement (Two Disk Drives)

Case	Data on Disk Drive #1 (Active)	Data on Disk Drive #2 (Inactive)
Case #1	Log	Customer, District, History, Item, NewOrders, OrderLine, Orders, Stock, Warehouse
Case #2	Log, OrderLine	Customer, District, History, Item, NewOrders, Orders, Stock, Warehouse
Case #3	Log, NewOrders, OrderLine, Orders, Stock	Customer, District, History, Item, Warehouse
Case #4	Log, Customer, NewOrders, OrderLine, Orders, Stock	District, History, Item, Warehouse

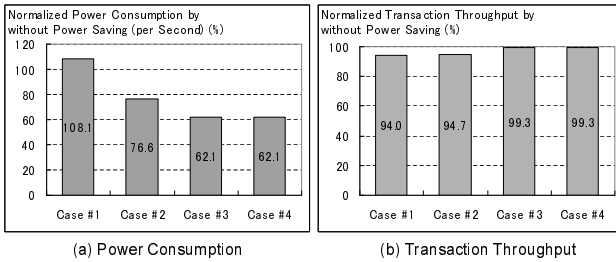
*Evaluation Results of Data Placement Method.* Fig. 4 shows the actual measured power consumption and transaction throughput of the two disk drives using the data placement, that is, data are distributed into two disk drives in Table 1. Here, we call the results using a spin down function of disk drive as “with power-saving”, the results which the spin down function are turned off as “without power-saving”.

As shown Fig. 4(a), the power consumption without power-saving is nearly equal to 18 VA for all cases. On the other hand, the power consumption with power-saving method is quite different among four cases. In case #1, the value is equal to the value without power-saving method. This means that in case #1, the data placement method does not reduce the power consumption of disk drives. In case #2, the power consumption is increased to 20.2 VA. In case #3 and #4, on the other hand, the power consumption values of disk drives are smaller than those without power-saving method. In most efficient case, case #4, the power consumption of the disk drives is approximately 23.1% smaller than those without the power-saving method. In Fig. 4(b), the transaction throughput with power-saving of case #1 is 196.3 transactions per second, and is nearly equal to the transaction throughput without power-saving. In case #2, the transaction throughput with power-saving is 144.6 transactions per second, and is shown to drop more than 22% compared with case #2 without power-saving. On the other hand, in cases #3 and #4 with power-saving, reduction of the transaction throughput keeps approximately 10% degradation.

From Fig. 4, we can find that the considerable data placement can achieve large power saving. In case #1, the power-saving function does not reduce the power consumption. This is because the all of idle length of the disk drives #1 and #2 are less than the standby timeout (5 s). In case #2, the power consumption is increased and transaction throughput is decreased because the length of the idle period of disk drive #2 is longer than the standby timeout but shorter than the required idle time (24 s). This causes an energy loss to spin up disk drive #2 and a transactions delay until the disk drive is spin up. Furthermore, this transaction delay stops I/Os to disk drive #1, causing another energy loss. In cases #3 and #4, the disk drive power consumption is reduced when using our proposed method because the idle periods are longer than the total length of required idle time (24 s). The degradation of transaction throughput was caused by waiting for disk drive #2 to spin up.



**Fig. 4.** Measured values of power consumption and transaction throughput (Two Disk Drives for DBMS). Proposed method reduces the disk drive power consumption by 23.1% with only 10% degradation of the transaction throughput.



**Fig. 5.** Power consumption and transaction throughput under two disk drives for DBMS with the delayed write I/O method. The power consumption of disk drives is reduced by 37.9% with little transaction throughput degradation.

## 6.2 Evaluation of Delayed Write I/O

We evaluated our proposed method with a delayed write I/O function. There is no implementation of a delayed write I/O function on commercial DBMS. Therefore, we simulate the I/O behavior of the delayed write I/O function using I/O trace information obtained from the experiments of data placement method. Then, we calculated power consumption and transaction throughput based on these I/O results.

Fig. 5 portrays the results of the disk drive power consumption and transaction throughput for each case. This result is normalized by the power consumption without power-saving. Here, the power-saving method contains both the data placement method and the delayed write I/O method.

As shown in Fig. 5(a), the disk drive power consumption is decreased except in case #1. The maximum reduction of power consumption was 37.9% for cases #3 and #4. Transaction throughput was reduced by 6% in case #1, 5.3% in case #2, and less than 1% in cases #3 and #4(Fig. 5(b)). The power consumption was increased and transaction throughput was decreased in case #1 for the reason described in the preceding subsection.

## 7 Conclusion and Future Works

As described in this paper, we measured the actual power consumption values of disk drives and considered the behavior of the TPC-C application in detail. We then proposed a novel power-saving method that enables reduction of power consumption of disk drives for TPC-C applications. The salient feature of our approach is to extend idle periods of disk drives using a data placement method and delayed write I/O method based on a comprehensive behavior of OLTP DBMS executing multiple transactions. We demonstrated that our method achieves an approximately 38% reduction of the disk drive power consumption for a TPC-C application without decreasing its throughput.

## References

1. U.S. Environmental Protection Agency ENERGY STAR Program, Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431, [http://www.energystar.gov/ia/partners/prod\\_development/downloads/EPA\\_Datacenter\\_Report\\_Congress\\_Final1.pdf](http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf)
2. Bauer, R.: Building the Green Data Center: Towards Best Practices and Technical Considerations. In: Storage Networking World Fall 2008 Conference (2008), <http://net.educause.edu/ir/library/pdf/bauer.pdf>
3. Chu, P.B., Riedel, E.: Green Storage II: Metrics and Measurement (2008), <http://net.educause.edu/ir/library/pdf/churiedel.pdf>
4. Reinsel, D.: White Paper Datacenter SSDs: Solid Footing for Growth, IDC #210290 (2008)
5. Poess, M., Nambiar, R.O.: Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. In: International Conference on Very Large Data Base, pp.1229–1240 (2008)
6. Product Manual Barracuda ES Serial ATA (2006), <http://www.seagate.com/staticfiles/support/disc/manuals/enterprise/Barracuda%20ES/SATA/100424667b.pdf>
7. Douglass, F., Krishnan, P., Bershad, B.: Adaptive Disk Spin-Down Policies for Mobile Computers. In: Proceedings of 2nd USENIX Symposium on Mobile and Location Independent Computing (1995)
8. Helmbold, D.P., Long, D.D.E., Sconyers, T.L., Sherrod, B.: Adaptive Disk Spin Down for Mobile Computers. In: Mobile Networks and Applications, vol. 5(4) (2000)
9. Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., Franke, H.: DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In: 30th Annual International Symposium on Computer Architecture (2003)
10. Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K., Wilkes, J.: Hibernator: Helping Disk Arrays Sleep through the Winter. In: Proceedings of Twentieth ACM Symposium on Operating Systems Principles (2005)
11. Papathanasiou, A.E., Scott, M.L.: Energy Efficient Prefetching and Caching. In: Proceedings of the USENIX 2004 Annual Technical Conference (2004)
12. Li, D., Wang, J.: EERAID: Energy Efficient Redundant and Inexpensive Disk Arrays. In: Proceedings of 11th Workshop on ACM SIGOPS European workshop (2004)



13. Yao, X., Wang, J.: RIMAC: A Novel Redundancy based Hierarchical Cache Architecture for Energy Efficient. In: Proceedings of High Performance Storage System 2006 EuroSys Conference (2006)
14. Colarelli, D., Grunwald, D.: Massive Arrays of Idle Disks for Storage Archives. In: Supercomputing, ACM /IEEE 2002 Conference (2002)
15. Pinheiro, E., Bianchini, R.: Energy Conservation Techniques for Disk Array Based Servers. In: Proceedings of 18th Annual International Conference on Supercomputing (2004)
16. Weddle, C., Oldham, M., Qian, J., Wang, A.A.: PARAD: A Gear-Shifting Power-Aware RAID. In: FAST'07: 5th USENIX Conference on File and Storage (2007)
17. Son, S.W., Chen, G., Kandemir, M.: Disk Layout Optimization for Reducing Energy Consumption. In: Proceedings of 19th Annual International Conference on Supercomputing (2005)
18. Gniady, C., Hu, Y.C., Lu, Y.H.: Program Counter Based Techniques for Dynamic Power Management. In: High Performance Computer Architecture (2004)
19. Heath, T., Pinheiro, E., Hom, J., Kremer, U., Bianchini, R.: Application Transformations for Energy and Performance-Aware Device Management. In: Parallel Architectures and Compilation Techniques (2002)
20. Son, S.W., Kandemir, M., Choudhary, A.: Software-Directed Disk Power Management for Scientific Applications. In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (2005)
21. Transaction Processing Performance Council, TPC-C, an online transaction processing benchmark, <http://www.tpc.org/tpcc/>
22. tpcc-mysql, <https://code.launchpad.net/~perconadev/perconatools/tpcc-mysql>

# A Disk-Based, Adaptive Approach to Memory-Limited Computation of Windowed Stream Joins

Abhirup Chakraborty and Ajit Singh

Dept. of Electrical and Computer Engineering  
University of Waterloo, ON, Canada, N2L3G1  
abhirupc@ieee.org, asingh@uwaterloo.ca

**Abstract.** We consider the problem of processing exact results for sliding window joins over data streams with limited memory. Existing approaches either, (a) deal with memory limitations by shedding loads, and therefore can not provide exact or even highly accurate results for sliding window joins over data streams showing time varying rate of data arrivals, or (b) suffer from large IO-overhead due to random disk flushes and disk-to-disk stages with a stream join, making the approaches inefficient to handle sliding window joins. We provide an Addaptive, Hash-partitioned Exact Window Join (AH-EWJ) algorithm incorporating disk storage as an archive. Our algorithm spills window data onto the disk on a periodic basis, and refines the output result by properly retrieving the disk resident data, and maximizes output rate by employing techniques to manage the memory blocks and by continuously adjusting the allocated memory within the stream windows. The problem of managing the window blocks in memory—similar in nature to the caching issue—captures both the temporal and frequency related properties of the stream arrivals. The algorithm adapts memory allocation both at a window level and a partition level. We provide experimental results demonstrating the performance and effectiveness of the proposed algorithm.

## 1 Introduction

With the advances in technology, various data sources (sensors, RFID Readers, web servers, etc.) generate data as high speed data streams. Traditional DBMSs stores data on a disk and processes streams of queries over the persistent data. Contrary to such a 'store first, query next' model, a data stream processing system should process results for long running, continuous queries incrementally as new data arrive in the system. So, continuous queries over bursty, high volume online data streams need to be processed in online fashion to generate results in real time. The new requirements unique to the data stream processing systems pose new challenges providing the motivation to develop efficient techniques for data stream processing systems. Examples of applications relevant to data stream processing systems include network traffic monitoring, fraud detection, financial monitoring, sensor data processing, etc. Considering the mismatch between the traditional DBMS and the requirements of data stream processing, a number of *Data Stream Management Systems* (DSMS) have emerged [1–3]. In this paper we investigate the problems that arise when processing joins over data streams.

Computing the approximate results based on load shedding [4–6] is not feasible for queries with large states (e.g., join with large window size). It is formally shown in

reference [5] that given a limited memory for a sliding window join, no online strategy based on load-shedding can be  $k$ -competitive for any  $k$  that is independent of the sequence of tuples within the streams. Thus, for a system with QoS-based [1] query output, secondary storage is necessary even to guarantee QoS above a certain limit. Pushing the operator states or stream tuples to the disk has already been adopted in several research works [7–10] that process finite data sets, carries out clean-up phase at the end of the streams. Contrary to this scenario, a sliding window stream join should properly interleave the in-memory execution and disk clean-up phases. Also, Algorithms for processing joins over finite streams [7, 8, 11] are not I/O-efficient and do not consider *disk I/O-amortization* over large number of input tuples. Moreover, these algorithms result in a low memory utilization and a high overhead of eliminating duplicate tuples in join output. Thus exact processing of sliding window stream joins within a memory limited environment is a significant and non-trivial research issue as promulgated in [5, 6, 10].

In this paper, we propose an Adaptive, Hash Partitioned Exact Window Join (AH-EWJ) algorithm that endeavors to smooth the load by spilling a portion of both the window blocks onto the disk. The proposed algorithm amortizes a disk-access over a large number of input tuples, and renders the disk access pattern largely sequential, eliminating small, random disk I/Os; it improves memory utilization by employing *passive removal* of the blocks from the stream window and by dynamically adjusting memory allocation across the windows. To increase the output generation rate, AH-EWJ algorithm employs a generalized framework to manage the memory blocks forgoing any assumption about the models (unlike previous works e.g. [5]) of stream arrival.

## 2 Related Works

Existing join algorithms on streaming data can be classified into two categories: the first one considers bounded or finite size relations, whereas the second category considers the streams of infinite size.

**Join over Bounded Stream.** Symmetric Hash Join [12], that extends the traditional hash join algorithm, is the first non-blocking algorithm to support pipelining. The XJoin algorithm [7] rectifies the situation by incorporating disk storage in processing joins: when memory gets filled, the largest hash buckets among  $A$  and  $B$  is flushed into disk. When any of the sources is blocked, XJoin uses the disk resident buckets in processing join. In reference [9], the authors present multi-way join (MJoin) operators, and claims performance gain while compared with any tree of binary join operators. Progressive-Merge Join (PMJ) algorithm [13] is the non-blocking version of the traditional sort-merge join. The Hash-Merge-Join (HMJ) [8] algorithm combines advantages of XJoin and PMJ. Rate-based progressive join (RPJ) [11] focuses on binary joins, and extends the existing techniques (i.e., those in XJoin, HMJ or MJoin) for progressively joining stream relations. Algorithm proposed in [14] is based on a state manager that switches between in-memory and disk-to-disk phases. In order to maximize overall throughput, RIDER [15] algorithm maximizes output rate and enables the system quickly switch between the in-memory stage and disk-to-disk stage.

These algorithms invokes reactive stages (i.e., phases involving disk-resident data) when the CPU is idle or there lies no more tuple to process. However, in case of the sliding window based joins, the clean up or invalidation is a continuous process and should be interleaved with the stream join processing.

**Join over Unbounded Data Stream.** Joins over infinite stream assumes application of window operators (time- or tuple-based) to limit the memory requirements of continuous queries, thus unblocking query operators. The work presented in [16] introduces a unit-time-basis cost model to analyze the performance of the sliding window joins over two streams. Reference [6] examines the MAX-subset measure and presents optimal, approximate offline algorithms for sliding window joins. Golab et al. [17] presented and evaluated various algorithms for processing multi-joins over sliding windows residing in main memory. Srivastava et al. [5] propose (age- and frequency-based) models that conserves memory by keeping an incoming tuple in a window up to the point in time until the average rate of output tuples generated using this tuple reaches its maximum value. In [4], the authors propose an adaptive CPU load shedding approach for multi-way windowed stream joins. All of the algorithms shed load and thus does not produce exact join results.

### 3 Exact Join

The basic join operator considered in this paper is a sliding window equi-join between two streams  $S_1$  and  $S_2$  over a common attribute  $A$ , denoted as  $S_1[W_1] \bowtie S_2[W_2]$ . The output of the join consists of all pairs of tuples  $s_1 \in S_1, s_2 \in S_2$  such that  $s_1.A = s_2.A$ , and  $s_1 \in S_1[W_1]$  at time  $s_2.t$  (i.e.,  $s_1.t \in [s_2.t - W_1, s_2.t]$ ) or  $s_2 \in S_2[W_2]$  at time  $s_1.t$  (i.e.,  $s_2.t \in [s_1.t - W_2, s_1.t]$ ). Here,  $s_i.t$  denotes the arrival time of a tuple  $s_i$ . The proposed algorithm (AH-EWJ) is based on the hashing methodology. Tuples in the stream windows are mapped to one of the  $n_{part}$  partitions, using a hash function  $\mathcal{H}$  that generates an integer in the range of  $[1, n_{part}]$ .

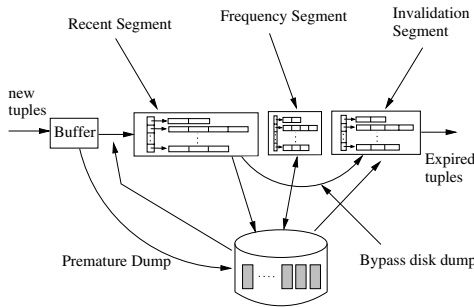


Fig. 1. Organization of incoming tuples within a window

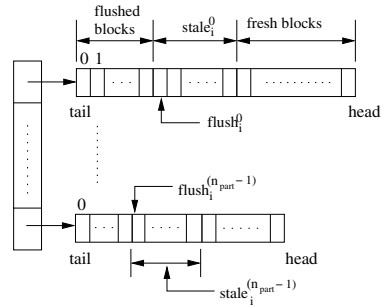


Fig. 2. Data structure of a Recent Segment of a Stream Window

The algorithm consists of four major sub-tasks: maintaining blocks in generative segments, adapting the memory allocation both within and across the stream windows,

probing the disk periodically to join the disk resident data with the incoming data, and invalidating the tuples. The algorithm is based on the framework given in Figure 1. The memory portion of each window  $W_i$  has three segments in total: invalidation segment ( $W_i^{inv}$ ), Recent segment ( $W_i^{rec}$ ), and Frequent segment ( $W_i^{freq}$ ). Each of the segments is maintained as a hashtable with  $n_{part}$  partitions or buckets. We denote the  $j$ -th partition of a segment  $W_i^{seg}$  as  $W_i^{seg}[j]$ , where  $seg \in \{rec, freq, inv, mem\}$ . Due to the lack of space, we omit detailed description of various sub-tasks and refer the readers to [18].

### 3.1 Memory Stage

The incoming tuples in stream  $S_i$  are mapped to the respective partitions using a hash function  $\mathcal{H}$  and added to the head block of the partition in the recent segment  $W_i^{rec}[p][H]$ . Each newly added block within a stream is given a unique number denoted as *block number*. Each block maintains a *productivity value* that records the number of output tuples generated by the tuples in that block. The arriving tuples are joined with the respective partition in the memory segment of the opposite window  $W_i^{mem}[p]$ . Blocks (in stream  $S_i$ ) are added to the recent segment at one end (i.e., head), while the blocks from the other end (i.e., tail) are stored to the disk sequentially during a disk dump. In our join algorithm, we maintain the following constraint:

*Constraint a: At a particular time, the blocks from recent segment of stream  $S_i$  should join with all the blocks in the disk portion of window  $W_i$  (i.e.,  $W_i^{disk}$ ) before being dumped to the disk*

Blocks in the recent segment are categorized into three types: flushed blocks, stale blocks and fresh blocks. The fresh blocks are newly arrived blocks that have not yet participated in the disk probing. The stale blocks have already participated in the disk probing, but have not been flushed onto the disk. The flushed blocks have already been dumped onto the disk, and, therefore, can be deallocated to accommodate newly arrived blocks. During the disk probing phase as described in subsection 3.2, blocks retrieved from the disk segment (of the opposite stream) are joined with the *fresh blocks* in the recent segment. After the disk probing, the fresh blocks are marked as stale ones. Blocks from the recent segment are evicted upon the arrival of new tuples.

### 3.2 Tasks Related to Disk Probing

**Frequent Segment Update.** The decision about placement or replacement of a block in Frequent segment, that is made periodically, is based on its productivity value. The productivity values are decayed, using a factor  $\alpha$  ( $0 < \alpha < 1$ ), during this update stage. This update stage is carried out during the disk probing. During the update stage, an incoming disk block is brought into the Frequent segment if its productivity exceeds that of a block already in Frequent segment at least by a fraction ( $\rho$ ). The block having the minimum productivity value among the blocks in the Frequent segment is replaced. To allow efficient decay of the productivity values of the disk resident blocks, we maintain a *productivity table* that stores the productivity values of all the disk resident blocks in memory as a list of tuples  $\langle \text{block number}, \text{productivity} \rangle$ .

**Algorithm 1.** EVICTRECENT-BLOCK(int k)

---

**Data:** k = stream index, global variables:  
 $W_i; i = 0, 1$   
**Result:** a freed block b

```

1   $i \leftarrow k;$ 
2  if CANSUPPLY( $\bar{k}$ ) then
3  |  $i \leftarrow \bar{k};$ 
4  if  $flush_i = 0$  then
5  | if
6  | |  $W_i^{disk} = \phi$  and  $free(W_i^{inv}) > 0$ 
7  | | then
8  | | |  $p \leftarrow$  partition having the block
9  | | | with the lowest  $BN$ ;
10 | | | remove trailing block b in
11 | | |  $W_i^{rec}[p];$ 
12 | | | copy b to  $W_i^{inv}[p];$ 
13 | | | decrement  $stale_i^p$ ;
14 | | else
15 | | |  $n \leftarrow$ 
16 | | |  $\min(B_{rec}^{min}, \sum_{p=0}^{n_{part}-1} stale_i^p);$ 
17 | | | DISKDUMP(i,n);
18 if  $flush_i \neq 0$  and  $free(W_i^{rec}) \leq 0$ 
19 then
20 |  $p \leftarrow \arg \min_{x | stale_i^x > 0} (prod_i^x);$ 
21 | remove trailing block b in  $W_i^{rec}[p];$ 
22 | decrement  $flush_i^p$ ;
23 | UPDATEFREQSEGMENT( $W_i^{freq}, b$ )
24 return b
```

---

**Algorithm 2.** AH-EWJ

---

```

// set to 0;  $i=0,1; 0 \leq p < n_{part}$ 
1 initialize variables  $premat_{rec}_i, freshN_i, UI_i,$ 
2  $flush_i^p, stale_i^p, BN_i;$ 
3 repeat
4 | retrieve a tuple  $s_i$  from input buffer of  $S_i$  in FIFO
5 | order;
6 |  $p \leftarrow \mathcal{H}(s_i);$ 
7 |  $W_i^{rec}[p][H] \leftarrow \{W_i^{rec}[p][H] \cup s_i\};$ 
8 | if  $W_i^{rec}[p][H]$  is full then
9 | | compute  $W_i^{rec}[p][H] \bowtie \{W_i^{mem}[p]\};$ 
10 | |  $W_{rec(i)}[p][H].BN \leftarrow BN_i;$ 
11 | | increment  $BN_i;$ 
12 | | increment  $freshN_i;$ 
13 | | if  $W_i^{rec}$  is full then
14 | | |  $W_i^{rec}[p] \leftarrow$  EVICTRECENTBLOCK(i);
15 | | else
16 | | | add a free block to the head of  $W_i^{rec}[p];$ 
17 if  $freshN_i \geq rN_{rec}^{min}$  then
18 |  $UI_i \leftarrow UI_i + freshN_i \times |b_i^m|;$ 
19 |  $freshN_i \leftarrow 0;$  // reset  $freshN_i$ 
20 | if  $UI_i \geq Epoch_i$  then
21 | | DISKPROBE( $W_i^{rec}, W_i^{disk}, W_i^{freq},$ 
22 | | 1);
23 | |  $UI_i \leftarrow 0;$  // reset the count
24 | else
25 | | DISKPROBE( $W_i^{rec}, W_i^{disk}, W_i^{freq}, 0$ )
26 | for  $p \leftarrow 0$  to  $n_{part} - 1$  do
27 | |  $stale_i^p \leftarrow |W_i^{rec}[p]| - flush_i^p$ 
28 until Streams ends;
```

---

**Eliminating Redundant Tuples.** A block  $b_i^p$  evicted from the Frequent segment of window  $W_i$  is already joined with the fresh blocks in the Recent segment of the stream  $W_i$ . If the evicted block is not already scanned on the disk, it will be read and be joined with the fresh blocks in the Recent segment of window  $W_i$ . To prevent this duplicate processing, we maintain a list  $E_i$  containing the block numbers of evicted blocks. If an incoming block from the disk is contained in  $E_i$ , we omit processing that block. List  $E_i$  is reset to *null* at the end of the update stage.

As described earlier, blocks from the Recent segment are removed on demand. Such passive removals of the blocks might lead to duplicate output generation. Let us consider a block  $b_i^j$  in partition  $\mathcal{H}(b_i^j)$  of the recent segment  $W_i^{rec}$ . The block  $b_i^j$  joins with a block  $b_i^k \in W_i^{rec}[\mathcal{H}(b_i^j)]$ . Later  $b_i^j$  participates in a disk dump and is stored on the disk. However, due to the passive removal of the blocks from the Recent segment, the same block ( $b_i^j$ ) remains in memory as a stale block. Now, when a block  $b_i^k$  of the opposite window participates in disk probe, it finds on the disk the block  $b_i^j$  already joined in previous step. We use the sequence number of a block, denoted as *block number*, in solving this issue: every incoming block is assigned a unique number from an increasing sequence. Every block  $b_i^k$  in partition  $\mathcal{H}(b_i^k)$  of the Recent segment  $W_i^{rec}$  stores the minimum block number (*minBN*) among the blocks, from the same partition of the Recent segment of the opposite window, that  $b_i^k$  joins with. When  $b_i^k$  participates in a

disk probe, it joins with a block  $b_i^p \in W_i^{rec}[\mathcal{H}(b_i^k)]$  if the block number of the block  $b_i^p$  is less than the  $minBN$  value of the block  $b_i^k$ , i.e.,  $b_i^p.BN < b_i^k.minBN$ . As  $b_i^k$  is already in  $W_i^{mem}$ , any block in  $W_i$  arriving after  $b_i^k.minBN$  is already joined with  $b_i^k$ . So, during disk probing any block  $b_i^p \in W_i^{disk}$  having  $b_i^p.BN \geq b_i^k.minBN$  can be omitted.

### 3.3 Adapting Window and Bucket Sizes

In the join scheme, sizes of the invalidation and the frequent segments are kept constant; the remnant join memory ( $M_{free}$ ) is allocated between the recent segments. The recent segments adapts their sizes depending on stream arrival rates [19]. We capture the instantaneous arrival rate of a stream using a metric termed as arrival frequency ( $C_i; i = 1, 2$ ), that is maintained using a decaying scheme [18]. Within a recent segment, a partition with the lowest value of the productivity metric ( $prod_i^p$ ) is selected as a victim partition, that yields a taling, flushed block. Algorithm 1 presents the algorithm for evicting a block from a recent segment.

### 3.4 Join Algorithm

We now present the join algorithm AH-EWJ in Algorithm 2. Within the join algorithm an infinite loop fetches, in each iteration, tuples from an input buffer and joins the fetched tuples with the opposite stream. At each step, the stream having a pending tuple/block with lower timestamp (i.e., the oldest one) is scheduled. A tuple fetched from the buffer is stored in the block at the head of the respective partition within the Recent segment. If the head-block of a partition  $p$  becomes full, it is joined with the memory portion of the partition ( $W_i^{mem}[p]$ ). The partition  $p$  is allocated a new head block (line 2-2). The variable  $freshN_i$  tracks the number of fresh blocks in  $W_i^{rec}$ . Whenever a recent segment is filled with at least  $N_{rec}^{min}$  fresh blocks, the disk probe phase is invoked (line 2). Frequent-segment-update phase, that occurs at a interval no less than  $Epoch_i$ , is merged with a disk-probe (described in 3.2). At the end of the disk-probe, parameter  $stale_i^p$  for each partition  $p$  is changed, converting the fresh blocks within the partition to stale ones (line 2-2).

## 4 Experiments

This section describes our methodologies for evaluating the performance of the AH-EWJ algorithm and presents experimental results demonstrating the effectiveness of the proposed algorithm. We begin with an overview of the experimental setup.

**Table 1.** Notations and the system parameters

Notations	Description
$S_i, S_{\bar{i}}$	stream $i$ and opposite to $i$ , respectively
$W_i, W_{\bar{i}}$	Window of stream $S_i$ and $S_{\bar{i}}$ , respectively
$W_i^{disk}$	Disk portion of window $W_i$
$W_i^{rec}$	recent segment of window $W_i$
$W_i^{freq}$	Frequent segment of window $W_i$
$W_i^{inv}$	invalidation segment of window $W_i$
$W_i^{mem}$	memory portion of window $W_i$ , ( $W_i^{rec} + W_i^{freq} + W_i^{inv}$ )
$W_i^{seg}[p]$	partition $p$ of $seg \in \{rec, freq, inv\}$
$W_i^{rec}[p][H]$	block at the head of $W_i^{rec}[p]$
$b_i^p$	block $p$ in window $W_i$
$b_i^p.minBN$	minimum block number from $W_i^{disk}$ & $W_i^{rec}$ that $b_i^p \in W_i^{rec}$ joins with
$b_i^p.prod$	Productivity of block $b_i^p$

## 4.1 Simulation Environment

All the experiments are performed on an Intel 3.4 GHz machine with 1GB of memory. We implemented the prototype in Java. The main focus of our experimentation is to observe the effectiveness of the join processing algorithm. The buffer is virtually unbounded and plays no role in the experiments. We generate the time varying data streams using two existing algorithms: PQRS [20] and *b-model* [21]. PQRS algorithm models the spatiotemporal correlation in accessing disk blocks whereas the *b-model* captures the burstiness in time sequence data [19].

To model the access time for the disk segment, we divide a disk segment ( $W_i^{disk}$ ) into  $n_i$  basic windows  $B_{ij}^{win} (j = 1 \dots n_i)$  [4]. We assume that disk blocks within a basic window are physically contiguous, and the delay in accessing the blocks in a basic window is estimated only by the disk bandwidth (i.e., no seek or latency). However, accessing blocks beyond the boundary of a basic window imparts an extra delay equivalent to the seek time and rotational latency (i.e., an access time). As a base device, we consider IBM 9LZX. We fix the memory page and also the disk block size to 1KB. Each record has a length of 64 bytes. We set the size of the basic window to 1MB. We allocate 70% of memory to the recent segments. The remaining memory is equally distributed among the invalidation and frequent segments. Minimum size of a recent segment ( $B_{rec}^{min}$ ) is set to the fraction 0.3 of the total memory reserved for the recent segments. We set the minimum delay between successive reactive stages ( for RPWJ) as 15sec. The domain of the join attribute  $A$  is taken as integers within the range  $[0 \dots 10 \times 10^6]$ . In addition to the total pending or buffered tuples, we measure *average production delay*, total CPU time, total disk time and maximum total window size. Unless otherwise stated, the default values used in the experiments are as given in Table 2.

## 4.2 Experimental Results

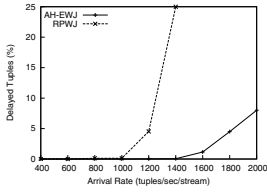
In this section, we present a series of experimental results for assessing the effectiveness of the proposed join algorithm. As a baseline algorithm, we use an algorithm termed as RPWJ (Rate-based Progressive Window Join) [19], which extends an existing algorithm RPJ [11]—a variant of XJoin [7]—to process the sliding window joins. The extension is imperative, as RPJ is an algorithm to join only the finite streams. For each set of the experimentations, we run the simulator for 1.6 simulation hours. We start to gather performance data after an startup interval of 12 minutes is elapsed.

We, now, present the experimental results with varying stream arrival rates. With the increase in the arrival rates, more and more tuples can not be processed within the time limit of the stream window; thus, The percent of *idelayed tuples*, that can not be

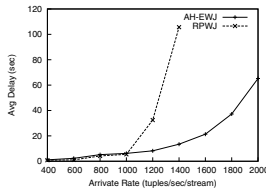
**Table 2.** Default values used in experiments

Parameter	Defaults	Comment
$W_i (i = 1, 2)$	10	Window length(min)
$\tau$	1	slide interval for a Window (min)
$\lambda$	1200	Avg. arrival rate(tuples/sec)
$\alpha$	0.4	Decay parameter for the productivity value
$b$	0.6	burstiness in traces (captured by <i>b-model</i> )
$\rho$	0.4	block eviction threshold
$r$	0.9	disk probe threshold
$M$	20	join memory (MB)
$N_{flush}$	0.4M	flush size for RPWJ
$n_{part}$	60	hash partitions or buckets

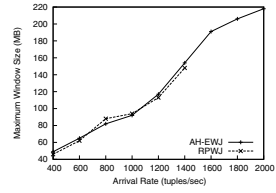




**Fig. 3.** Percentage of delayed tuples vs stream rates



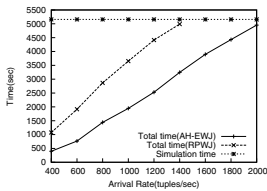
**Fig. 4.** Average delay vs stream rates



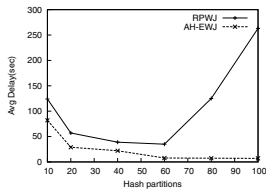
**Fig. 5.** Maximum window size (MB) vs rates

processed within the time limit of the stream window, increases with the increase in the arrival rates as shown in Figure 3. These delayed tuples get expired at a time later than the usual after they get joined with the respective window. For RPWJ, the percentage of the delayed tuples increases sharply with the increase in arrival rates beyond 1000 tuples/sec; however, in case of AH-EWJ, this percent of delayed tuples remains low even for an arrival rate of 1800 tuples/sec/stream (i.e., 3600 tuples/sec in the system). Here it should be noted that, *only arrival rates do not provide the complete indication about the load applied to the system; the load of the join module is also determined by the window size.*

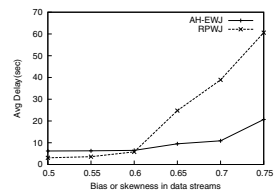
Figure 4 shows, for both the algorithms, the average delay in producing output tuples with the increase in arrival rates. Figure 5 shows the maximum window size during the system activity. Though the allocated memory per stream window is 20MB, spilling the extra blocks onto the disk does not impart significant increase in average output delay of the AH-EWJ even for arrival rates up to 1600 tuples/sec/stream, at a point where maximum total window size is around 190MB (i.e., 9 times the join memory size). Techniques based on load-shedding would have discarded the extra blocks beyond 20MB losing a significant amount of output tuples that would never have been generated. The RPWJ algorithm becomes saturated for an arrival rate 1400 tuples/sec/stream at a point where the average delay attains a very high value. Hence, for a large window, the proposed technique attains a low average delay of output generation; at the same time, the percentage of the tuples missing the time frame set by the window is very low (0.05% for an arrival rate 1400 tuples/sec/stream). This demonstrates the effectiveness of the AH-EWJ algorithm.



**Fig. 6.** Total time (disk-I/O and CPU time) vs rates



**Fig. 7.** Avg production delay vs total hash partitions



**Fig. 8.** Avg production delay vs bias in stream arrivals

Figure 6 shows the total system time (disk I/O and CPU time) with varying per stream arrival rates. We notice that with the RPWJ algorithm the system is saturated

at the rate of 1400 tuples/sec/stream. On the other hand, with the same arrival rate, the AH-EWJ algorithms achieves around 40% reduction in total execution time.

Figure 7 shows the effect of increasing hash partitions on the performance of the two algorithms. Algorithms based on sort-merge joins can not be applied in processing sliding window joins due to temporal order of the tuples [19]. So, increasing the number of hash partitions is a simple approach to lower the processing overhead. But, the performance of the RPWJ algorithm deteriorates with the increase in the hash partitions; this phenomenon renders the RPWJ unfit for the sliding window joins. On the other hand, AH-EWJ attains lower average delay with an increase in the hash partitions. Figure 8 shows the average delay in generating output tuples with varying bias or burstiness of the data stream. As shown in the figure, with the increase in the bias, AH-EWJ performs significantly better than the RPWJ.

## 5 Conclusion

In this paper, we address the issue of processing exact, sliding window join between data streams. We provide a framework for processing the exact results of an stream join, and propose an algorithm to process the join in a memory limited environment having burstiness in stream arrivals. Storing the stream windows entirely in memory is infeasible in such an environment. Like any online processing, maximizing output rate is a major design issue in processing exact join. Hence, we propose a generalized framework to keep highly productive blocks in memory and to maintain the blocks in memory during systems activity, forgoing any specific model of stream arrivals (e.g., age based or frequency based model [5]). The algorithm reduces disk dumps by adapting the sizes of both the windows and the partitions based on, respectively, the stream arrival rates and the productivity of blocks within the partitions. The experimental results demonstrate the effectiveness of the algorithm.

## References

1. Carney, D., etintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.B.: Monitoring streams – a new class of data management applications. In: Proc. Intl. Conf. on Very Large Databases (VLDB), Hong Kong, China, August 2002, pp. 215–226 (2002)
2. Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M.A.: TelegraphCQ: Continuous dataflow processing for an uncertain world. In: Proc. Conf. on Innovative Data Systems Research, CIDR (January 2003)
3. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Processing sliding window multi-joins in continuous queries over data streams. In: Proc. ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), Madison, Wisconsin, USA, June 2002, pp. 1–16 (2002)
4. Gedik, B., Wu, K.-L., Yu, P.S., Liu, L.: A load shedding framework and optimizations for m-way windowed stream joins. In: Proc. Intl. Conf. on Data Engineering (ICDE), Istanbul, Turkey, April 2007, pp. 536–545 (2007)

5. Srivastava, U., Widom, J.: Memory-limited execution of windowed stream joins. In: Proc. Intl. Conf. on Very Large Databases (VLDB), Toronto, Canada, September 2004, pp. 324–335 (2004)
6. Das, A., Gehrke, J., Riedewald, M.: Approximate join processing over data streams. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, San Diego, USA, June 2003, pp. 40–51 (2003)
7. Urhan, T., Franklin, M.J.: XJoin: A reactively-scheduled pipelined join operator. *IEEE Data Engineering Bulletin* 23(2), 7–18 (2000)
8. Mokbel, M., Liu, M., Aref, W.: Hash-merge-join: A non-blocking join algorithm for producing fast and early join results. In: Proc. Intl. Conf. on Data Engineering (ICDE), pp. 251–263 (2004)
9. Viglas, S.D., Naughton, J.F., Burger, J.: Maximizing the output rate of multi-way join queries over streaming information sources. In: Proc. Intl. Conf. on Very Large Databases (VLDB), Berlin, Germany, September 2003, pp. 285–296 (2003)
10. Liu, B., Zhu, Y., Rundensteiner, E.A.: Run-time operator state spilling for memory intensive long-running queries. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Chicago, Illinois, USA, June 2006, pp. 347–358 (2006)
11. Tao, Y., Yiu, M.L., Papadias, D., Hadjieleftheriou, M., Mamoulis, N.: RPJ: Producing fast join results on streams through rate-based optimization. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Baltimore, Maryland, USA, June 2005, pp. 371–382 (2005)
12. Wilschut, A.N., Apers, P.M.G.: Dataflow query execution in a parallel main-memory environment. In: Proc. Intl. Conf. on Parallel and Distributed Information Systems (PDIS), Miami, Florida, USA, December 1991, pp. 68–77 (1991)
13. Dittrich, J.-P., Seeger, B., Taylor, D.S., Widmayer, P.: Progressive merge join: A generic and non-blocking sort-based join algorithm. In: Proc. Intl. Conf. on Very Large Databases (VLDB), Hong kong, China, August 2002, pp. 299–310 (2002)
14. Levandoski, J., Khalefa, M.E., Mokbel, M.F.: Permjoin: An efficient algorithm for producing early results in multi-join query plans. In: Proc. Intl. Conf. on Data Engineering (ICDE), Cancun, Mexico, pp. 1433–1435 (2008)
15. Bornea, M.A., Vassalos, V., Kotidis, Y., Deligiannakis, A.: Double index nested-loop reactive join for result rate optimization. In: Proc. Intl. Conf. on Data Engineering (ICDE), pp. 481–492 (2009)
16. Kang, J., Naughton, J.F., Viglas, S.: Evaluating window joins over unbounded streams. In: Proc. Intl. Conf. on Data Engineering, Bangalore, India, March 2003, pp. 341–352 (2003)
17. Golab, L., Ozsu, T.: Processing sliding window multi-joins in continuous queries over data streams. In: Proc. Intl. Conf. on Very Large Databases (VLDB), Berlin, Germany, September 2003, pp. 500–511 (2003)
18. Chakraborty, A., Singh, A.: A partition-based approach to support streaming updates over persistent data in an active data warehouse. To Appear Proc. IEEE Intl. Symp. on Parallel and Distributed Processing (IPDPS), Rome, Italy, May 2009, pp. 1–11 (2009)
19. Chakraborty, A., Singh, A.: A Disk-based, Adaptive Approach to Memory-Limited Computation of Exact Results for Windowed Stream Joins. Department of Electrical & Computer Engineering, Technical Report UW-ECE #2009-09, University of Waterloo, Canada (2009)
20. Wang, M., Ailamaki, A., Faloutsos, C.: Capturing the spatio-temporal behavior of real traffic data. In: IFIP Intl. Symp. on Computer Performance Modeling, Measurement and Evaluation, Rome, Italy (September 2002)
21. Wang, M., Papadimitriou, S., Madhyastha, T., Faloutsos, C., Change, N.H.: Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In: Proc. Intl. Conf. on Data Engineering, February 2002, pp. 507–516 (2002)

# Prediction Functions in Bi-temporal Datastreams

André Bolles, Marco Grawunder, Jonas Jacobi,  
Daniela Nicklas, and H.-Jürgen Appelrath

Universität Oldenburg, Department for Computer Science

**Abstract.** Modern datastream management system (DSMS) assume sensor measurements to be constant valued until an update is measured. They do not consider continuously changing measurement values, although a lot of real world scenarios exist that need this essential property. For instance, modern cars use sensors, like radar, to periodically detect dynamic objects like other vehicles. The state of these objects (position and bearing) changes continuously, so that it must be predicted between two measurements. Therefore, in our work we develop a new bi-temporal stream algebra for processing continuously changing stream data. One temporal dimension covers correct order of stream elements and the other covers continuously changing measurements. Our approach guarantees deterministic query results and correct optimizability. Our implementation shows that prediction functions can be processed very efficiently.

**Keywords:** Datastream Management, Query Processing, Prediction.

## 1 Introduction and Related Work

Datastream management systems (DSMS) continuously process queries over streaming data. To prove determinism and correctness of query results and to provide optimizability, a DSMS needs a formally defined algebra. However, most existing DSMS prototypes [1–3] have no formally defined algebra. They are used to investigate processing mechanisms for datastreams, but cannot guarantee a deterministic and correct execution of continuous queries. Only PIPES [4] provides a formally defined stream algebra that guarantees deterministic results and correct behavior for query optimization. However, PIPES' relational stream algebra does not support prediction of measurements. This does not fit real world requirements. E. g. in advanced driver assistance systems (ADAS) continuously moving objects have to be tracked. To compare object detections of the last scan period with object detections of the current scan period in these systems the continuously changing states, e. g., position and bearing, have to be predicted to the same point in time. In other scenarios, such as wireless sensor networks or patient monitoring prediction of values is also useful, e. g., for energy saving or earlier hazard detection.

To support prediction functions, on the one hand specialized predictions functions must be developed as done in [5–7]. On the other hand, these prediction functions must be used for processing streams of detected continuously changing values. On this note, especially [8] is related to our work. Here, prediction functions are attached to stream elements to reduce the communication overhead in a wireless sensor network. However,

a clear integration into a query algebra that is necessary for provable correct query processing is missing. Furthermore, this work is restricted to linear or linearized prediction functions. This is not adequate for some scenarios: for example, processing dynamic models with covariance matrices leads to quadratic functions. Therefore, in our work we integrate arbitrary prediction functions into the algebra of a DSMS. For this, we use a bi-temporal model based on [9]. The first temporal dimension in our model is the stream time to cover the correct temporal processing order of elements (cf. [4]). The second dimension is the prediction time to cover continuously changing values. Our contribution is twofold:

- We formally define the semantics of our query algebra. We also show that our algebra can benefit from optimization rules.
- We present an efficient implementation of our query algebra in a DSMS.

The rest of this paper is structured as follows: We present an example scenario to illustrate prediction enabled query algebra requirements in Section 2. This algebra is formally defined in Section 3. We show the evaluation of our approach in Section 4 and give a conclusion in Section 5.

## 2 Stream Processing with Prediction Functions

This section illustrates the requirements and concepts for query processing over streams with prediction functions. For this, see the example query shown in Listing 1 that continuously compares the distance between the own vehicle (ego-car) and vehicles ahead to avoid a rear end collision.

---

```

1 SELECT ego.pos FROM ego RANGE 10 seconds, s2 RANGE 15 seconds
2 WHERE ego.speed > 30 AND s2.pos - ego.pos < 15
3 SET PREDFCT ego.pos:=ego.pos+ego.speed*t WHERE true
4       s2.pos:=s2.pos+s2.speed*t WHERE s2.type == "vehicle"

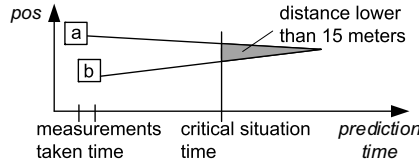
```

---

**Listing 1.** Example Query

Measurements from two sensors are processed. Sensor `ego` provides measurements about the ego-car and sensor `s2` detects objects ahead of the ego-car. This query compares the position of the ego-car with the positions of detected objects ahead. `SET PREDFCT` defines prediction functions to predict the continuously changing positions. All measurements from `ego` are predicted with the same prediction function while measurements from `s2` are only predicted, if they represent a moving vehicle (expressed by `WHERE` in the `SET PREDFCT` clause). Figure 1 illustrates prediction function evaluation. Measurement values `a` and `b` are not compared directly, but the outcomes of their prediction functions are. So, the points in time are determined when the difference between their predicted values gets under a specified threshold (e. g., 15 meters).

The straight forward approach to process the example query (probably used in [8]) would attach matching prediction functions to each stream element and solve them for



**Fig. 1.** Predicting specific situations

each query predicate evaluated over the element. However, in this case, multiple solving of prediction functions is necessary. To avoid this, we use prediction functions to redefine query predicates *before* query processing starts. The prediction functions  $ego.pos := ego.pos + ego.speed * t$  and  $s2.pos := s2.pos + s2.speed * t$  redefine the query predicate  $s2.pos - ego.pos < 15$  from our example query into  $s2.pos + s2.speed * t - (ego.pos + ego.speed * t) < 15$ . This inequality is then solved for time (denoted by variable  $t$ ), which in our example leads to the following four solutions for  $t$  with mutually exclusive guards:

$$s2.pos + s2.speed * t - (ego.pos + ego.speed * t) < 15 \Leftrightarrow \begin{cases} t < \frac{15 - s2.pos + ego.pos}{s2.speed - ego.speed} & \text{if } s2.speed - ego.speed > 0 \\ t > \frac{15 - s2.pos + ego.pos}{s2.speed - ego.speed} & \text{if } s2.speed - ego.speed < 0 \\ t & \text{if } s2.speed - ego.speed = 0 \wedge 15 - s2.pos + ego.pos > 0 \\ \emptyset & \text{if } s2.speed - ego.speed = 0 \wedge 15 - s2.pos + ego.pos \leq 0 \end{cases}$$

If there is more than one prediction function, query predicates will be redefined for each of these prediction functions. To decide which redefined predicate to use, in our solution a key is attached to each stream element that points to the corresponding prediction function.

To evaluate a redefined predicate each guard is checked for each input element. If the guard is true, the expression for  $t$  will be evaluated and used to determine the interval in which the query predicate is true. If multiple predicates are combined by AND/OR, the resulting intervals must be combined by pairwise intersection (AND) and union (OR) of the resulting intervals. Sorting the guards according to their frequency to be true can be used for performance optimization (see Section 4).

The intervals returned by redefined query predicates represent the prediction time that describes when a query predicate will be true according to the defined prediction functions. The order of stream elements and by this the windows over which the query is evaluated (cf. [4]) are not affected by these intervals. This is another temporal dimension we denote with stream time. To cover both prediction time and stream time, we use a bi-temporal model similar to [9]. In this model, the two dimensions are orthogonal. This is necessary, since stream time and prediction time do not necessarily correlate, e. g., due to network delays or system overload. If we had only one temporal dimension, predictions would cause a reorder of stream elements. Newer elements would eventually be processed before older ones and therefore deterministic results could not be guaranteed any more (e. g., a join cannot decide to prune elements). Figure 2 illustrates this 2-dimensional time domain.

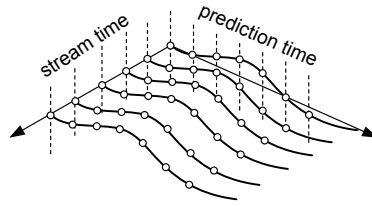


Fig. 2. Our discrete bi-temporal datastream model

Each element in the result of the query in Listing 1 describes *when* the situation expressed by the predicates will be valid. An element with stream time  $t_S$  and prediction time  $t_P$  for example tells us that we already know at time  $t_S$  that something will happen at time  $t_P$ . This information reaches the application before all elements with start timestamps (stream time) greater than  $t_S$ . Maybe, later elements (stream time) will give other information. However, this problem exists in each prediction scenario and is therefore not specific to our system.

### 3 Bi-temporal Stream Algebra

To guarantee deterministic results of prediction enabled queries and to take benefits of well-known relational optimization heuristics, the underlying query algebra must be formally defined. We first define our datastream model on which our algebra is based (Section 3.1) and then describe the semantics of the operators of our algebra (Section 3.2).

#### 3.1 Datastream Model

The semantics of our operators are defined on logical datastreams similar to [4]. By this we can show the reducibility of our algebra to the well-defined relational algebra due to snapshot-reducibility.

**Logical Datastream.** A logical stream  $S^L$  is a set of elements  $r^L = (e, t_S, t_P, i)$ , where  $e$  is a relational tuple containing payload data.  $t_S$  (stream time) and  $t_P$  (prediction time) are timestamps of the finest granularity in a discrete time domain  $\mathbb{T}$  [10]. The circles in Figure 2 illustrate the points in time in our bi-temporal datastream model.  $i$  is an identifier describing the measurement from which the values in  $e$  are taken. If a sensor measurement is valid for  $n$  time instants in stream time, there exist  $n$  elements in a logical stream with different values for  $t_S$ , but the same value for  $i$ .

#### 3.2 Operators

We first define a new prediction assignment operator. Then standard operators with predicates are adapted to prediction functions. Operators with no predicates like mapping, projection and set operators can also be defined over our bitemporal datamodel. However, due to space reasons we omit details here.

**Prediction Assignment.** To process prediction functions, we need to assign predicted values to logical stream elements. Our prediction assignment operator  $\rho$  is parameterized by a function  $f_s$  that decides which prediction function  $f_p$  to use for each element (WHERE clause in SET PREDFCT clause). The semantics is defined by  $\rho(S_L) = \{(\hat{e}, t_S, t_P, i) | \exists(e, t_S, t_P, i) = r \in S_L \wedge f_s(r) = f_p \wedge f_p(t_P, i) = \hat{e}\}$ .

For each point in prediction time and each logical stream element this operator calculates the prediction function and sets the corresponding value for that element. Without prediction functions the values would be constant over prediction time. With prediction functions at each point in prediction time other values can be valid. We can integrate arbitrary prediction functions into  $\rho$ .

**Selection.** A selection operator  $\sigma$  decides for each element, whether it should be in the output stream of the operator or not. Our selection operator behaves slightly different. It finds out for each element when the selection predicate  $p$  will be true according to the prediction function used for the element. So, the result of a selection predicate is not a boolean value, but a set of time intervals in which the selection predicate is true. This is shown in Figure 3. Each logical stream element is evaluated separately as defined by  $\sigma(S_L, p) = \{(e, m, t_S, t_P, i) | (e, m, t_S, t_P, i) = r \in S^L \wedge p(r)\}$ .

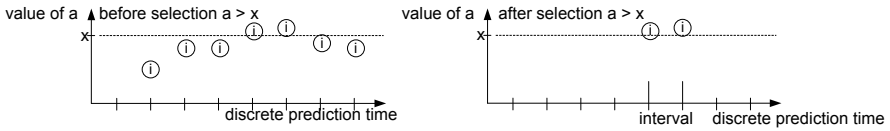


Fig. 3. Semantics of selection

**Join.** A join  $\bowtie$  evaluates elements that are valid at the same stream time  $t_S$ . Similar to the selection predicate a join predicate  $p$  returns a set of intervals in which the join predicate is fulfilled. The formal definition of our join operator again shows that each logical element is evaluated separately:  $\bowtie(S_{L,0}, S_{L,1}, p) = \{(\bowtie_e(e_0, e_1), t_S, t_P, \bowtie_i(i_0, i_1)) | \exists(e_k, t_S, t_P, i_k) \in S_{L,k}, k \in \{0, 1\} \wedge p((\bowtie_e(e_0, e_1)))\}$ . The result contains the merge  $\bowtie_e$  of the payload data and the merge  $\bowtie_i$  of the identifiers.

### 3.3 Algebraic Optimization

Algebraic optimization allows switching operators in a query plan without changing the result of the query. Many equivalence rules, on which algebraic optimization is based, exist for the relational algebra. In this section, we show that our algebra can benefit from relational equivalence rules. For this, we define a bi-temporal snapshot at point time  $t = (t_S, t_P)$  of a logical datastream by the snapshot operator  $\tau$  (cf. [4, 9]):  $\tau_t(S^L) := \{(e, m, i) | (e, m, t_S, t_P, i) \in S^L\}$

A snapshot of logical stream  $S^L$  is the nontemporal set of all elements being valid at the 2-dimensional point in time  $t = (t_S, t_P)$ . Since our operators selection, join, projection and set-operators evaluate each logical stream element separately, they are snapshot-reducible. Thus, relational equivalence rules are valid in our algebra. Due to space limitations detailed proofs are omitted here.



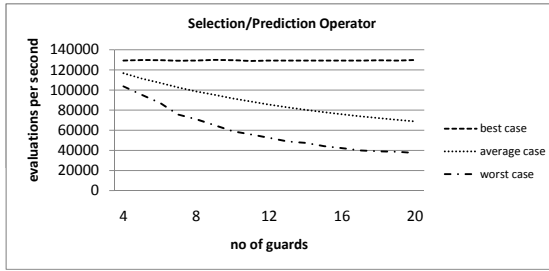


Fig. 4. Performance of Prediction Assignment and Selection

## 4 Evaluation

That prediction functions can increase accuracy of query results has already been shown in [5, 8, 11]. Furthermore, the results of algebraic optimization heuristics are also well-known. Thus, due to space limitations in our evaluation we concentrate on the performance of our algebraic approach. We show that our algebra is suitable for high-volume stream processing.

We implemented our algebra in our DSMS Odysseus [12] and used the predicate preprocessing step shown in Section 2. We ran our evaluation on an Intel Core 2 DUO at 2.6 GHz with 3 GB RAM. The performance of our operators depends on the guard expressions to evaluate. We used combinations of  $+$ ,  $-$ ,  $\cdot$ ,  $\div$  and  $\sqrt{\phantom{x}}$ , which is a suitable set of operations for many application scenarios.

The performance of the selection operator is shown in Figure 4. Here, you can see the number of evaluations possible in a selection operator. By evaluation we mean the evaluation of a selection predicate (with multiple guards, see section 2). Since different prediction functions with different polynomial degrees lead to different numbers of guards to be evaluated in a selection predicate, we tested the performance of our selection operator with up to 20 guards (a quadratic function has 12 guards). Figure 4 shows that performance depends linearly on the number of guards in a selection predicate. Here you can also see three cases. The worst case means that the guard that is true is always evaluated last. With linear prediction functions (4 guards), we then have a performance of about 105000 evaluations/sec (for comparison a selection with the predicate  $a.i.d \geq 40$  can be evaluated approx. 300000 times per second). The best case means that the guard that is true is always evaluated first. In this case we have performance of about 130000 evaluations/sec. This enough in most scenarios, especially for advanced driver assistance systems where sensors usually sample at 25 Hz. Furthermore, this is for instance orders of magnitude faster than the processing in [8], due to our redefinition of predicates.

The prediction assignment operator has the same behavior as the selection operator. Here, for each possible prediction function there exists a guard that must be evaluated before the prediction function can be assigned to a stream element.

The performance of our join operator also depends linearly on the number of guards to check. However, the join operator is a little bit slower than the selection. This is,

because the expressions to evaluate are larger, since they contain the prediction functions of at least two streams (or more in the case of a multiway-join). However, in the worst case and at 20 guards, we still reach a performance of 20000 evaluations/sec.

The performance can be improved by guard reorder. Here, we determine the frequency of the guards to be to be true and evaluate the most frequent guards first. Using this, we theoretically can achieve best case performance, however this depends on the stream elements.

**Conclusion of Evaluation Results.** The above evaluation shows that our approach is suitable for stream processing. Although not presented in this paper, using the moving objects data generator [13] we also found out that prediction functions are useful for identifying specific future traffic situations.

## 5 Conclusion

To the best of our knowledge, no work exists that integrates prediction functions into the query algebra of a datastream management system. Therefore, in this paper, we present a novel bi-temporal relational stream algebra that can process prediction functions. Our approach uses two dimensions of time, stream time for efficient processing of unbounded streams and prediction time for representing continuously changing measurement values. Our logical definition of datastreams and the corresponding algebra operators allow to reduce our algebra to the nontemporal relational algebra. Therefore, our algebra has the same characteristics as the relational algebra, especially optimization heuristics known from the relational algebra can be used in our algebra, too.

Our evaluation shows that a high throughput can be achieved, if prediction functions are embedded into the predicates (redefinition).

Future work concentrates on the integration of non-snapshot-reducible object tracking operators and their interaction with relational operators.

## References

1. Arasu, et al.: STREAM: The Stanford Stream Data Manager. *IEEE Data Engineering Bulletin* 26(1) (2003)
2. Abadi, et al.: The Design of the Borealis Stream Processing Engine. In: *CIDR 2005* (2005)
3. Chandrasekaran, et al.: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In: *CIDR 2003* (2003)
4. Krämer, J., Seeger, B.: Semantics and implementation of continuous sliding window queries over data streams. *ACM Trans. on Database Syst.* 34(1) (2009)
5. Morzy, M.: Mining frequent trajectories of moving objects for location prediction. In: Perner, P. (ed.) *MLDM 2007*. LNCS (LNAI), vol. 4571, pp. 667–680. Springer, Heidelberg (2007)
6. Goel, S., Imielinski, T.: Prediction-based monitoring in sensor networks: taking lessons from MPEG. *SIGCOMM Comput. Commun. Rev.* 31(5) (2001)
7. Lee, C.C., Chiang, Y.C., Shih, C.Y., Tsai, C.L.: Noisy time series prediction using m-estimator based robust radial basis function neural networks with growing and pruning techniques. *Expert Syst. Appl.* 36(3) (2009)
8. Ilarri, S., Wolfson, O., Mena, E., Illarramendi, A., Sistla, P.: A query processor for prediction-based monitoring of data streams. In: *EDBT 2009*. ACM, New York (2009)

9. Snodgrass, R.T., Kucera, H.: The TSQL2 Temporal Query Language. Kluwer Academic Publishers, Dordrecht (1995)
10. Bettini, C., Dyreson, C.E., Evans, W.S., Snodgrass, R.T., Wang, X.S.: A glossary of time granularity concepts. In: Temporal Databases, Dagstuhl (1997)
11. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and indexing of moving objects with unknown motion patterns. In: SIGMOD 2004. ACM, New York (2004)
12. Jacobi, J., Bolles, A., Grawunder, M., Nicklas, D., Appelrath, H.J.: A physical operator algebra for prioritized elements in data streams. Computer Science - Research and Development 1(1) (2009)
13. Brinkhoff, T.: Generating network-based moving objects. In: SSDBM 2000. IEEE Computer Society, Los Alamitos (2000)

# A Load Shedding Framework for XML Stream Joins

Ranjan Dash<sup>1</sup> and Leonidas Fegaras<sup>2</sup>

<sup>1</sup> University of Texas at Arlington  
416 Yates Street, P.O. BOX 19015  
Arlington, TX 76019

Ranjan.Dash@Mavs.Uta.edu

<sup>2</sup> University of Texas at Arlington  
416 Yates Street, P.O. BOX 19015  
Arlington, TX 76019

Fegaras@cse.uta.edu

**Abstract.** Joining data streams using various types of windows is an established method of stream processing. The limitation of window size due to memory constraint takes a heavy toll on the accuracy of the query result. Through this paper, we propose a unique windowing technique based on innovative cost functions for join query processing under memory constraints. The logical window construction is controlled through unique data structure and maintained using load shedding technique with least overhead. We applied our technique on XML streams domain and proved the effectiveness of our strategy through measuring the accuracy of the result from joining two XML streams using standard XQuery. With assumption of acceptability of an approximate solution with acceptable error bound in the face of unbounded, complex XML stream, we have tried to come up with a low overhead architecture for load shedding and tested its usefulness through a set of cost functions.

**General Terms:** Management, Measurement, Performance.

**Keywords:** Data Streams, XML Streams, Load Shedding, Stream Joining, Synopsis, Approximate Query Processing, Quality of Service.

## 1 Introduction

XML stream joins have been widely used in various systems such as Publish/Subscribe systems [7, 10], message brokers in Web service based architectures etc. With rapid growth of xml stream sources such as RSS feeds etc. and ubiquitous handheld devices, there is a need to merge these stream sources at the client end rather on the server end to produce customized results. Thus the xml streams are need to be join processed with limited resources such as memory or CPU [3].

Usually the memory limitation is addressed through maintaining continuous slide windows for various streams and applying join between them. The windows can be either *time based* or *tuple based* [8]. Also limitations on memory and CPU can be addressed through various data stream summarization algorithms such as quantiles,

histograms, sketches, wavelets etc. Due to limitation on size of the window, the result is approximate and solution paradigms such as max subset or random samplings etc are resorted to.

In this paper we focus on problem of load shedding for continuous queries that contain equi-joins on multiple XML streams. We strive to optimize the accuracy of query result through innovative method of maintaining windows that is built on the concept of relevance rather than time or frequency and applying join between them. We attempt to propose a framework that is built on the basis of both frequency based and age based model [6]. It draws its strength from both models through using a cost function that addresses the best of both families.

## 1.1 Related Work

As we have not found any work that addresses the load shedding for XML stream joins, we have limited our study on load shedding on relational streams. The problem of joining streams under resource constraints has been addressed for relational streams [4, 5, 6, 12] using windowed joins. Solution through Load shedding in relational stream joins has been addressed by [4, 6] whereas through randomized sketch in [5]. But ours is the first paper in XML streams that tries to address the resource constraint through load shedding.

Multi-join algorithms with join order heuristics has been proposed by [9] based on processing cost per unit time. The concept of load shedding in case of multiple pair wise joins is discussed in [5]. It uses a light sketching technique to estimate the productivity of each incoming tuple and thereby taking decisions to shed or keep on the basis of it. Their methods of *MSketch*, *MSketch-RS* and *MSketch\*Age* have substantial productivity gain over other loadshedding methods such as *Random* and *Aging* methods. The problem of memory-limited approximation of sliding-window joins for relational streams has been discussed in [6]. They have proposed a age-based model that addresses the max-subset problem and a random sample method on join results for aggregate queries.

Joining XML streams for a pub/sub systems is proposed through *Massively Multi-Query Join Processing* technique [7]. It covers the problem of processing a large number of XML stream queries involving value joins over multiple XML streams and documents.

In our previous paper [1] we dealt with problem of XML stream shedding for single source suggesting two approaches, one syntactic and another semantic approach. We have implemented a simple random load shedder and a structured predicate load shedder. The earlier one simple in its implementation keeps the structural integrity of the data while the later is based on structural synopsis of the stream data. We also extended the solution to predicate queries by constructing and maintaining value synopsis of elements.

## 1.2 Paper Main Result and Outline

The present paper tries to find a solution framework for join queries on multiple XML streams using both structural and value synopsis. It extends concept of sliding

window to a window of fixed size that sheds tuples based on relevance rather than time or frequency.

In this paper, we formulate a strategy for load shedding in XML streams for Join Queries. We propose a novel cost based element expiration. We limit our discussions to binary joins using a fixed window that employs the shedding on arriving at steady state. The cost based victim selection will be our main process to maintain the window and thereby ensuring fixed memory utilization.

Section 2 provides an overview for our XML join query processing model. Section 3 discusses various ways to maintain structural and value synopses for XML data stream. Section 4 gives an in-depth view of our proposed load shedding techniques for binary sources of XML data stream. In Section, 5 we provide experimental results that show the effectiveness of our proposed load shedding schemes. Finally we conclude in Section 6.

## 2 The XML Join Query Processing Model

XML stream  $S_i$  is a stream of events like SAX events or document fragments arriving at the system with varying rate. Unlike the relational stream having well defined data units such as tuples, the XML stream system does a small preprocessing to convert these streams or fragments into regular xml nodes before running queries on them. There are standing queries that run in the system on these tokenized regular XML elements as they become available in the system.

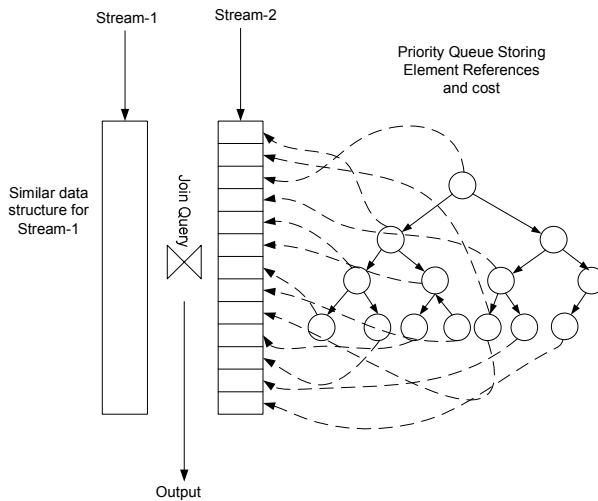


Fig. 1. Overall XML Stream Join Query Processing Architecture

Due to continuous nature of stream our system depends on a logical window  $S_i[W_i]$  that provides the necessary basis to run join queries. We limit our discussions to two streams  $i = 1,2$ . The size of the window  $S_i[W_i]$  is determined by the size of the

memory from resource limitation point of view. As the window is not a time based or a tuple based one, it determines its content through relevance. The join query result is denoted as  $T = S_1[W_1] \times S_2[W_2]$ .

## 2.1 Frequency, Age and Relevance Based Stream Model

As mentioned in [6] the frequency based model does not work for all kinds of streams. It fails for streams where the relevance distribution is skew over the life of the tuple as in the case of online auction scenario where more bids come towards closing of an item.

Similarly age based models do require more monitoring and frequent adjustment to window mechanism to yield max sub set result [6]. Depending on the age curve shape different strategies to be followed to optimize the result. On the other hand if time based window model is followed, it will result in decreased productivity due to discard of relevance tuples when the resource is limited.

Hence the solution to optimize the productivity in a resource crunched situation is to collect all relevant tuples while discarding the irrelevant ones. The relevance is decided based on its participation in a join process.

## 2.2 Limiting Resources - Relevance Based Window

To address all these deficiencies in existing stream models and to achieve the max sub set result or optimize the productivity we have resorted to a new window model. The model ensures the high usefulness of all stored tuples while making a judicious decision on load shedding. We have come up with a framework to measure relative relevance of various elements and there by making the shedding decisions wisely. We have formulated a unique cost function that is central to this model and an innovative data structure that is very efficient in implementing the framework. The effectiveness of the approach is measured objectively as described in section 5.

## 2.3 Limiting Resources – Innovative Data Structure

The window construction though a part of the overall system, yet an overhead eats out valuable resources such as memory and CPU. We have tried to make this extra layer as lean as possible with least possible processing effort or CPU cycle churning.

Besides, we have implemented the symmetric hash join as the join mechanism to make the system more efficient from both time and space point of view.

# 3 Synopsis for Limited Resource

The main idea of this paper is to make best utilization of limited resources such as memory and CPU by making best judgment to prepare an intelligent synopsis for each stream that guides the load shedding decision. The synopsis construction is light and has least overhead for its maintenance. The heart of the synopsis is the construction and maintenance of the heap based priority queue with relevance measure of elements of the stream. The relevance measure of each element is calculated using the cost function that is described in the following section. Each time an element comes in any

stream the cost of it is calculated and updated or inserted into the heap based on its pre-existence in the heap. So also the cost of all nodes gets updated at this instance. The heap is kept updated with the node with least value at the root, ready to be shedded if need arises.

### 3.1 Cost Function

The calculation of weight for each element in the logical queue is done using the following cost function [Equation 1]. This has two significant parts (a) Age factor and (b) Utility factor. The age component is derived from forward decay based model [11] which is in turn based on the philosophy that old data is less important. The utility part is derived from the intuition that any tuple or element that takes part successfully in a join is useful. The degree of usefulness or relevance is based on the context of the stream. A stream that has time sensitive data might have lower utility for future joins even if it has taken part in the join at present. So depending on the type of stream these factors contribute differently towards the over all relevance. The cost of relevance for an element  $e_i$  of stream  $S_i$  that has arrived at time  $t_i$  and measured at time  $t$  such that  $t \geq t_i$  and  $t_i > L$  is

$$C = C_1 f(t_i, t) + C_2 f(u_i) \quad (1)$$

Where

$f(t_i, t) = g(t_i-L)/g(t-L)$  a decay function for an element with a timestamp  $t$  and landmark time stamp  $L$

$f(u_i) =$  Count of times that this element has been part of the join

$C_1, C_2$  are arbitrary constants whose values can be tuned based on data type of XML stream.

#### 3.1.1 Age Based Relevance

This part influences the overall relevance through age based load shedding model. Based on the stream context we are using the following three decay functions that varies between 0 and 1.

**Linear Decay.** This follows a linear distribution for the weight with any element that arrives at the time of measuring is having a weight of 1 and that has arrived at the beginning of the system start (Landmark time  $L$ ) with a weight of 0. The function is as follows.

$$f(t_i, t) = (t_i-L)/(t-L) \quad (2)$$

At  $t = t_i$  the weight is 1 and as  $t$  increases it decreases linearly.

Most of the normal streams follow this decay pattern.

#### Polynomial Decay

$$f(t_i, t) = (t_i-L)^2/(t-L)^2 \quad (3)$$

**Exponential Decay.** We have found out much of resource (both space and time) can be freed up using the exponential decay with reasonable impact on the recall.



$$f(t_i, t) = \exp(t_i - t) \quad (4)$$

### 3.1.2 Utility Based Relevance

The second part of the cost function comes from the utility of each element from their participation in join operation. If the element is a subset of the join result it bumps the count. We have implemented this part as a simple count in our experiment. This part represents a simple form of the output history. More accurate weights can be calculated based on the timestamps of its appearance in the result stream.

## 3.2 Relevance Based Window

The priority queue that is implemented as a heap acts as a logical window for the input stream. Its size determined by the memory availability decides when to start the shedding. The victim selection is facilitated through the heap structure to shed the lowest weight element at the root. As the relative weight of each element node driven by the choice of cost functions discussed in section 3.1.1 and 3.1.2, the age of the element and number of times its appearance in the result stream play a crucial role in determining which element to be kept in the window irrespective of their arrival.

## 4 Our Shedding Mechanism

Our basic algorithm for executing join  $S_1[W_1] \times S_2[W_2]$  is shown in Figure 2. If memory is limited, we need to modify the algorithm in two ways. First, in Line 2, we update  $S_1[W_1]$  in addition to  $S_2[W_2]$  to free up memory occupied by expired tuples. More importantly, in Line 4, memory may be insufficient to add  $s$  to  $S_1[W_1]$ . In this case, we need to decide whether  $s$  is to be discarded or admitted into  $S_1[W_1]$ , and if it is to be admitted, which of the existing tuples is to be discarded. An algorithm that makes this decision is called a *load-shedding strategy* [1, 2, 4]. Due to load-shedding, only a fraction of the true result will actually be produced. We denote the fraction of the result tuples produced as *recall*.

We adopt a window based approach to process the join query. The window in our case is a fixed size buffer for each source stream. The data is kept in the form of a heap data structure of elements sorted by the cost of each element. The least cost element remains at the top ready to be shed. The shedding action is triggered as a complete element arrives at the source. Upon arrival, the cost is updated for the new element if it matches with an existing element in the heap, else the new element is added to the heap. If the action is update, the heap is re-sorted through heapify operation. On insert, the shedding decision is taken if the buffer is already full. If the buffer is not full, the element is added to the heap and heapified. The shedding is decided if the cost of the new element is more than the cost of the element at the top of heap. Else the new element is dropped without any shedding. The shedding enacts the deleting of the element at the top and adding of new element to the heap. Once again the heap gets heapified after the operation completes.

## 4.1 Join Processing

For simplicity, we have processed the join between two data stream sources. We implemented our join query processing as a symmetric hash join between our two window buffers. The reference of these elements has been maintained in respective hash tables for faster access.

-----  
 $N_i$ : Max size of logical buffer  $S_i$  [ $W_i$ ]  
 $Q_i$ : Associated relevance queue of stream  $S_i$   
 $B_i$ : Associated hash based buffer of stream  $S_i$   
 $f_i$ : Hash functions for streams  $S_i$ . For simplicity we kept  $f_1 = f_2$   
 $attr_i$ : join attribute of stream  $S_i$

1. When a new element  $e$  arrives in any stream  $S_1$ -

### Phase I: Symmetric Hash Join

2. Calculate hash value of  $e$  by applying  $f_1$  on  $attr_1$  and insert in  $B_1$ .
3. Calculate hash value of  $e$  by applying  $f_2$  on  $attr_2$
4. Probe  $B_2$  using the value from step 3.
5. Emit the result

### Phase II: Synopsis Construction

6. If size of  $Q_1 < N_i$ , Insert into  $Q_i$
7. Else
  - a. calculate cost of  $e$   $C_e$
  - b. If  $C_e < \text{cost of element at head of } Q_1$ , throw  $e$  and remove the corresponding element from  $B_1$
  - c. Else shed head of  $Q_i$ , Insert  $C_e$  into  $Q_i$
8. Similarly repeat the steps for any element that reaches in stream  $S_2$  symmetrically with converse data structures.

-----  
**Fig. 2.** Algorithm XJOIN - Join Execution Steps

## 5 Experiments

In this section we present the results that we get from the system that we implemented based on the framework that we presented in this paper. We compare the results with that from two other systems that deal with non-stream data. They are Stylus Studio [13] and SQL Server 2005 [14]. Our comparison is mostly based on three factors. One, the over all quality of the result, known as recall or productivity. Second, the overall memory consumption by system. Third, the time of processing.

Our implementation is in Java and we ran the systems in 2-GHz Intel Core 2 Duo machine with 2.0 GB of main memory running windows XP. We tested three different implementations of our cost function (linear, polynomial and exponential)

for the test. Furthermore, our experimental results reveal several other properties and characteristics of our embedding scheme with interesting implications for its potential use in practice.

We tested our framework on representative datasets derived from synthetic and real-life datasets. The size of the dataset is controlled to avoid the memory limitations of the systems used [13, 14].

## 5.1 Synthetic Data Sets

We used the synthetic data from XMark [17] XML data benchmark that is modeled on the activities of an on-line auction site ([www.xml-benchmark.org/](http://www.xml-benchmark.org/)). We controlled the size of the XMark data to 50 MB using the scaling factor input to the data generator. The ceiling of 50 MB is considered due to the Java heap space for Stylus Studio. We ran a set of join queries similar to the following one on all three implementations (ours, Stylus Studio and SQL Server 2005). The queries have been modified to suit the implementation [14].

### Sample XQuery

```
let $auction := doc("auction.xml") return
for $p in $auction/site/people/person
let $a :=
  for $t in $auction/site/closed_auctions/closed_auction
  where $t/buyer/@person = $p/@id
  return $t
return <item person="{ $p/name/text()}">{count($a)}</item>
```

### 5.1.1 Effect of Decay

Our first set of experiment is to see the effect of decay algorithm on the productivity. We calculated the number of output tuples for each of the three algorithms and compared it with the exact query recall to compute the accuracy. They are measured for each of the memory size. The 100% of memory size refers to no load shedding having buffer equal to the size of the stream; in our case of 50 MB for each stream. The other memory sizes are reduced according to the ratio. Figure 3 shows the relative quality of the result for different implementations of the load shedding mechanisms. The *Exponential Decay* based cost function produces better result for almost all memory sizes. The stream characteristic best suits the decay function. Figure 4 indicates the better processing time for *Linear Decay* based cost function relative to other two implementations due to less maintenance overhead of cost calculation for shedding. As the amount of shedding decreases for higher memory sizes the gap narrows down.

## 5.2 Real Life Data Sets

We use the real life XML data sets, DBLP [16] for the following queries. We fragment it into two parts DBLP1 and DBLP2 based on journal and conference series [18]. We adjusted the file size to 50 MB each to make the joining load even. Once again we ran it through all five systems; three of our own implementation, Stylus

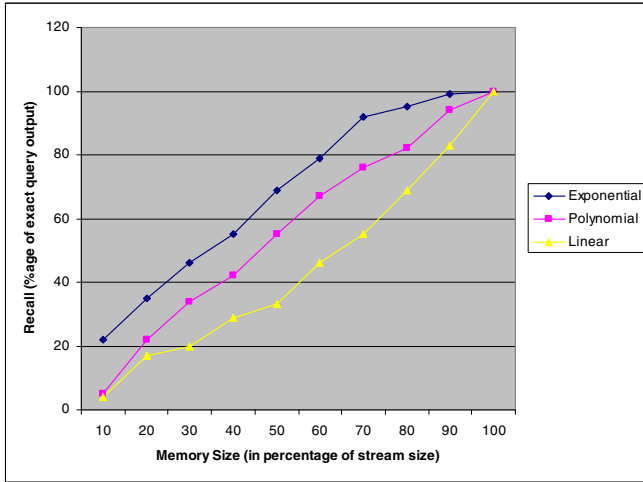


Fig. 3. Effect of Cost Function on Productivity for synthetic XMark data

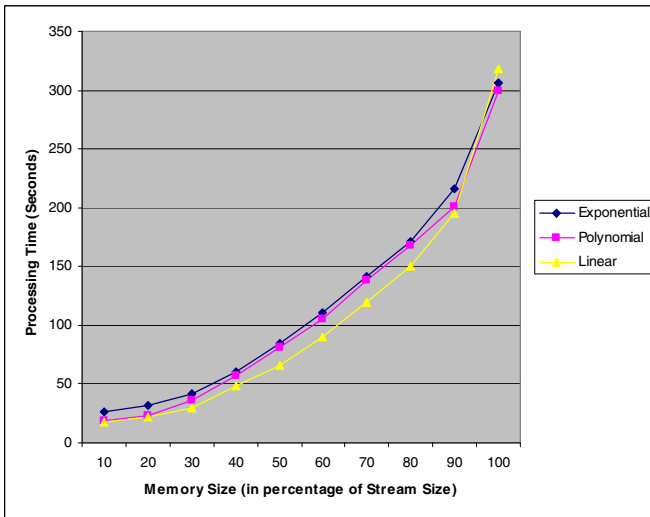


Fig. 4. Effect of Cost Function on Processing Time for synthetic XMark data

Studio and SQL Server 2005. On this dataset, we use the following XQuery template that asks for authors that have published in at least 2 different journals and/or conference series:

```

for $a1 in doc("dblp1.xml")//author,
   $a2 in doc("dblp2.xml")//author
where $a1/text() = $a2/text()
return $a1

```

Similar to synthetic data we plotted the results for accuracy for all of our three implementations. The recall percentage is calculated with respect to the output that is acquired from SQL Server 2005. The result is presented in Figure 5. However as the data is no more dependent on the time or not temporal in nature, the type of decay cost function has relatively less effect on the recall. Rather the linear function has better effect compared to other two implementations due to its simpler implementation. The high fan out characteristic of DBLP might have contributed to this implementation.

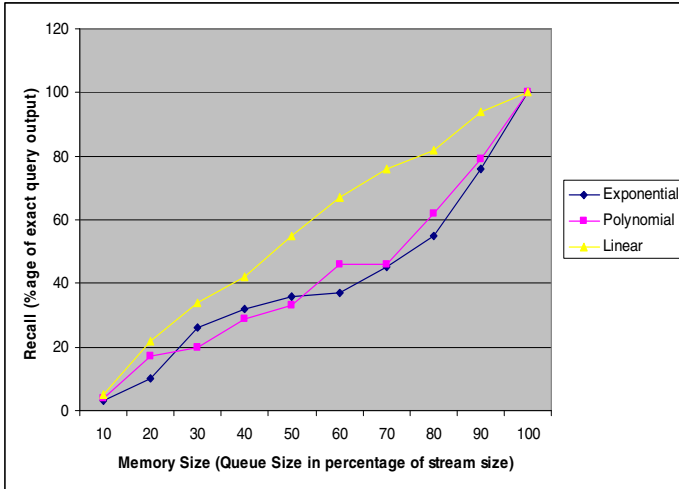


Fig. 5. Effect of Cost Function on Productivity for real life DBLP data

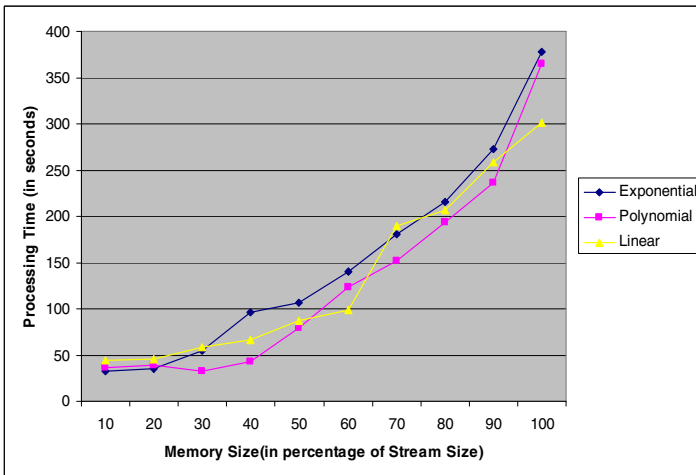


Fig. 6. Effect of Cost Function on Processing Time for real life DBLP data

The processing time is calculated for various cost function implementations for all ten memory sizes and plotted in Figure 6. It is quite clear that the linear implementation provides better timing relative to other two implementations. Combining both the recall study and the processing time, it is evident that the linear costing function based load shedding strategy is the best one out of the three for dblp dataset.

## 6 Conclusion and Future Work

The major focus of this paper is to assure the quality of service in the face of limited resources and query complexity such as joining. We tried to prove the effectiveness of our approach by implementation of a logical window that extends over the whole stream by shedding low cost nodes. We have shown the efficiency of our approach by using various cost functions that gave us tools to attenuate the data characteristics of the input stream. Though we have explained the framework using two streams, but this framework could also be extended to commutative joins between multiple streams.

We could have compared our experimental result with a simple system that implements a fixed element based or time based sliding window. But intuitively, it would have been similar to a simple shedding implementation of *first in first out* without any relevance to element's probable effectiveness in join state and would have resulted in less productivity.

We have excluded the several other types of joining queries from our framework, such as predicate based joins, aggregation queries and quantiles. We are in the process of implementing this framework for aggregation join queries. The load shedding in case of predicate based join queries can be best addressed through combination of our previous work on semantic synopsis construction [15] and this relevance synopsis that is covered in this paper. Also we have not studied the effect of time on utility based relevance factor that is included in all of our cost functions. Similar to age based relevance factor the costing of this part could have been changed based on its time stamp of participation in the result. We are also planning to expand the cost functions to include this output history and other parameters such as element size and inter-arrival characteristics etc.

## References

1. Fegaras, L., Dash, R., Wang, Y.H.: A Fully Pipelined XQuery Processor. In: XIME-P (2006)
2. Polyzotis, N., Garofalakis, M., Ioannidis, Y.: Approximate XML Query Answers. In: ACM SIGMOD (2004)
3. Ayad, A., Naughton, J., Wright, S., Srivastava, U.: Approximating Streaming Window Joins Under CPU Limitations. In: ICDE 2006 (2006)
4. Gedik, B., Wu, K.-L., Yu, P.S., Liu, L.: A Load Shedding Framework and Optimizations for M-way Windowed Stream Joins. IEEE, Los Alamitos (2007)
5. Law, Y.-N., Zaniolo, C.: Load Shedding for Window Joins on Multiple Data Streams. In: The First International Workshop on Scalable Stream Processing Systems (SSPS'07), April 16-20, Istanbul, Turkey (2007)

6. Srivastava, U., Widom, J.: Memory-limited execution of windowed stream joins. In: VLDB (2004)
7. Hong, M., Demers, A., Gehrke, J., Koch, C., Riedewald, M., White, W.: Massively Multi-Query Join Processing. In: SIGMOD 2007, June 11-14, Beijing, China (2007)
8. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proc. of the 2002 ACM Symp.on Principles of Database Systems, June 2002, pp. 1–16 (2002)
9. Golab, L., Ožsu, M.T.: Processing sliding window multijoins in continuous queries over data streams. In: VLDB (2003)
10. Chandramouli, B., Yang, J.: End-to-end support for joins in large-scale publish/subscribe systems. In: Proceedings of the VLDB Endowment, August 2008, vol. 1(1) (2008)
11. Cormode, G., Shkapenyuk, V., Srivastava, D., Xu, B.: Forward Decay: A Practical Time Decay Model for Streaming Systems. In: ICDE 2009, pp. 138–149 (2009)
12. Das, A., Gehrke, J., Riedewald, M.: Approximate join processing over data streams. In: Proc. 2003 ACM SIGMOD Conf., pp. 40–51 (2003)
13. Stylus Studio – XML Editor, XML Data Integration, XML Tools, Web Services and XQuery, <http://www.stylusstudio.com/>
14. Pal, S., Cseri, I., Seeliger, O., Rys, M., Schaller, G., Yu, W., Tomic, D., Baras, A.: XQuery Implementation in a Relational Database System. In: VLDB 2005 (2005)
15. Dash, R., Fegaras, L.: Synopsis based Load Shedding in XML Streams. In: DataX'09 (2009)
16. Ley, M.: Dblp xml records,  
<http://www.informatik.uni-trier.de/~ley/db/>
17. Schmidt, A., Waas, F., Kersten, M.L., Carey, M.J., Manolescu, I., Busse, R.: Xmark: A benchmark for xml data management. In: Proceedings of the International Conference on Very Large Data Bases, pp. 974–985 (2002)
18. Kader, R.A., Boncz, P., Manegold, S., van Keulen, M.: ROX: Run-time Optimization of XQueries. In: SIGMOD'09, June 29-July 2 (2009)

# Supporting Multi-criteria Decision Support Queries over Time-Interval Data Streams

Nam Hun Park<sup>1,\*</sup>, Venkatesh Raghavan<sup>2</sup>, and Elke A. Rundensteiner<sup>2</sup>

<sup>1</sup> Department of Computer Science, Anyang University, Incheon, Republic of Korea  
nmhnpark@anyang.ac.kr

<sup>2</sup> Department of Computer Science, Worcester Polytechnic Institute, Massachusetts, USA  
{venky, rundenst}@cs.wpi.edu

**Abstract.** Multi-criteria result extraction is crucial in many real-time stream processing applications, such as habitat and disaster monitoring. The ease in expressing user preferences makes *skyline* queries a popular class of queries. Skyline evaluation is computationally intensive especially over continuous time-interval streams where each object has its own individual expiration time. In this work, we propose *TI-Sky* – a continuous skyline evaluation framework. *TI-Sky* strikes a perfect balance between the costs of continuously maintaining the result space upon the arrival of new objects or the expiration of old objects, and the costs of computing the final skyline result from this space whenever a pull-based user query is received. This is achieved by incrementally maintaining a precomputed skyline result space at a higher level of abstraction and digging into the more expensive object-level processing only upon demand. Our experimental study demonstrates the superiority of *TI-Sky* over existing techniques.

## 1 Introduction

A wide array of applications including outpatient health care research and sensor monitoring need to provide real-time support for complex multi-criteria decision support (MCDS) queries. The intuitive nature of specifying a set of user preferences has made *skyline* queries a popular class of MCDS queries [1, 2]. Skyline result enable analysts to specify preferences along several different criteria and to learn about the trade offs between the different criteria.

The increased usage of sensors and RFID networks in various real-world scenarios has increased the availability of data streams [3]. Stream query processing typically deals with incoming data that are unbounded and time-variant (each with its own life span). Time-sensitive applications, such as sensor monitoring require query execution to keep up with incoming objects to produce real-time results.

**State-of-the-Art Techniques.** Skyline algorithms [1] over static databases are not viable for computing continuous skylines over time-interval data streams as they tend to assume that the data is static and rely on having all data a priori organized into indices to

---

\* This collaborative work was conducted as a research visiting scholar at Worcester Polytechnic Institute.



facilitate query evaluation. In addition, they do not consider time-variant data as found in time-interval data streams.

Existing continuous skyline algorithms [3,4] use a *sliding window model* that fixes the expiration times for *all* data objects to be the end of the window period. A new object  $o_{new}$  always expires after all objects in the current window and therefore can forever eliminate all existing objects that it dominates. In contrast, for time-interval streams an older object with a longer life span than  $o_{new}$  could revert back to becoming a skyline point in the future. This complicates the object dominance reasoning and worst yet may cause in some cases all objects to have to be retained. Thus a time-interval model is more general and complex than the sliding window model. Methods proposed for the time-interval model could also be used for the sliding window model, but not vice versa.

While [5] also handles skylines over time-interval streams, it suffers from the drawback of re-evaluating the skyline for each update and for each pull-based query. [3-5] work exclusively at the object granularity level - which is much more computation intensive for high volume data streams than our proposed higher-level abstraction.

**Our Proposed Approach.** We propose the *TI-Sky* a framework to evaluate continuous skylines over time-interval streams. Fully maintaining the skyline result space for each time instance makes answering user queries cheap. However, for high volume data streams the CPU resources needed to maintain this moving skyline result space are too prohibitive to be practical. Conversely, if we do not maintain this space, the results would need to be computed from scratch for each user pull request thereby negatively affecting performance. Therefore, our **optimization mantra** is “*to strike the perfect balance between the cost of maintaining the moving skyline result space and the cost of computing the final skyline result.*” This is achieved by incrementally maintaining the partially precomputed skyline result space – however doing so efficiently by working at a higher level of abstraction. We introduce the notions of *Macro and Micro-TDominance* to model the time-based dominance relationships at the individual object- as well as at the abstract-granularity. We then design algorithms for insertion, deletion, purging and result retrieval that effectively exploit both layers of granularity. Our contributions include:

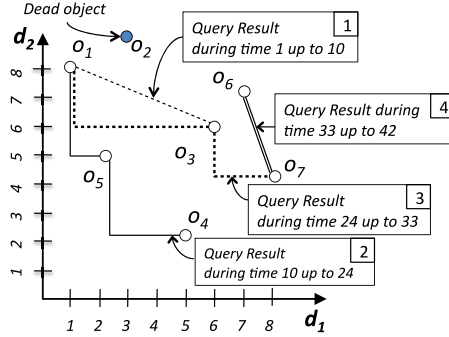
- We propose *TI-Sky* – an efficient execution framework to process multi-criteria decision support queries over time-interval data streams.
- We take the idea of time-based dominance to a new level by designing a two-layered model of time-based dominance, namely, *macro-* and *micro-* time-dominance.
- We propose efficient algorithms to handle real-time insertion, deletion, purging and result retrieval that effectively exploit both layers of our dominance model.
- Our experimental study demonstrates *TI-Sky*’s superiority over existing techniques.

## 2 Skylines over Time-Interval Streams

Let  $o_i.t_{arr}$  and  $o_i.t_{exp}$  denote the arrival and expiration time of an object  $o_i$ . A *strict partial order*  $o_i \succ_{\mathbb{P}} o_j$  exists if for all attribute dimensions  $k$ ,  $o_i[k] \leq o_j[k]$  and there exist at least one dimension  $k$  such that  $o_i[k] < o_j[k]$ . Below we extend the concept of value-based dominance proposed in [4] to incorporate the notion of time-intervals.

	Arrival Time		Expiration Time	
	$d_1$	$d_2$	$t_{arr}$	$t_{exp}$
$o_1$	1	8	1	33
$o_2$	3	9	1	29
$o_3$	6	6	1	33
$o_4$	5	2	10	24
$o_5$	2	5	10	24
$o_6$	7	7	16	42
$o_7$	8	4	18	48

(a) Sample Input Stream



(b) Continuous Skylines over Time-Interval Stream

**Fig. 1.** Motivating Example

**Definition 1. Time-Interval Dominance.** Given a preference  $\mathbb{P}$ , the time-interval stream  $S$ , and  $o_i, o_j \in S$  s.t.  $o_i \succ_{\mathbb{P}} o_j$  and  $o_j.t_{exp} > o_i.t_{exp}$ . Then  $o_i$  **TDominates**  $o_j$  ( $o_i \xrightarrow{T} o_j$ ) iff  $\nexists o_k \in S$  s.t.  $(o_k \succeq_{\mathbb{P}} o_j) \wedge (o_i.t_{exp} < o_k.t_{exp} < o_j.t_{exp})$ .

In Figure 1 we observe that object  $o_4 \xrightarrow{T} o_7$  and  $o_5 \xrightarrow{T} o_3$ . This means objects  $o_3$  and  $o_7$  become part of the skyline when the objects  $o_5$  and  $o_4$  expire respectively.

**Definition 2 (Skyline Over Time-Interval Stream).** Given a time-interval stream  $S$  and a preference  $\mathbb{P}$ , the skyline over  $S$  at time  $t_i$  (denoted as  $S_{\mathbb{P}}(S, t_i)$ ) is the set of all non TDominated objects  $o_k \in S$  with  $o_k.t_{arr} \leq t_i$  and  $o_k.t_{exp} > t_i$ .

In Figure 1 at  $t = 9$ ,  $o_1$  and  $o_3$  are the skyline objects. At  $t = 10$ , the new object  $o_5$  (2, 5) that dominates the older object  $o_3$  (6, 6) arrives. However, the newer  $o_5$  is valid for a shorter time frame ( $o_5.t_{exp} = 24$ ) than the older object  $o_3$  which is valid even after  $o_5$  expires ( $o_3.t_{exp} = 33$ ). This is reflected in Figure 1b where the older object  $o_3$  contributes to the result between the time frame 24 to 33 when  $o_5$  expires.

### 3 TI-Sky: Our Proposed Approach

In this work, we partition the input space composed of incoming streaming objects (see Definition 3). Similar in spirit to the principle of TDominance between two objects, henceforth referred to as Micro-TDominance. We propose to elevate the notion of time-based dominance to the granularity of abstractions called Macro-TDominance.

**Definition 3.** A **partition cell**<sup>1</sup> abstracts the set of objects that map into a the  $d$ -dimensional bounding box ( $P_k$ .RBoundary) defined by its lower and upper bounds.  $P_k$ .SkyHeap( $t$ ) is a list of objects not dominated by other objects in  $P_k$  at time instance  $t$ .  $P_k$ .RestHeap( $t$ ) maintains the objects in  $P_k$  currently dominated by objects

<sup>1</sup> Henceforth, a **partition cell**  $P_i$  is termed for short as a **partition**  $P_i$ .

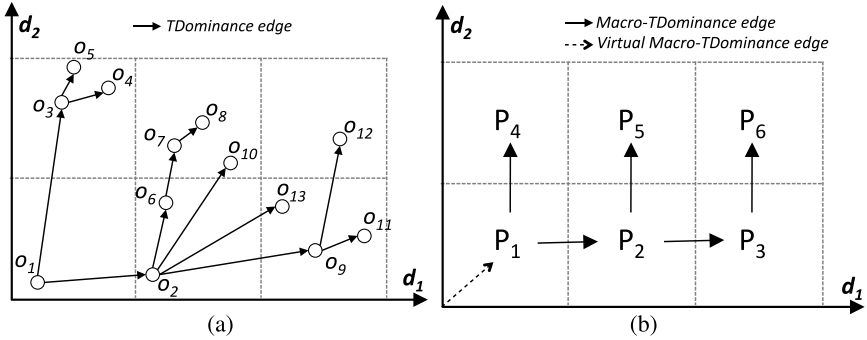


Fig. 2. Level of TDomination: (a) Object-Level (Micro) (b) Partition-Level (Macro)

in  $P_k$ . *SkyHeap*.  $P_k$ . **VBoundary** denotes a vector of the min and max attribute values across all objects in  $P_k$ .  $P_k$ . **TBoundary** defines the minimum and maximum expiration time among objects in  $P_k$ .

The concept of **Macro-TDomination** enables us to capture the time-based dominance relationship between partitions. The intuition is that non-empty root partitions of the Macro-TDomination tree are guaranteed to have skyline results and therefore should be investigated during the skyline data maintenance and result determination steps.

**Definition 4.** Partition  $P_f$  **Macro-TDominates** partition  $P_g$ , denoted as  $P_f \xrightarrow{T} P_g$ , if and only if  $\exists o_i \in P_f \wedge \exists o_j \in P_g$  such that  $o_i \xrightarrow{T} o_j$ .

*Example 1.* In Figure 2 the Macro-TDomination relationships  $P_1 \xrightarrow{T} P_4$  and  $P_1 \xrightarrow{T} P_2$  exist since  $o_1 \in P_1$  such that  $o_1 \xrightarrow{T} o_3$ , and  $o_1 \xrightarrow{T} o_6$ , where  $o_3 \in P_4$  and  $o_6 \in P_2$  respectively. Also,  $P_{virtual} \xrightarrow{T} P_1$  is the *virtual Macro-TDomination* relationship.

In the Macro-TDomination tree, a partition  $P_i$  that is not Macro-TDominated by any other partition  $P_j$  is termed as a root partition and is guaranteed to at-least contain one skyline result. The arrival of new object  $o_{new}$  to one such root partition and the expiration of an older object from any root partition can affect the result space and therefore need to be processed immediately. In contrast operations in non-root partitions can be performed more lazily without affecting correctness.

During skyline result generation the Macro-TDomination tree aids us to identify partitions that have a higher likelihood of delivering results - without having to examine object-level dominance. To find the precise result set upon request, we begin by looking at objects within such root partitions and then iterate over partitions whose boundary values are not being dominated by the current skyline points. This approach of traversing the space avoids unnecessary object-level comparisons.

**Definition 5.** Given a time-interval stream  $S$  and partition  $P_k$ , the **Micro-TDomination** relationship for the objects in  $P_k$  is simply the TDomination relationship among the objects in  $P_k$  rather than between all objects in  $S$ .

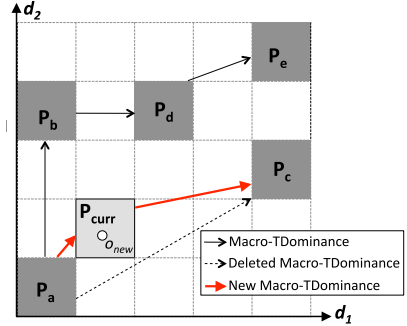
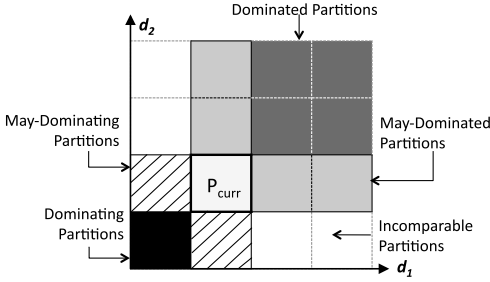


Fig. 3. Value-Based Categorization w.r.t.  $P_{curr}$  Fig. 4. Inserting  $o_{new}$  in  $P_{curr}$  and Traversing

## 4 Skyline Data Maintenance

We now describe the process of inserting a newly arriving object  $o_{new}$  into the abstract output skyline space. This is accomplished at two levels of granularity, namely both at the level of partitions as well as at the level of individual objects. For each new object  $o_{new}$ , we first map it to its corresponding partition. In Figure 4,  $o_{new}$  is mapped to the partition  $P_{curr}$ . Next, we perform local partition-level updates which encompass: (a) performing pairwise object-level comparisons against objects mapped to  $P_{curr}$  to determine whether  $o_{new}$  belongs to either *SkyHeap* or *RestHeap* of  $P_{curr}$ , and (b) updating the value- and temporal- bounds of  $P_{curr}$  (See Definition 3).

After local partition level operations, we determine the effects of  $o_{new}$  on the current and future skyline results. To avoid traversing the entire space as done in existing techniques [3-5], we propose first traverse the space at the higher abstraction of partitions and dig into the pairwise object comparisons as and when need. This is achieved by performing a breadth first search of Macro-TDominance tree starting with the *root partitions*. The goal of this exercise is to identify new Macro-TDominance relationships of the form  $P_i \xrightarrow{T} P_{curr}$  and  $P_{curr} \xrightarrow{T} P_i$  where  $P_i$  is a non-empty partition in the abstract output skyline space. To aid abstract-level decision making, we classify the visited partition  $P_i$  into four categories with respect to the partition  $P_{curr}$  as follows:

1. *Dominating Partition*: Partition  $P_i$  value-dominates  $P_{curr}$ .
2. *Dominated Partition*: Partition  $P_i$  is value-dominated by  $P_{curr}$ .
3. *May-Dominating Partition*: Objects mapped to the partition  $P_i$  can potentially value-dominate the new object  $o_{new} \in P_{curr}$ . But this cannot be determined without performing pairwise object-level comparisons.
4. *May-Dominated Partitions*: Objects mapped to the partition  $P_i$  can potentially be value-dominated by the new object  $o_{new} \in P_{curr}$ . Similar to category 3 this can only be determined by performing pairwise comparisons.

Based on the above classification (as depicted in Figure 3) only partitions that are either in  $P_{curr}$ 's *Dominating Partition* or *May-Dominating* lists can Macro-TDominate  $P_{curr}$ .

Let us now consider the scenario where the currently visited partition  $P_i$  is in the May-Dominating list of  $P_{curr}$ . If the relationship  $P_i \xrightarrow{T} P_{curr}$  already exists we merely update the time-stamp information of  $P_i \xrightarrow{T} P_{curr}$ . In contrast if the relationship  $P_i \xrightarrow{T} P_{curr}$  does not exist - we first compare the VBoundary values of  $P_i$  and  $o_{new}$ . If upper bound of  $P_i$ 's VBoundary dominates  $o_{new}$  then  $P_i \xrightarrow{T} P_{curr}$ , else there may still exist an object in  $P_i$  that dominated  $o_{new}$ . Next, we perform object-level comparisons to identify the object, say  $o_k$ , that dominated  $o_{new}$  that is last to expire. We then proceed to traverse the Macro-TDominance tree to check if there exist other partitions that can potentially Macro-TDominate  $P_{curr}$ .

*Example 2.* In Figure 4 initially,  $P_a \xrightarrow{T} P_b$ ,  $P_a \xrightarrow{T} P_c$ ,  $P_b \xrightarrow{T} P_d$  and  $P_d \xrightarrow{T} P_e$  are the Macro-TDominance relationships before the arrival of  $o_{new}$ . When  $P_{curr}$  is populated with  $o_{new}$ , we traverse the Marco-TDominance tree starting with  $P_a$ . Since  $P_a$  value dominates  $P_{curr}$  and there does not exist another partition in  $P_{curr}$ 's "Dominating" or "May-Dominating" list we add the Macro-TDominance  $P_a \xrightarrow{T} P_{curr}$ .

Next, we consider the relationship between  $P_{curr}$  and all partitions belonging to the *May-Dominated* and *Dominated* lists of  $P_{curr}$ . That is, we investigate if any  $P_{curr} \xrightarrow{T} P_i$  relationships exist. Decisions can be made at the abstract level for *Dominated* lists and object-level comparisons are needed for partitions in the *May-Dominating* list.

*Example 3.* In Figure 4,  $o_{new}.t_{exp} > P_a.TBoundary$ . Thus we can delete the relationship  $P_a \xrightarrow{T} P_c$  and insert the new Macro-TDominance relationship  $P_{curr} \xrightarrow{T} P_c$

When building these Macro-TDominance relationships we want to avoid traversing the entire Macro-TDominance tree by exploiting the temporal properties of Macro-TDominance as in Theorem 1.

**Theorem 1.** *For the new object  $o_{new} \in P_{curr}$  and a Macro-TDominance  $P_i \xrightarrow{T} P_j$ . If  $\exists o_k \in P_i \not\# o_m, o_n \in P_j$  s.t.  $(o_k \xrightarrow{T} o_m) \wedge (o_{new} \xrightarrow{T} o_n) \wedge (o_k.t_{exp} < o_{new}.t_{exp})$ , then we do not need to investigate the Macro-TDominance sub-tree with  $P_j$  as its root.*

*Example 4.* In Figure 4, the Macro-TDominance  $P_b \xrightarrow{T} P_d$  holds even after the arrival of the new object  $o_{new} \in P_{curr}$ . In such cases by Theorem 1 checking  $o_{new}$  against objects mapped to partitions with  $P_d$  as its root is unnecessary.

Detailed descriptions of the algorithms used to maintain the skyline result space due to object expiration and reducing memory usage by effective purging techniques are presented in the our technical report [6].

## 5 Skyline Result Determination

The *skyline result determination* phase piggybacks on the Macro-TDominance relationships gathered in the previous phase to determine the skyline results at any given time  $t_i$ . The root nodes (partitions) of the Macro-TDominance tree are guaranteed to have at least one query result. This translates into querying the Macro-TDominance tree [6].

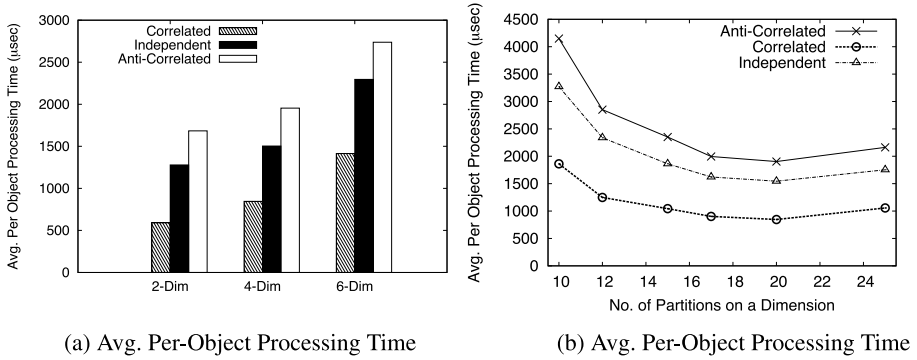


Fig. 5. Performance Evaluation of TI-Sky: (a) Effects of Dimensionality

## 6 Performance Study

**State-of-the-Art Techniques Compared.** We compare *TI-Sky* against *LookOut* [5] and the extension of the technique proposed in [4] here called *Stabbing-the-Sky+*.

**Experimental Platform.** All algorithms were implemented in C++. All measurements were obtained on a Windows machine with Intel 2.2GHz Dual Core and 2GB memory.

**Data Sets.** Generated by the *de-facto* standard in literature for stress testing skyline algorithms [1] contain three extreme attribute correlations: *independent*, *correlated*, or *anti-correlated*. We vary skyline dimensions  $d$  [2–6]. Since data sets generated by [1] have no associated time-stamps we add these time stamps. The validity period of each object is normally distributed with 100K as the mean (objects expire after 100K new objects have been processed) and a standard deviation of  $\alpha$ K objects (denoted as 100K+/- $\alpha$ K). The cardinality of the data stream is 1 million data objects.

**Experimental Analysis of TI-Sky.** We study the robustness of *TI-Sky* by varying: (1) the number of partitions on each dimension, and (2) number of dimensions.

- *Number of Skyline Dimensions ( $d$ ).* Figure 5a varies  $d=2$  to 6 for all three distributions. As  $d$  increases more partitions are created to maintain and query the data.
- *Number of Partitions ( $k$ ).* The number of partitions on each dimension affects how many objects will map to each partition. As  $k$  decreases more objects map to a partition cell increasing the partition-level memory and CPU usages. An increase in  $k$  will reduce local partition-level processing but increase abstract-level processing as the number of Macro-TDominance relations increases. Figure 5b shows avg. per object processing time of *TI-Sky* when the number of partitions is varied from 10 to 25 across three distributions. In Figure 5b the cross-over point for all distributions is  $k=20$ .

**Comparison of Alternative Techniques.** In Figure 6 for correlated data *TI-Sky* is 1 and  $\approx 2$  fold faster than *Stabbing-the-Sky+* and *LookOut*. For independent data *TI-Sky* is 29% and 60% faster than *Stabbing-the-Sky+* and *LookOut* respectively. Lastly, for anti-correlated data *TI-Sky* has a performance benefit of being on an average of 38% and

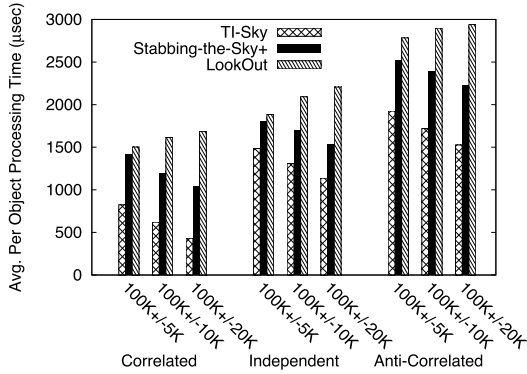


Fig. 6. Performance Comparison Against State-of-the-Art Algorithms ( $d = 4$ )

69% faster than *Stabbing-the-Sky+* [4] and *LookOut* [5] respectively. As the standard deviation increases the average per-object processing cost for both *TI-Sky* and *Stabbing-the-Sky+* decreases. This is because each object has a higher likelihood of remaining valid longer and can be effective in purging more dominated objects. However *LookOut* maintains all objects without purging and therefore requiring more time and space. These performances are consistent with those for other dimensions  $d[2-6]$ .

## 7 Conclusion

Existing techniques that support skyline queries over streams either focus on the simpler model of continuous sliding window, or require to re-compute the skyline by performing multiple index scans for insertion and expiration of objects to handle time-interval streams. We present *TI-Sky* an efficient framework to evaluate skylines over time-interval streams. We propose a novel technique called *Macro-TDominance* to model the output dependencies between abstractions of the skyline result space. By analyzing the *Macro-TDominance* relationships, *TI-Sky* efficiently maintains the output space as well as deliver real-time results for user requests. Our experimental study confirms that *TI-Sky* has a superior performance compared to existing techniques.

## Acknowledgment

This work was supported by National Science Foundation (IIS-0633930, ISS-0917017 and CRI-0551584) and the Korea Research Foundation (KRF-2008-357-D00214).

## References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
2. Raghavan, V., Rundensteiner, E.A.: Progressive result generation for multi-criteria decision support queries. In: ICDE, 733–744 (2010)

3. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. *TKDE* 18(2), 377–391 (2006)
4. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: *ICDE*, pp. 502–513 (2005)
5. Morse, M., Patel, J., Grosky, W.: Efficient continuous skyline computation. *Inf. Sci.*, 3411–3437 (2007)
6. Park, N., Raghavan, V., Rundensteiner, E.: Supporting multi-criteria decision support queries over time-interval data streams. Technical Report WPI-CS-TR-10-12, Dept. of Computer Science, Worcester Polytechnic Institute (2010)



# Faster Algorithms for Searching Relevant Matches in XML Databases

Rung-Ren Lin<sup>1</sup>, Ya-Hui Chang<sup>2</sup>, and Kun-Mao Chao<sup>1</sup>

<sup>1</sup> Department of Computer Science and Information Engineering  
National Taiwan University, Taipei, Taiwan  
{r91054, kmchao}@csie.ntu.edu.tw

<sup>2</sup> Department of Computer Science and Engineering  
National Taiwan Ocean University, Keelung, Taiwan  
yahui@ntou.edu.tw

**Abstract.** Keyword search is a friendly mechanism for the end user to identify interesting nodes in XML databases, and the SLCA (smallest lowest common ancestor)-based keyword search is a popular concept for locating the desirable subtrees corresponding to the given query keywords. However, it does not evaluate the importance of each node under those subtrees. Liu and Chen proposed a new concept *contributor* to output the *relevant matches* instead of all the keyword nodes. In this paper, we propose two methods, *MinMap* and *SingleProbe*, that improve the efficiency of searching the relevant matches by avoiding unnecessary index accesses. We analytically and empirically demonstrate the efficiency of our approaches. According to our experiments, both approaches work better than the existing one. Moreover, *SingleProbe* is generally better than *MinMap* if the minimum frequency and the maximum frequency of the query keywords are close.

**Keywords:** keyword search, XML, smallest LCA, contributor.

## 1 Introduction

Keyword search provides a convenient interface to quickly obtain desired information from XML documents. In general, an XML document could be viewed as a rooted tree. The conventional keyword search approach returns all the LCA nodes in the XML tree if they contain every keyword in their own subtrees. However, this often gets too many nodes since some of them have low relationship with the query keywords. Xu and Papakonstantinou [6] thus proposed the concept of SLCA. A node is said to be an SLCA if (i) it contains all the keywords under its subtree, and (ii) it has no descendant node that also contains all the keywords.

The advantage of the SLCA-based keyword search algorithm is that the obtained SLCA nodes are semantically closer to the query keywords, but all the nodes under the subtrees rooted at the SLCA nodes are considered equally important. Therefore, Liu and Chen [5] defined a new concept *contributor* to determine the important keyword nodes. They also gave an efficient algorithm *MaxMatch* to

locate all the contributors. Furthermore, MaxMatch is the first algorithm that satisfies the *monotonicity* and *consistency* properties, which capture a reasonable connection between the new query result and the original query result after an update to the query or to the data. Briefly, the monotonicity property indicates the change to the number of SLCA nodes, and the consistency property describes the change to the content of query result. These properties are advisable and worthwhile, yet none of the existing LCA-based approaches [1][2][3][4][6] satisfy both properties.

In this paper, we propose two algorithms MinMap and SingleProbe to improve the efficiency of MaxMatch. The main contributions are as follows:

- We first introduce the MinMap algorithm. Similar to the MaxMatch algorithm, it first finds all the SLCAs, and then constructs the essential information of the subtrees rooted at the SLCAs. It substantially improves MaxMatch by avoiding unnecessary index accesses.
- The second algorithm, SingleProbe, directly constructs the full tree without first finding the SLCAs, and then locates the SLCAs and the relevant matches by traversing the full tree. Specifically, when the total size of the subtrees rooted at the SLCAs is close to the size of the full tree, SingleProbe may save the time for locating the SLCA nodes.
- We empirically evaluate algorithms MinMap and SingleProbe. The experiments show that our new approaches outperform the previous work. Moreover, SingleProbe works better than MinMap when the minimum frequency and the maximum frequency of the query keywords are close.

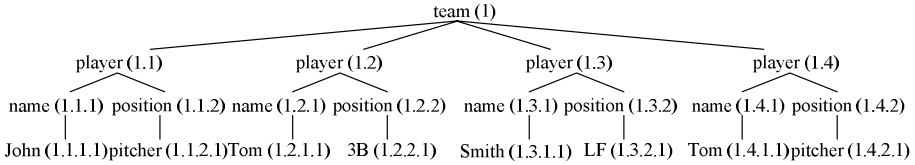
The rest of this paper is organized as follow. In Section 2, we introduce the MaxMatch algorithm. Section 3 and Section 4 present our approaches that improve the execution time of searching the relevant matches. We discuss some experimental studies in Section 5. Conclusions are discussed in Section 6.

## 2 Preliminaries

We use the labeled ordered tree model to represent XML trees. Every node in the tree has a tag name or the content and a unique Dewey number. In the following, we deliver the definitions given in MaxMatch [5].

A node is a *match* if its tag name or the content corresponds to a given query keyword. The *descendant matches* of a node  $n$ , denoted as  $dMatch(n)$  or  $n.dMatch$ , are a set of query keywords, each of which has at least one match in the subtree rooted at  $n$ .  $dMatch(n)$  could also be seen as a bit array of size  $w$  for simplicity, where  $w$  is the number of keywords. Besides, a node  $n$  is a *contributor* if (i)  $n$  is the descendant of a given SLCA or  $n$  itself is one of the SLCAs, and (ii)  $n$  does not have a sibling  $n_2$  such that  $dMatch(n_2) \supset dMatch(n)$ .

The *dMatchSet* value of a node is a bit array of size  $2^w$  to record the *dMatch* values of its children. All the bits are initialized as 0 at the beginning. The  $j^{th}$  bit,  $j \leq 2^w - 1$ , is set to 1 if it has at least one child  $n_c$  such that  $dMatch(n_c) = j$ .



**Fig. 1.** A sample XML tree

At last, a node  $n$  is considered *relevant* if (i)  $n$  has an ancestor-or-self  $t$  such that  $t$  is one of the SLCAs, and (ii) each node on the path from  $n$  to  $t$  is a contributor.

For example, consider query  $Q_1$  ( $team, pitcher, Tom$ ) over Figure 1. Suppose the keywords (from left to right) correspond to the first (right-most), the second, and the third bits of  $dMatch$ . We have  $dMatch(1.1) = 010_{binary}$ ,  $dMatch(1.2) = 100_{binary}$ , and  $dMatch(1.4) = 110_{binary}$ . Nodes 1.1 and 1.2 are not the contributors since both  $dMatch(1.1)$  and  $dMatch(1.2)$  are the proper subsets of  $dMatch(1.4)$ . We directly skip node 1.3 because it has no descendant match. In addition, the second, the fourth, and the sixth bits of  $dMatchSet(1)$  are set to 1.

The MaxMatch algorithm could be decomposed into four steps as follows:

1. Retrieve the matches of each query keyword.
2. Compute the SLCAs by the algorithm given in [6].
3. Group all keyword matches according to their SLCA ancestors.
4. For each SLCA, construct the correct value of  $dMatch$  and  $dMatchSet$  for each node from the matches up to the SLCA. Then determine the contributors in the preorder traversal sequence.

### 3 MinMap

We explain the MinMap algorithm in this section. Recall that the tree constructed in Step 4 of MaxMatch is composed of the matches up to the SLCA. Note that every match has its corresponding query keyword, and the nodes without matching any query keyword are called *non-keyword nodes*. MaxMatch retrieves the tag names of all the non-keyword nodes from the *Dewey index* during the construction of the tree. However, some of the non-keyword nodes are eventually pruned. In this proposed approach, we do not retrieve the tag names for those pruned non-keyword nodes, which therefore saves a lot of disk I/O time. In the following, we first deliver the definitions of our approach and then describe the MinMap algorithm and its time complexity.

The *match tree* of a node  $t$ , denoted as  $mTree(t)$ , consists of the nodes along the path from each match up to  $t$ . Besides, a node  $n$  is a *hit node* if (i)  $n$  is contained in the match tree rooted at a given SLCA, (ii)  $n$  is a non-keyword

<sup>1</sup> Interested readers please refer to the original paper for more details.

<sup>2</sup> The Dewey index is based on the B-tree structure, where the key is the Dewey number of a node, and the associated value is the tag name.

<pre> <b>MinMap</b>(keyword[w]) 1: kwMatch <math>\leftarrow</math> FindMatch(keyword[w]) 2: SLCA <math>\leftarrow</math> FindSLCA(kwMatch) 3: group <math>\leftarrow</math> GroupMatches(kwMatch, SLCA) 4: <b>for</b> each group[j] = (t, M) <b>do</b> 5:   ConstructTree(t, M) 6:   PreorderTrav(t)  <b>ConstructTree</b>(t, M) 1: i <math>\leftarrow</math> M.size - 1 2: <b>while</b> i <math>\geq</math> 0 <b>do</b> /* suppose M[i] corresponds to the j<sup>th</sup> keyword */ 3:   <b>for</b> each node n on the path from M[i] to t <b>do</b> 4:     nc <math>\leftarrow</math> n.child on this path 5:     <b>if</b> nc <math>\neq</math> null <b>then</b> 6:       n.dMatchSet[nc.dMatch] <math>\leftarrow</math> true 7:     <b>else</b> 8:       n.keywordFlag <math>\leftarrow</math> j 9:       <b>if</b> the j<sup>th</sup> bit of n.dMatch is 1 <b>then</b> 10:        break the for loop 11:      <b>else</b> 12:        set the j<sup>th</sup> bit of n.dMatch to 1 13:    i <math>\leftarrow</math> i - 1 </pre>	<pre> <b>PreorderTrav</b>(n) 1: np <math>\leftarrow</math> n.parent 2: <b>if</b> np = null <b>or</b> IsContributor(n) = true <b>then</b> 3:   <b>if</b> n.keywordFlag = 0 <b>then</b> 4:     retrieve n's tag name 5:   output n /*including the tag name*/ 6:   <b>for</b> each child nc of n <b>do</b> 7:     PreorderTrav(nc)  <b>IsContributor</b>(n) 1: np <math>\leftarrow</math> n.parent 2: i = n.dMatch 3: <b>for</b> j <math>\leftarrow</math> i+1 to 2<sup>w</sup>-1 <b>do</b> 4:   <b>if</b> np.dMatchSet[j] = true <b>and</b>       AND(i, j) = i <b>then</b> 5:     return false 6: return true </pre>
--	--

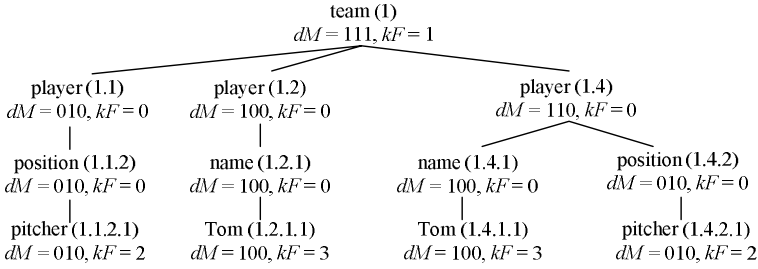
**Fig. 2.** The *MinMap* algorithm

node, and (iii)  $n$  is eventually a relevant match. On the contrary, if  $n$  is eventually being pruned, it is called a *miss node*. Furthermore, *miss rate* is defined as:  $\Sigma$  miss nodes / ( $\Sigma$  hit nodes +  $\Sigma$  miss nodes) for all match trees of the SLCA's. We give an example in the following.

Consider query  $Q_1$  again. Recall that nodes 1.1 and node 1.2 are pruned by node 1.4. According to the definitions, the nodes in  $mTree(1.1)$  and  $mTree(1.2)$  are miss nodes. Specifically, the hit node list is [1.4, 1.4.1, 1.4.2], and the miss node list is [1.1, 1.1.2, 1.2, 1.2.1]. Therefore, the miss rate is  $4/(3+4) = 57\%$ .

The pseudocode of *MinMap* is shown in Figure 2. This algorithm could be decomposed into four steps, too. The first three steps (lines 1 to 3 of procedure *MinMap*) are the same as *MaxMatch*, but we improve the fourth step of *MaxMatch* by avoiding unnecessary index accesses. Specifically, the *ConstructTree* procedure and the *PreorderTrav* procedure are introduced in the *MinMap* algorithm. The former constructs the match tree for each SLCA, and the latter prunes the irrelevant matches in the preorder traversal sequence. In the following, we explicitly illustrate these two procedures.

For each iteration of the while loop (lines 2 to 13 of *ConstructTree*), we set the values of *dMatch*, *dMatchSet*, and *keywordFlag* for each node along the path from  $M[i]$  up to  $t$ . The additional variable *keywordFlag*, which is initialized as 0, is designed to determine whether  $n$  is a match or not. In line 6, we record the  $dMatch(n_c)$  value in  $n.dMatchSet$  if  $n_c$  is not null, where  $n_c$  is the child of



**Fig. 3.** The match tree of query  $Q_1$

$n$  on this path. Otherwise, we set the  $n.keywordFlag$  as  $j$  if  $M[i] \in kwMatch[j]$ . Actually, the setting of *keywordFlag* (line 8) is active only when we first enter the for loop, so node  $n$  is a match if  $n.keywordFlag = j$ ,  $1 \leq j \leq w$ . After constructing the match tree, we can directly output  $keyword[j]$  as the tag name of node  $n$  if  $n$  is eventually a relevant match. However, if  $n.keywordFlag$  is zero after constructing the match tree, it means that  $n$  is a non-keyword node. We have to retrieve  $n$ 's tag name by the Dewey index. In addition, in line 9, we check if the  $j^{th}$  bit of  $n.dMatch$  is 1. If so, we can break the for loop because the  $j^{th}$  bits of all the  $m.dMatch$  values are also 1, where  $m$  are the nodes along the path from  $n$  up to the SLCA. We set the  $j^{th}$  bit of  $n.dMatch$  to be 1 in line 12 if line 10 is not active.

Procedure *PreorderTrav* takes a node  $n$  as the input, and outputs the Dewey number and the tag name of  $n$  if  $n$  is a contributor<sup>3</sup>. In line 4, we retrieve the tag name of node  $n$  only when it is a hit node. In addition,  $n$ 's children are iteratively checked in lines 6 to 7.

The complete state of the match tree of query  $Q_1$  is shown in Figure 3. Variable  $kF$  represents the *keywordFlag* value and variable  $dM$  represents the *dMatch* value in the binary mode. Variables *keywordFlag* of the nodes in the non-keyword node list [1.1, 1.1.2, 1.2, 1.2.1, 1.4, 1.4.1, 1.4.2] are all zero. Since only the last three nodes are relevant matches, we look up the Dewey index exactly three times. Note that we suppose the tag name is the essential information of the query result. However, the Dewey index would be no longer needed if the query result contains the Dewey number only. This will make our approaches perform even better since every non-keyword node skips index accesses.

We then give the time complexity of MinMap. Let  $|M_1|$ ,  $|M_2|$ , ..., and  $|M_w|$  denote the frequencies of the query keywords, and  $d$  is the maximum depth of the XML tree. The main memory complexity of MinMap is  $O(d|M| \cdot 2^w)$ , where  $|M| = \sum_{i=1}^w |M_i|$ . Besides, since B-tree implementations usually keep upper-level nodes in memory, we assume the number of disk accesses for retrieving the tag name of a given node is  $O(1)$ . Hence, the total number of disk accesses is bounded in  $O(|M| + C)$ , where  $C$  is the number of hit nodes.

<sup>3</sup> Note that procedure *IsContributor* is a part of MaxMatch which runs in  $O(2^w)$  for each checking.

<p><b>SingleProbe</b>(keyword[w])</p> <pre> 1: <math>kwMatch \leftarrow FindMatch(keyword[w])</math> 2: <math>M \leftarrow MergeMatches(kwMatch)</math> 3: <math>ConstructTree(root, M)</math> 4: <math>LocateSLCA(root)</math> </pre>	<p><b>LocateSLCA</b>(n)</p> <pre> 1: <math>result \leftarrow 0</math> 2: <b>if</b> <math>n.dMatch \neq 2^w - 1</math> <b>then</b> 3:   <b>return</b> 0 4: <b>for</b> each child <math>nc</math> of <math>n</math> <b>do</b> 5:   <math>result \leftarrow result + LocateSLCA(nc)</math> 6: <b>if</b> <math>result = 0</math> <b>then</b> 7:   <math>PreorderTrav(n) /* n</math> is an SLCA */ 8: <b>return</b> 1 </pre>
--	---

Fig. 4. The *SingleProbe* algorithm

## 4 SingleProbe

In the MinMap algorithm, the keyword matches are first used to find the SLCAs (Step 2), and then form groups (Step 3). These groups will be again accessed to build individual trees. In this section, we present the SingleProbe algorithm, where the keyword matches are directly merged to construct a single tree. We first illustrate SingleProbe and then give an example in the following.

The pseudocode of the SingleProbe algorithm is shown in Figure 4. In this algorithm, we first retrieve all the *kwMatch* arrays, and then merge the match arrays according to their Dewey numbers by procedure *MergeMatches* (line 2). The *MergeMatches* procedure is actually a part of the *GroupMatches* procedure. Instead of constructing the match tree for each SLCA, SingleProbe constructs only one match tree  $mTree(root)$  by procedure *ConstructTree* (line 3), where *root* is the root of the XML tree. We skip the *FindSLCA* procedure required in MaxMatch because we can easily locate the SLCAs after constructing  $mTree(root)$ .

Procedure *LocateSLCA* is a recursive function. It takes a given node  $n$  as the input, and returns 1 if  $n.dMatch$  equals to  $2^w - 1$ , which implies that the subtree rooted at  $n$  contains at least one match for each of the query keywords. We quit the procedure in line 3 if line 2 is active, because all the nodes under the subtree rooted at  $n$  are impossible to be the SLCA. If the *result* variable is zero in line 7, it indicates that none of  $n$ 's children is an SLCA. As a result,  $n$  is confirmed to be an SLCA. Next, we call the *PreorderTrav* procedure to prune the irrelevant matches of  $mTree(n)$ .

Take query  $Q_2$  (*pitcher, name*) as an example. Since  $dMatch(1) = 2^w - 1$ , it iteratively checks its children as shown in lines 4 to 5 of *LocateSLCA*. Eventually, nodes 1.1 and 1.4 return 1, and nodes 1.2 and 1.3 return 0. Hence, we know that node 1 (*team*) is not an SLCA. On the other hand, nodes 1.1 and 1.4 are SLCA nodes and output the relevant matches of  $mTree(1.1)$  and  $mTree(1.4)$  before they return 1 to their parent.

Note that in query  $Q_2$ , nodes 1, 1.2, 1.2.1, 1.3, and 1.3.1 are redundantly processed since they do not belong to any  $mTree(t)$  where  $t$  is the SLCA node. Let  $SizeOf(n)$  denote the number of nodes of  $mTree(n)$ . Clearly,  $SizeOf(root)$  is always no less than  $\sum_{t \in SLCA} SizeOf(t)$ . It seems that it is better to use the SingleProbe algorithm to search the relevant matches when  $\sum_{t \in SLCA} SizeOf(t)$

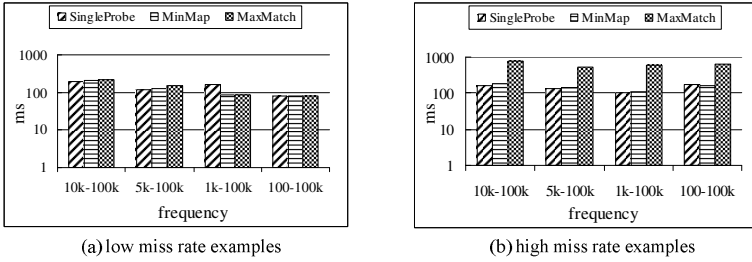


Fig. 5. Varying the ratio of the minimum frequency to the maximum frequency

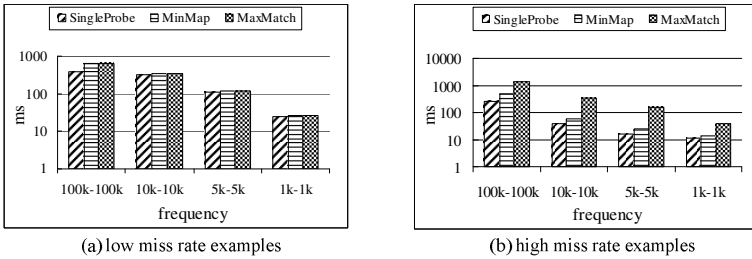


Fig. 6. Keeping the minimum frequency and the maximum frequency to be close

and  $SizeOf(root)$  are about the same size. In addition, the time complexity of SingleProbe is also  $O(d|M| \cdot 2^w)$ .

## 5 Experimental Studies

In this section, we discuss the implementation issues and compare the performance of our two algorithms with MaxMatch. The MinMap, SingleProbe, and MaxMatch algorithms are implemented in C++ with the environment of Windows XP and Visual Studio 6.0. Recall that the Dewey index is used to retrieve the tag name for a given Dewey number. Here we construct another B-tree index to efficiently retrieve all the matches of a given keyword (for Step 1). In this index, the data associated with each keyword  $k$  is a sorted list of Dewey numbers of all the nodes which contain keyword  $k$ . Thus we can retrieve the match array of the keyword in a single query instruction. The indices are created and accessed based on the Oracle Berkeley DB [7].

We applied `dblp.xml`<sup>4</sup> to evaluate the processing time. The experiments were performed on a 1.67GHz dual-core CPU with 1.5GB RAM, and the cache size of Oracle Berkeley DB was set as 1.0GB. We adopted the hot cache policy for the processing time testing. That is, for each query we executed three times and calculated the average processing time of the last two times. In Figure 5, we fixed the number of keywords as three, and varied the ratio of the minimum frequency to the maximum frequency. The experiments showed that the three algorithms

<sup>4</sup> <http://www.cs.washington.edu/research/xmldatasets/>

have similar performance in the low miss-rate cases, but our two approaches are substantially more efficient than MaxMatch in the high miss-rate cases. However, in the case of “1k-100k” of Figure 5(a), the SingleProbe algorithm is worse than the others, because  $SizeOf(root)$  is much larger than  $\Sigma_{t \in SLCA} SizeOf(t)$  in this case. In Figure 6, we kept the minimum frequency and the maximum frequency of the keywords to be close, and varied the frequencies simultaneously. It is obvious that the SingleProbe algorithm performs better than the others. The reason is that  $SizeOf(root)$  is close to  $\Sigma_{t \in SLCA} SizeOf(t)$  in most cases, and SingleProbe benefits from saving the time of searching the SLCA nodes.

The scalability was tested with different data sizes. We chose the `baseball.xml`<sup>5</sup> (1.01MB) as the data source and duplicated the content 100, 200, 300, and 400 times respectively. The experimental results show that the processing time is roughly in linear proportion with the data size for all the three algorithms. The figures are omitted due to space limitation.

## 6 Conclusions

In this paper, we propose two algorithms, MinMap and SingleProbe, that improve the efficiency of searching the relevant matches. Given a set of query keywords, the approaches return all the relevant matches including the Dewey number and the tag name. The MinMap algorithm improves the MaxMatch algorithm by eliminating unnecessary index accesses during the construction of the match tree. The SingleProbe algorithm combines Step 2, Step 3, and Step 4 of MaxMatch by constructing the full XML tree. The experimental results show that the two proposed approaches outperform MaxMatch when the miss rate is high. The SingleProbe algorithm works particularly well when the frequencies of keywords are close.

**Acknowledgements.** Rung-Ren Lin and Kun-Mao Chao were supported in part by NSC grants 97-2221-E-002-097-MY3 and 98-2221-E-002-081-MY3 from the National Science Council, Taiwan. Ya-Hui Chang was supported in part by the NSC grant 97-2221-E-019-028- from the National Science Council, Taiwan.

## References

1. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A Semantic Search Engine for XML. In: VLDB, pp. 45–56 (2003)
2. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked Keyword Search over XML Documents. In: SIGMOD, pp. 16–27 (2003)
3. Li, G., Feng, J., Wang, J., Zhou, L.: Effective Keyword Search for Valuable LCAs over XML Documents. In: CIKM, pp. 31–40 (2007)
4. Li, Y., Yu, C., Jagadish, H.V.: Schema-Free XQuery. In: VLDB, pp. 72–83 (2004)
5. Liu, Z., Chen, Y.: Reasoning and Identifying Relevant Matches for XML Keyword Search. In: VLDB, pp. 921–932 (2008)
6. Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: SIGMOD, pp. 527–538 (2005)
7. Oracle Berkeley DB.: <http://www.oracle.com/database/berkeley-db/index.html>

<sup>5</sup> <http://www.cafeconleche.org/books/biblegold/examples/>



# Efficient SLCA-Based Keyword Search on XML Databases: An Iterative-Skip Approach

Jiaqi Zhang, Zhenying He, Yue Tao, Xiansheng Wang, Hao Hu, and Wei Wang

School of Computer Science, Fudan University, Shanghai China  
{09210240048,zhenying,09210240035,072021106,  
06300720212,weiwang1}@fudan.edu.cn

**Abstract.** Keyword search is a popular way to discover information from XML data. To return meaningful results, SLCA (smallest lowest common ancestor) has been proposed to identify interesting data nodes. Since the SLCA is obtained from a series of intermediate LCAs, it can often incur many unnecessary LCA computations even when the size of final results is small. In this paper, we propose an Iterative-Skip approach to address this challenge. We first study the relation between SLCA candidates and propose a series of properties. Then based on these properties, we design a novel skip strategy to skip more SLCA computations. Experimental results show the effectiveness of our approach.

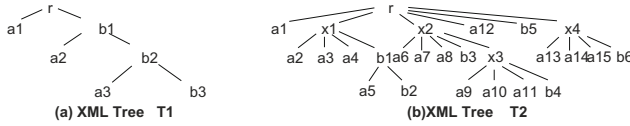
## 1 Introduction

Keyword search on XML data has been recently received an increasing interest in database community. Lots of research has been conducted on the semantics and techniques to keyword search over XML databases.

In this paper we concentrated on query processing for SLCA-based keyword search on XML data. To the best of our knowledge, the state-of-art algorithm for SLCA-based keyword search is *Incremental Multiway-SLCA (IMS)* [10]. *IMS* algorithm computes each potential SLCA by taking one data node from each keyword list  $S_i$  in a single step. It picks an anchor node among the keyword lists to drive the SLCA computation. By doing so, *IMS* skips a lot of redundant nodes. However, some meaningless SLCA candidates for final results are still involved in computations as the following example illustrates.

*Example 1.* Consider the XML tree  $T_1$  in Figure 1(a). The SLCA for the keywords  $\{a, b\}$  in  $T_1$  is  $\{b_2\}$ . According to *IMS* algorithm, anchor nodes are  $\{b_1, b_2, b_3\}$ . Here, anchor nodes  $b_1$  and  $b_2$  are meaningless for final results.

Another drawback of *IMS* algorithm is that it costs a lot in computing the corresponding match from anchor node. The process of obtaining the closest nodes costs  $O(d \log(|S_i|))$  time using Dewey labels ( $d$  is the depth of the XML document and  $|S_i|$  is the set's size), since it involves a binary search on keyword node set which corresponds to each input keywords.



**Fig. 1.** Example XML Trees  $T_1$  and  $T_2$

We make the following technical contributions in this paper:

1. We study the relationship between LCA nodes and matches, then present effective rules to iteratively skip more redundant computations.
2. We propose the MMPS algorithm to answer SLCA-based queries and implement IMS and MMPS with Dewey and Region encoding schemes respectively. The experimental results demonstrate that our approach outperforms former work.

The rest of this paper is organized as follows: Section 2 introduces the notations used in this paper. Section 3 propose the iterative skip rules and SLCA finding methods, and analyzes the complexity of our approach. In Section 4, we discuss the implement of our approach and experimental results. We introduce related work briefly in Section 5 and give a conclusion in Section 6.

## 2 Notations

An XML document is viewed as a directed, rooted, ordered, and node-labeled tree. We assign to each node of the tree a numerical id  $pre(v)$  which is compatible with pre-order numbering. Given two nodes  $u$  and  $v$ ,  $u \prec_p v$  donates  $pre(u) < pre(v)$ , and  $u \preceq_p v$  donates  $u \prec_p v$  or  $u = v$ . Similarly,  $u \prec_a v$  donates node  $u$  is an ancestor of  $v$ , and  $u \preceq_a v$  donates  $u \prec_a v$  or  $u = v$ .

Let  $K = (k_1, k_2 \dots k_n)$  donates a set of input keywords, where each  $k_i$  associated with  $S_i$  (set of XML data nodes, sorted in pre-order).  $V(v_1, v_2 \dots v_n)$  is called a match, where  $v_i \in S_i$  and  $vl$  is the leftmost node in match while  $vr$  is the rightmost node (with the property that  $vl \preceq v_i \preceq vr$ ). Function  $lca(V)$  (also represented as  $lca(v_1, v_2, \dots, v_n)$ ) returns the LCA of nodes  $v_1, v_2 \dots v_n$ . Function  $vl(V)$  (also  $vl(v_1, v_2 \dots v_n)$ ) returns the leftmost node of  $V$ .  $vr(V)$  (also  $vr(v_1, v_2 \dots v_n)$ ) returns the rightmost node of  $V$ .

For matches  $W(w_1, w_2 \dots w_n)$  and  $V(v_1, v_2 \dots v_n)$ . We say match  $W$  precedes match  $V$  (or  $V$  behind/succeeds  $W$ ) donated as  $W \prec_p V$ , if and only if  $W$  and  $V$  satisfy (1)  $w_i \prec_p v_i$ , ( $1 \leq i \leq k$ ), and (2)  $W \neq V$ . We say matches  $W$  and  $V$  share the same node, if there exists  $i$ , satisfied  $w_i = v_i$ .

## 3 Our Approach

In this section, we introduce properties and rules to skip matches and present key concepts  $SMatch$  and  $SNMatch$ . Then, we propose our  $MMPS(Multi-way Match Progress SLCA)$  algorithm in detail and give a complexity analyze on it.

### 3.1 Properties and Skipping Rules

**Lemma 1.** *Given a match  $V(v_1, v_2 \dots v_n)$ .  $lca(V) = lca(vl(V), vr(V))$*

**Lemma 2.** *Given matches  $V(v_1, v_2 \dots v_i \dots v_n)$  and  $V'(v_1, v_2 \dots v_{i'} \dots v_n)$ . If  $v_i \prec_a v_{i'}$ , then  $lca(V) \preceq_a lca(V')$*

**Lemma 3.** *Given matches  $V_1$  and  $V_2$ . If they share the same node  $v$  and  $lca(V_1)$  is an SLCA, then there must be  $lca(V_2) \preceq_a lca(V_1)$*

**Lemma 4.** *Given matches  $V$  and  $W$  ( $V \prec_p W$ ), if  $lca(V) \not\preceq_a lca(W)$ , then for any match  $X$  that  $W \preceq_p X$  there must be  $lca(V) \not\preceq_a lca(X)$*

**Lemma 5.** *Given matches  $V$  and  $W$  that  $V \prec_p W$ , if  $lca(W) \not\preceq_a lca(V)$ , then for any match  $X$  that  $X \prec_p V$  there must be  $lca(W) \not\preceq_a lca(X)$ .*

**Rule 1.** *For two nodes  $u$  and  $v$  in a keywords list  $S_i$ , if  $u \prec_a v$  node  $u$  can be skipped in SLCA computations.*

**Rule 2.** *Let  $vr$  be the rightmost node in match  $V$ , for other node  $v_i$ , if there exists  $v_{i'}$  ( $v_i \prec_p v_{i'} \prec_p vr$ ), then node  $v_i$  can be skipped.*

**Definition 1 (SMatch).** *Given a match  $V(v_1, v_2, \dots, v_n)$ , it is a SMatch if and only if rule 1 and rule 2 can not skip any node of it*

**Definition 2 (NMatch).** *Given matches  $V$  and  $W$  ( $V \prec_p W$ ). If there exists no match  $X$  satisfied  $V \prec_p X \prec_p W$ , then we say  $W$  is match  $V$ 's NMatch.*

**Definition 3 (SNMatch).** *Let  $vl'$  be the next node after  $vl$  in the keywords set  $S_i$ . We call the match  $V'(v_1, v_2, \dots, vl', \dots, v_n)$  the SNMatch of SMatch  $V$ .*

*Example 2.* Consider the keyword search  $\{a, b\}$  on tree  $T_2$  in Figure 1(b). The keyword sets are  $S_1 = \{a_1, a_2 \dots a_{15}\}$ ,  $S_2 = \{b_1, b_2, \dots b_6\}$ , respectively. The very beginning match is  $\{a_1, b_1\}$  and matches  $\{a_1, b_1\}$ ,  $\{a_2, b_1\}$ , and  $\{a_3, b_1\}$  can be safely skipped according to rule 2. And node  $b_1$  can be skipped as it has a descendant  $b_2$  in  $S_2$  according to rule 1. By repeatedly using rule 1 and rule 2, we finally get SMatch(smallest match)  $\{a_5, b_2\}$  and NMatches(next matches) of it are  $\{a_6, b_2\}$  and  $\{a_5, b_3\}$  in which  $\{a_6, b_2\}$  is SNMatch(special next matches).

Since each SLCA corresponds to one SMatch, other matches are not considered in SLCA computation. However because not all SMatches contribute to the final results, two methods are used to prune unnecessary SMatches.

First, the LCA of SNMatch of the SMatch must be a non-descendant node of the LCA of SMatch, or progress to the next SMatch. We call such LCA node candidate SLCA node. This method guarantees that candidate SLCA node does not have a descendant LCA node of matches behind it. NMatches (except SNMatch) have LCAs that are not descendant nodes of the LCA of SMatch, because NMatches (except SNMatch) share the same  $vl$  node with SMatch but

have a big pre-order numbering  $vr$  node. So if the LCA of  $SNMatch$  is not a descendant node of the LCA of  $SMatch$ , none of the LCAs behind are descendants of the LCA of this  $SMatch$  according to lemma 4 with the fact that matches behind  $SMatch$  either succeed  $NMatch$  or are  $NMatch$  themselves.

Second, the LCA of this  $SMatch$  should not be an ancestor of (nor the same node with) the previous SLCA node. This guarantees that the LCA of this  $SMatch$  will not have any descendant LCA previously according to lemma 5.

Using this two methods, all unnecessary  $SMatches$  are pruned and is sure to generate an SLCA node.

### 3.2 MMPS Algorithm

We use a cursor array  $C$  to maintain the current matches of which each  $C[i]$  ( $1 \leq i \leq k$ ,  $k$  is the number of input keywords) points to a node in  $S[i]$ . The  $SMatch$  is computed by repeatedly using rule 1 and 2 to skip nodes until no match can be skipped. The  $SNMatch$  is computed from an  $SMatch$  by progressing the  $vl$  node to the next node in the keyword set according to its definition.

---

#### Algorithm 1. $SMatch$ && $SNMatch$

---

```

1: procedure  $SMatch(C[1], C[2] \dots C[k])$ 
2:   repeat ▷ use some variables to detect any match been skipped
3:     for each  $C[i]$  in  $C$  do
4:       repeat
5:          $tmp = C[i]; C[i] = tmp++;$  ▷  $tmp$  is a cursor
6:         until  $C[i] \not\prec_a tmp$  &&  $tmp \not\prec_p vr(C)$  ▷ Using rule 1 and 2 to skip
7:         end for ▷  $\leftrightarrow$  unnecessary matches
8:       until no match can be skipped
9:     return  $C$ 
10: end procedure

11: procedure  $SNMatch(C[1], C[2] \dots C[k])$ 
12:   Let  $C[l]$  be the  $vl$  node of  $C$ 
13:    $C[l]++;$ 
14:   return  $C$ 
15: end procedure

```

---

The general idea of our approach is showed in algorithm 2 MMPS. First, we get an initial match (at the very begin is  $\{S[1][1], S[2][1] \dots S[k][1]\}$ ). Then we use  $SMatch$  and  $SNMatch$  to get a candidate SLCA node in line 3-6 and validate the candidate on line 7, and update the final result in line 8 if true. From line 10 to 14, we calculate the next initial match, in which each node is a non-descendant of  $\alpha_a$ . By doing this repeatedly, we generate final results. This loop stops when any  $C[i]$  points to the end of  $S[i]$  which is caught by *try/catch* structure.

**Algorithm 2.** MMPS ( $S[1], S[2] \dots S[k]$ )

---

**Require:** each  $S[i]$  is the nodes list that responds to a input keywords  $k_i$

```

1: establish a cursor array  $C$ ;  $n=0$ ;  $\alpha=null$     ▷ each  $C[i]$  points to the first node of
2: repeat                                       ▷  $\hookrightarrow S[i]$ ,  $\alpha$  maintains the final result
3:   repeat
4:      $\alpha_a=lca(SMatch(C[1], C[2] \dots C[k]))$ 
5:      $\alpha_b=lca(SNMatch(C[1], C[2] \dots C[k]))$ 
6:   until  $\alpha_a \not\leq \alpha_b$ 
7:   if  $n==0$  or  $\alpha_a \not\leq \alpha[n]$  then
8:      $\alpha[n++] = \alpha_a$ ;
9:   end if
10:  for each  $C[i]$  do                            ▷ calculate the next initial match
11:    repeat
12:       $C[i]++$ 
13:    until  $\alpha_a \not\leq_p C[i]$ 
14:  end for
15: until some  $C[i]$  points to the end of  $S[i]$     ▷ try/catch structure should be used
16: return  $\alpha$                                   ▷  $\hookrightarrow$  to avoid boundary exceeded error

```

---

### 3.3 Analysis

Our approach is suitable for both Dewey and Region scheme but costs differently. For Dewey scheme, it costs  $O(d)$  time to compute the relationships (ancestor/descendant and precede/succeed) of two nodes as well as the LCA of two nodes ( $d$  is the average depth of the XML tree), while it costs  $O(1)$  time on Region scheme [1]. The  $vl$  and  $vr$  functions cost  $O(kd)$  time to compute the leftmost(rightmost) node of a match on Dewey, which only costs  $O(k)$  time on Region code ( $k$  is the keywords number). In MMPS, we do not enumerate all the cases but only consider the matches which are liable to be an SLCA candidate. Let  $S_{max}$  be the largest keyword node set and  $S_{min}$  be the size of the smallest set, then at most  $O(|S_{max}|)$  matches are visited which costs  $O(kd|S_{max}|)$  time on Dewey and  $O(k|S_{max}|)$  time on Region, and the whole candidate SLCA match number is no larger than  $|S_{min}|$ . So the time complexity of our approach is  $O(kd|S_{max}| + d|S_{min}|)$  on Dewey code and  $O(k|S_{max}| + |S_{min}|)$  on Region code. In comparison, the time complexity of IMS are  $O(k|S_{min}|\log(|S_{max}|))$  on Region code and  $O(kd|S_{min}|\log(|S_{max}|))$  on Dewey code, respectively.

## 4 Experimental Results

To verify the effectiveness of our approach, we implement both MMPS and IMS algorithm on Dewey and Region scheme and compare the results.

### 4.1 Experiment Setup

All Experiments are implemented in C++ language. We run experiments on the data sets of DBLP [8] with 8.8 million data nodes and XMark [6] with 3.6 million

data nodes. These two data sets are encoded with Dewey scheme and Region scheme, respectively. Similar to other SLCA-based algorithms, all the data nodes are organized as a B+-tree where keys are the keywords of the data nodes and data associated with each key is a list of Dewey codes ( or Region codes) of the data node contained that keyword. Our implementation use Berkeley DB (v4.8 C++ Edition) [5] to store the keyword data list. Berkeley DB is configured using 4KB size page and 1 GB size cache . All the experiments are done on a 2GHz dual-core laptop PC with 2 GB of RAM.

In particular, we pre-process the XML data and built an index to accelerate the *LCA* computation [2]. It costs 6.602 seconds to build the index for XMark data and 14.202 seconds for DBLP data. Before keyword search, index is loaded into memory first. It costs 4.218 seconds to load index of XMark and 9.839 seconds for DBLP. These extra works can be done off-line, and will not affect I/O time and query performance.

Similar to [9], experiments are carried out by using different classes of queries. Each class is written as *data\_kN\_L\_H*, where *data* denotes the XML data and *N*, *L* and *H* are three positive integer values: *N* shows the number of the keywords, *L* and *H* with  $L \leq H$  represent two keyword frequency that one of the *N* keywords has the low frequency *L* while each of the remaining  $N - 1$  keywords has the high frequency *H*.

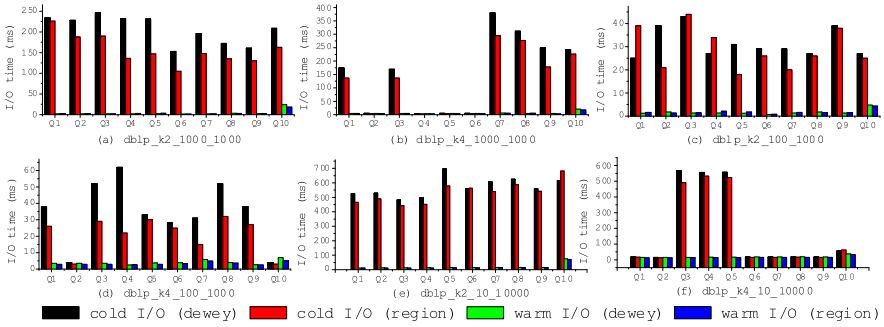


Fig. 2. I/O costs on DBLP

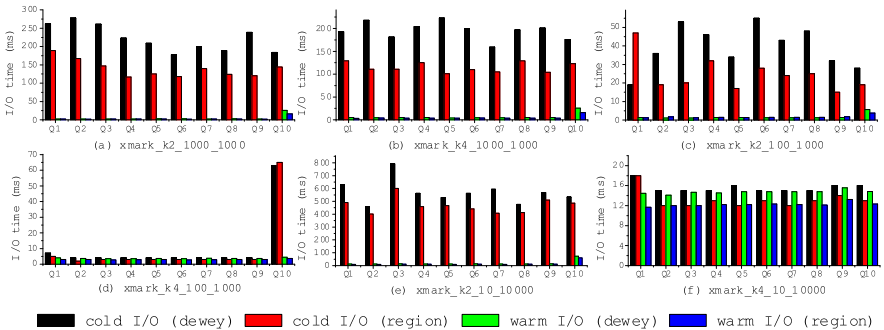


Fig. 3. I/O costs on XMark

### 4.2 Result Analysis

Figure 2 and 3 show the I/O cost of the two encoding schemes on DBLP and XMark. Generally speaking, MMPS on Dewey scheme costs more time than Region for the length of keywords indexed by Dewey scheme is not fixed and more time is needed when reading data from index. But there are exceptions in cold I/O for the initial position of the disk head is random. On the other hand, Region scheme has advantage in warm I/O because the initial position of the disk head is beside the data.

Figure 4 and 5 show the performances of two algorithms on both encoding schemes. First, it is clear that the *SLCA* calculation based on Dewey costs more time for in each comparison operation of data node or *LCA* computation, Dewey scheme costs  $d$  times than Region. Second, we can see that algorithm MMPS performs better than IMS in general. Though exceptions exist few, such as *Q5* in figure 4(d) and *Q10* in figure 5(e). However, in such cases, the time cost is few and IMS does not perform much better. Figure 4 shows the performances on DBLP and Figure 5 shows the performances on XMark. When the  $L$  value decreases, both algorithms trends to reduce the calculating time and MMPS performs better. In any cases, MMPS outperforms IMS.

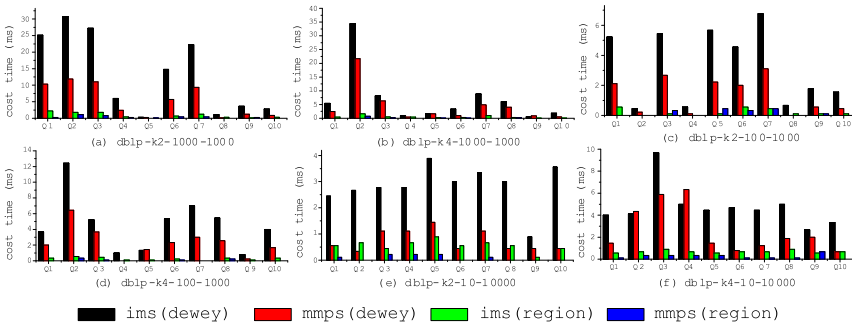


Fig. 4. Processing Time on DBLP

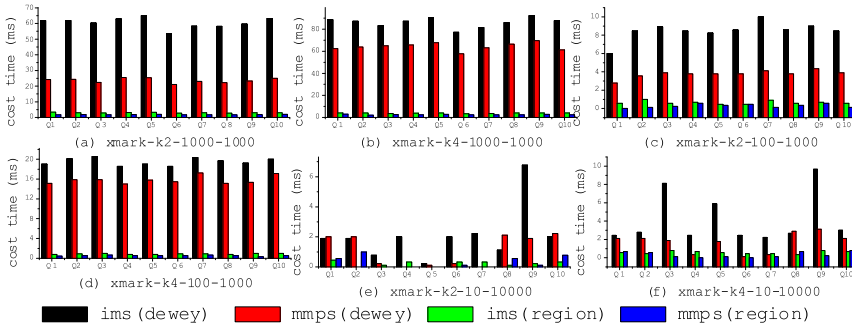


Fig. 5. Processing Time on XMark

## 5 Related Work

Recently, lots of research has been concentrated on semantics and techniques for keyword search over XML data, such as XRank [3], SLCA [10], VLCA [4], XSeek [7]. In these research, LCA computation is a common problem. Lots of research has been concentrated on the techniques for this problem [12].

Our work is closely related to two research work [10,9]. [10] proposes two efficient algorithms: the Scan Eager (SE) algorithm and the Index Lookup Eager (ILE) algorithm. [9] proposes a more efficient algorithm named IMS than the ILE and SE algorithms by using several optimization strategies to reduce the number of LCA computation. Moreover, [9] extended its algorithm to support more general search queries involving AND and OR semantics. In particular, the IMS algorithm is compared with in this paper.

## 6 Conclusions

In this paper, we propose a novel SLCA-based keyword search approach MMPS. Since MMPS utilize iterative skip to reduce redundant SLCA computations, it outperforms the state-of-the-art algorithm IMS. We also compare results of our method with Dewey and Region encoding schemes. Experimental results show that Region code is effective to compute SLCAs over XML data.

## Acknowledgements

The work was supported by the National Natural Science Foundation of China under Grants No. 60703093 and No. 60673133, the National Grand Fundamental Research 973 Program of China under Grant No. 2005CB321905.

## References

1. Bender, M.A., Farach-Colton, M.: The lca problem revisited. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 123–125. Springer, Heidelberg (2000)
2. Fischer, J., Heun, V.: Theoretical and practical improvements on the rmq-problem, with applications to lca and lce. In: Lewenstein, M., Valiente, G. (eds.) CPM 2006. LNCS, vol. 4009, pp. 36–48. Springer, Heidelberg (2006)
3. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over xml documents. In: SIGMOD 2003 (2003)
4. Wang, J., Li, G., Feng, J., Zhou, L.: Effective keyword search for valuable lcas over xml documents. In: CIKM 2007 (2007)
5. Berkeley DB: <http://www.oracle.com/database/berkeleydb/index.html>
6. XMark Project, <http://www.xmlbenchmark.org/>
7. Liu, Z., Chen, Y.: Identifying meaningful return information for xml keyword search. In: SIGMOD 2007 (2007)
8. DBLP XML records, <http://dblp.unitriuer.de/xml/>
9. Sun, C., Chan, C.-Y., Goenka, A.K.: Multiway slca-based keyword search in xml data. In: WWW 2007 (2007)
10. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD 2005 (2005)



# SFL: A Structured Dataflow Language Based on SQL and FP

Qiming Chen and Meichun Hsu

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

**Abstract.** SFL (pronounced as Sea-Flow) is an analytics system that supports a declarative language that extends SQL for specifying the dataflow of data-intensive analytics. The extended SQL language is motivated by providing a top-level representation of the converged platform for analytics and data management. Due to fast data access and reduced data transfer, such convergence has become the key to speed up and scale up data intensive BI applications.

A SFL query is constructed from conventional queries in terms of Function Forms (FFs). While a conventional SQL query represents a dataflow tree, a SFL query represents a more general dataflow graph. We support SFL query execution by tightly integrating it with the evaluation of its component queries to minimize the overhead of data retrieval, copying, moving and buffering, which actually turns a query engine to a generalized dataflow engine. The experimental results based on a prototype built by extending the PostgreSQL engine are discussed.

## 1 Introduction

A Business Intelligence (BI) application often form a dataflow process from collected data to derived information and decision. With the current technology, BI system software architecture generally separates the BI analytics layer (BI applications and tools) from the data management layer (DBMS or Data Warehouses), where applications are coded as database client programs in C, Java, T-SQL, P/L-SQL, etc, which invoke SQL statement for data retrieval. As the steep increase in the amount, the data transferred between the analytics platform and the database platform has become the scalability and performance bottleneck of BI applications.

Converging data-intensive analytics computation into the DB engine is the key to address these problems [2,4]. One option for the next generation BI system is to have the data intensive part of analytics executed inside the DB engine, which implies that the application dataflow and database access should be specified by a single, integrated declarative language. It is argued that SQL is a reasonable candidate for such a language.

In order to express analytics operations which are beyond the standard relational database operations, we rely on User Defined Functions (UDFs) [3]. However, a SQL

query is limited to express, and the existing query engine is limited to orchestrate the tree-structured dataflow; to handle complex, graph-structured inter-query dataflow inside the database system, extending the query language as well as the query engine is required.

To *declaratively* express complex dataflow graph, we introduce certain *construction primitives* and Function Forms (FFs) from the functional programming language FP [1] into the SQL framework. The extended SQL language uses these primitives and FFs to glue queries and UDFs based on a calculus of queries. We use this extended language as the top-level representation of a converged platform for analytics and data management called SFL (pronounced as Sea-Flow) platform. While a conventional SQL query represents a dataflow tree, a SFL query represents a more general dataflow graph declaratively. This extension has two important features: first, the extended language specifies complex application data flows using queries invoking functions *declaratively* rather than imperatively, thus isolates much of the complexity of data streaming into a well-understood system abstraction; second the same data object can “flow” to multiple operations without copying, repeated retrieval from database or duplicated query evaluations, which supports the justification of pushing analytics down to the query engine rather than at the client level.

The SFL query is supported in the SFL platform with an query engine with the functionalities that orchestrate actions (queries) interactively with the underlying query processing. Our experience shows the value of the proposed approach in multiple dimensions: modeling capability, efficiency, as well as usability; all these represent a paradigm shift from traditional BI in converging data-intensive analytics and data management.

## 2 SFL Language Framework

A SFL system is used to combine queries and user defined functions for representing application dataflow graphs. In the following we introduce the operators – functional forms, and operands – query functions and user-defined relational operator functions (RVFs), of the SFL language framework.

First, we rely on UDFs to extend the action capability of the database engine. In order to handle operations applied to a set of tuples (a relation) rather than a single tuple, we have introduced the kind of UDFs with input as a list of relations and with return value as a relation, called Relation Valued Functions (RVFs) [3]. RVFs can be integrated to SQL queries and treated as relational operators or relational data objects.

Next, we distinguish the notion of *Query Function* from the notion of *Query Variable*. A query variable is just a query such as

```
SELECT * FROM Orders, Customers WHERE Orders.customer_id = Customers.id;
```

that is bound to actual relations such as the above Orders and Customers. A query variable can be viewed as a relation data object, say  $Q_v$ , denoting the query result.

A query function is a function applied to a sequence of parameter relations. For instance, the query function corresponding to the above query can be expressed as

```
 $Q_f :=$  SELECT * FROM $1, $2 WHERE $1.customer_id = $2.id;
```

Then applying  $Q_f$  to a sequence of relations  $\langle$  Orders, Customers  $\rangle$  with matched schemas, is expressed by

```
 $Q_f : \langle$  Orders, Customers  $\rangle \rightarrow Q_v$ 
```

The major constraint of query functions is *schema-preservation*, i.e. the schemas of the parameter relations must match the query function. It is obvious that the above query function is not applicable to arbitrary relations.

Further, we introduce the notions of constructive-primitive and functional form. A constructive-primitive expresses the path of function application, or the path of dataflow. For example, applying a *Selector Function*  $\$i$  to a sequence of objects  $\langle r_1, \dots, r_n \rangle$  returns  $r_i$ ; applying the *Identity Function*  $id$  to object  $r$  returns  $r$  itself.

A functional form (or function combining form), FF, is an expression denoting a function; that function depends on the functions which are the parameters of the expression. Thus, for example, if  $f$  and  $g$  are RVFs, then  $f \bullet g$  is a functional form denoting a new function such that, for a relation  $r$ ,  $(f \bullet g):r = f:(g:r)$  provided that  $r$  matches the input schema of  $g$ , and  $g:r$  matches the input schema of  $f$ .

The dataflow shown in Fig. 1(a), where  $Q_f$  is a query function “SELECT \* FROM \$1, \$2”,  $f_1$  and  $f_2$  are RVFs with inputs  $r_1$  and  $r_2$ , can be expressed with the use of the *constructive function combining form* denoted as  $[ ]$ , which constructs a list from the content in the bracket:

$$Q_f \bullet [f_1 \bullet \$1, f_2 \bullet \$2] : \langle r_1, r_2 \rangle$$

that denotes  $Q_f(f_1(r_1), f_2(r_2))$ . Note that this statement could be expanded to

$$\text{“SELECT * FROM } f_1(r_1), f_2(r_2)\text{”}.$$

For the dataflow in Fig.1 (b) where  $r_2$  needs to feed into both  $f_1$  and  $f_2$ , the dataflow is forked. This dataflow is written as:

$$Q_f \bullet [f_1 \bullet id, f_2 \bullet \$2] : \langle r_1, r_2 \rangle$$

where  $id$  and  $\$2$  are primitive constructors applied to the argument list. In this case,  $id$  applied to  $\langle r_1, r_2 \rangle$  returns  $\langle r_1, r_2 \rangle$  itself, while  $\$2$  applied to  $\langle r_1, r_2 \rangle$  returns  $r_2$ , and thus  $Q_f$  applies to (or composes with) results of  $f_1$  and  $f_2$ , and  $f_1$  applies to *both*  $\langle r_1, r_2 \rangle$ , while  $f_2$  applies to second argument of  $\langle r_1, r_2 \rangle$ . We see that forking cannot be accomplished without extension to SQL.

Further, suppose we need to express the dataflow shown in Fig 1(c) where  $Q_f$  has three input objects, and  $r_1$  needs to feed into both  $f_1$  and  $Q_f$ . Then the query statement would be

$$Q_f \bullet [\$1, f_1 \bullet id, f_2 \bullet \$2] : \langle r_1, r_2 \rangle$$

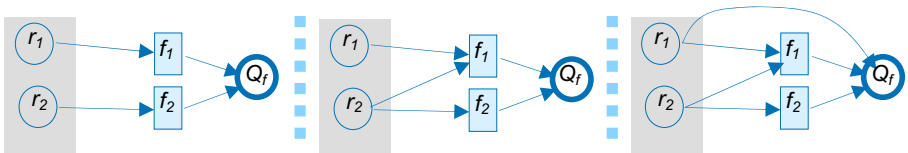


Fig. 1. (a) dataflow without fork (b) dataflow with fork (c) dataflow with fork & crossover

In general, a SFL system is founded on the use of a fixed set of FFs for combining query functions (a query can invoke RVFs). These, plus simple definitions, provide the simple means of building new functions from existing ones; they use no variables or substitution rules, and they become the operations of an associated algebra of

queries. All the functions of a SFL system are of one type: they map relations into relations; and they are the operands of FFs. Applying an FF to functions denotes a new function; and applying that function to relations denotes a *SFL query*. A SFL query has the expressive power for specifying a dataflow graph, in a way not possible by a regular query. Specifically, a regular SQL query represents a tree-structured dataflow, while a SFL query can further represent graph structured dataflow. A more formal description can be found in [2].

### 3 An Example

In this section we shall illustrate how to express the dataflow scheme of K-Means clustering using a single SFL statement and discuss our experimental results. The *k*-means algorithm illustrated in Fig 2 is an algorithm to cluster *n* objects based on attributes into *k* partitions,  $k < n$ . It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data. It assumes that the object attributes form a vector space.

The K-Means computation used in this example has two original input relations, *Points* and *Centers*, and two relational operations, *cluster* and *check*. The *cluster* operation generates a new set of centers to be compared with the existing centers by the *check* operation; if they have not converged enough, the *cluster* will be re-run iteratively with the same *Points* data and the new centers. Note that SQL is unable to express the iteration.

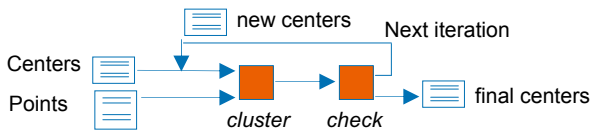


Fig. 2. K-Means clustering

Let us consider the K-Means calculation on two dimensional geographic points. The original *Points* data are stored in relation *Points* [*x*, *y*, ...], and the *Centers* data are stored in relation *Centers* [*cid*, *x*, *y*, ...]. Two RVFs are involved. The RVF

$$cluster\_rvf: \langle Points, Centers \rangle \rightarrow Centers'$$

is used to derive a new set of centers, i.e. a new instance of relation *Centers*, from relations *Points* and *Centers* in a single iteration, in the following two steps:

- the first step is for each point in relation *Points* to compute its distances to all centers in relation *Centers* and assign its membership to the closest one, resulting an intermediate relation for new centers [*x*, *y*, *cid*];
- the second step is to re-compute the set of new centers based on the average location of member points.

After an iteration, the newly derived centers (in relation *Centers'*) are compared to the old ones (in relation *Centers*) by another RVF

$$check\_rvf: \langle Centers', Centers \rangle \rightarrow \{T; F\}$$

for checking the convergence of the sets of new and old centers to determine whether to terminate the K-Means computation or to launch the next iteration using the current centers, as well as the original points as the input data.

Our goal is to define a SFL query  $kmeans : \langle \text{Points}, \text{Centers} \rangle$

that derives, in multiple iterations, the Centers of the clusters with minimal total intra-cluster variance, from the initial Points and Centers relations, towards the final instance of relation Centers. During the  $kmeans$  computation, the relation Centers is updated in each iteration, but the relation Points remains the same. A key requirement is to avoid repeated retrieval of either the Centers relation or the Points relation from the database, which should be explicitly expressible at the language presentation level.

The function  $kmeans$  is defined by the following

```

clustering := [$1, $2, cluster_rvf];
iterating := (check_rvf • [$2, $3] → $3; iterating • clustering • [$1, $3]);
kmeans := iterating • clustering;

```

Applying function  $kmeans$  to the points and the initial centers for generating the converged center positions is expressed by the SFL query

$kmeans : \langle \text{Points}, \text{Centers} \rangle$

The execution of SFL query  $kmeans : \langle \text{Points}, \text{Centers} \rangle$  is depicted in Fig 6. As described later, the different versions of the instances of relation Centers output from  $cluster\_rvf$  in multiple iterations, actually occupy the same memory space.

## 4 Implementation Issues

Introducing SFL system aims to support general graph structured dataflow. From the implementation perspective, the key point consists in controlling the sharing and the buffering of data with desired life-span, in order to ensure that data is “pipelined” in the data flow rather than unnecessarily copied, repeatedly retrieved or duplicated evaluated.

We extended the PostgreSQL engine to support RVFs for integrating applications into queries, and to support SFL queries for expressing application data flows. We have extended the query engine by providing the RVF Manager and the SFL Manager. As we have reported the support to RVF in [3], in this paper we focus on the implementation of SFL Manager.

### 4.1 SFL Query Memory Context

A SFL query invokes queries during its execution. The memory context of a SFL query execution is accessible to the invoked queries. Such memory sharing allows the data passed in or returned from an invoked query to be referenced without copying, and carried across multiple queries without repeated derivation.

When a SFL query, say,  $Q_{ff}$ , is to be executed, an instance of it is created with a memory context for holding the relation data retrieved from the database or generated by the component queries. Based on PostgreSQL internal, the common data structure

for holding these relation data is called *tuple-store*. A tuple-store is an in-memory data structure with memory overflow handled automatically by the underlying query executor (e.g. expanded to temporal files). The system internal query facility, such as PostgreSQL SPI (Server Program Interface), is extended with tuple-store access-method, and the handles of multiple related queries are made sharable at the SFL layer.

During the execution of  $Q_{ff}$ , all the component query results are kept in tuple-stores. A query invoked by  $Q_{ff}$ , say  $q$ , does not copy input data into its own memory context; instead the data is passed in by reference. More concretely, the memory context of  $Q_{ff}$ , say  $MC$  and that of  $q$ , say,  $MC_q$  have the following relationship.

- The life-span of  $MC$  covers that of  $MC_q$ .
- The tuple-stores in  $MC$  are used by  $q$  as required; that is, the input relations of  $q$  are directly obtained from  $MC$  with pass-by-reference; the output relation of  $q$  is returned to  $MC$ .

Specifically, associated with a SFL query instance is a handle structure for its execution environment,  $h_{env}$ , with a pointer to its memory context handle, say  $h_{mc}$ . Associated with a component query is also a handle for holding its own execution information, and that handle has a pointer to  $h_{mc}$  for accessing and updating the data flowing through the SFL query process.

The mechanisms for PostgreSQL memory management, including the mechanism for managing shared buffer pool, are leveraged. The cached data are kept in the memory resided tuple-stores. An overflow tuple-store can be backed up by a file automatically and transparently with the same access APIs. This ensure our system survive with large initial data. In fact, different from data warehousing, the basic principle in dealing with dataflow applications is to keep data “moving” rather than “staying”, therefore most dataflow applications are designed to fit in moderate on-the-flow buffer size.

## 4.2 SFL Query Data Flows

The functionality difference of a query/RVF and a constructive primitive has been explained above. Here we examine it from the dataflow point of view.

- A data object  $R$  flowing through a query or RVF means that the query or RVF produces an output relation out of  $R$ ;
- $R$  flowing through a constructive primitive  $G$  such as  $id$  or  $\$I$  means that  $R$  is to remain referenced, and thus active; what actually passed through  $G$  is the reference, or pointer if handled in a single query engine, of  $R$ ; such reference-flow mechanism ensures that  $R$  is not to be copied; and based on the *reference-counting* mechanism, garbage-collected when not being referenced anymore (following a common rule, we do not distinguish the value and the reference of a primitive typed object such as an integer used as a function argument).

To be concrete, let us refer to Fig 3 and consider two data buffer pools  $POOL_1$  and  $POOL_2$ , i.e. the memory closures for *clustering* and *iterating*. The original relations *Points* ( $P$ ) and *Centers* ( $C$ ) retrieved from the database are buffered at  $POOL_1$ , the intermediate relation new centers derived by RVF *luster\_rvf* in each iteration is buffered at  $POOL_2$ . The data flows of  $P$  and  $C$  can be described as below.

- The flow of Points data P:
  - o at  $POOL_1$ , P is referenced by functions *cluster\_rvf* and *clustering* thus active;
  - o at  $POOL_2$ , P is still referenced by function *iterating* thus active;
  - o if a new iteration is launched, P is not reproduced since it is then referenced by *clustering* again as its  $\$1$  argument, thus kept active rather than reproduced;
  - o therefore, P is buffered only once after retrieved from the database, with its reference “flows”.
- The flow of Centers data C:
  - o at  $POOL_1$ , C is referenced by functions *cluster\_rvf* and *check\_rvf*, thus active;
  - o a new version of C, C', is generated by *cluster\_rvf* at  $POOL_2$ ;
  - o after *check\_rvf* is executed, C is not referenced by anyone so will be garbage collected or refreshed;
  - o if a new iteration is launched, the data of C' flow to  $POOL_1$  to refresh the buffer.

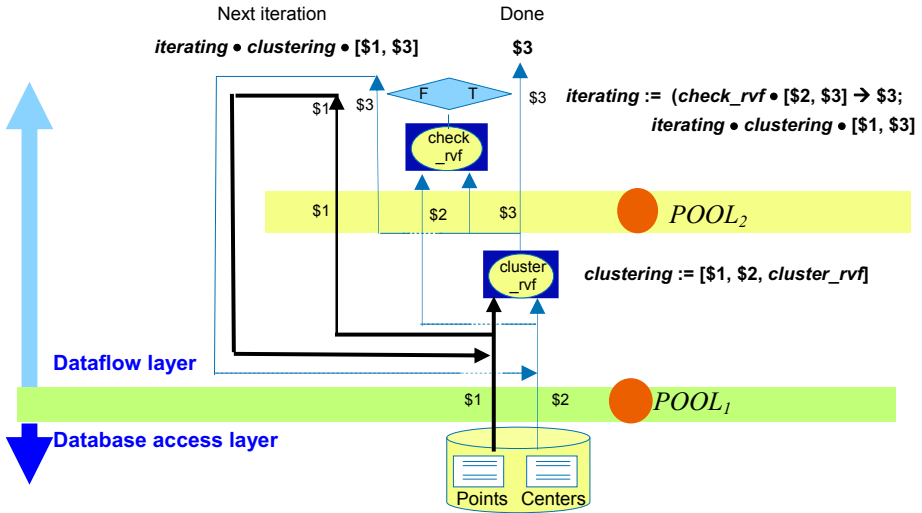


Fig. 3. K-Means dataflow without repeated querying and database retrieval

### 4.3 SFL Query Execution

To support SFL, we have plugged a preliminary Structured Dataflow Manager (SFM) in the PostgreSQL query engine, with two major components:

- FH – i.e. Flow Handler for creating and managing SFL instances, and for scheduling operations;
- QH – i.e. Query Handler for launching queries (possibly involving RVFs) using the PostgreSQL internal SPI query facility extended with tuple-store access.

With the FH and QH, a SFL query is scheduled in the following way.

- Based on the template of a SFL query, the FH creates an SFL query process *instance* and derives one control-sequence of its operations, resulting in an execution plan.
- The FH creates the start operation instances based on the control-sequence.
- For running every operation, the FH identifies the incoming tuple-stores, identifies or creates the outgoing ones, generates a corresponding *operation-item*, and puts it to the *operation queue*. At the minimum, an operation item includes the ID of the containing query-instance (i.e. the SFL query instance), the operation-instance-ID, the query (may involve RVFs) of this operation, and the references to incoming and outgoing tuple-stores (a reference is a key name, not a physical pointer).
- The operation items, once put in the operation queue, can be executed in any order, and in fact they are de-queued in pipeline and executed by individual, time-overlapping threads.
- The QH runs as a separate thread of the FH, it de-queues the operation items one by one; for each item, it launches a thread to execute the query associated with that item, using the high-efficient PostgreSQL internal SPI facility (extended with tuple-store access), then puts the returned query result into the outgoing tuple-stores, and sends a return value, *rv*, to the FH. The return value is a message that contains, at the minimum, the process-instance-ID, the operation-instance-ID, the query execution status returned from SPI, and the references to outgoing tuple-stores.
- Upon receipt of an *rv*, the FH updates the corresponding operation instance and process instance, checks the control-sequence and triggering condition or timer, then selects the next eligible operation or operations to run, rolling forward the process instance, towards the end of it.

The SFM is actually a “*middleware inside the query engine*” for handling multiple SFL queries with interleaved operation executions. Its scheduling mechanism is similar to that of a BPM; however, the running environments of a business process and its tasks are in general isolated, but the running environments of a SFL query and a component query is joined.

## 5 Conclusions

To effectively handle the scale of analytical tasks in an era of information explosion, pushing data-intensive analytics down to the database engine is the key. In this work we tackled two problems for converging analytics and data management: first, integrating general analytic operations into SQL queries, and second, extend SQL framework to express general graph structured dataflow.

We support SFL by building a super-query processing container inside the database engine that deals with data buffering, dataflow, control-flow and function orchestration. During execution, the memory context of a SFL query is tightly integrated with that of the queries it invokes, which effectively eliminates the overhead for data copying and duplicated retrieval or derivation.



The advantages of SFL lies in its expressive power for specifying complex dataflow, as well as in its simplicity, as it uses only the most elementary fixed naming system (naming a query) with a simple fixed rule of substituting a query for its name. Most importantly, they treat names as functions that can be combined with other functions without special treatment.

## References

- [1] Backus, J.: Can Programming Be Liberated from the von. Neumann Style? A. Functional. Style and Its. Algebra of Programs. ACM Turing award lecture (1977)
- [2] Chen, Q., Hsu, M.: Cooperating SQL Dataflow Processes for In-DB Analytics. In: CoopIS'09 (2009)
- [3] Chen, Q., Hsu, M., Liu, R.: Extend UDF Technology for Integrated Analytics. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 256–270. Springer, Heidelberg (2009)
- [4] DeWitt, D.J., Paulson, E., Robinson, E., Naughton, J., Royalty, J., Shankar, S., Krioukov, A.: Clustera: An Integrated Computation And Data Management System. In: VLDB 2008 (2008)


# Incorporating Data Provenance in a Medical CSCW System

Konrad Stark<sup>1</sup>, Christian Koncilia<sup>3</sup>, Jonas Schulte<sup>2</sup>,  
Erich Schikuta<sup>1</sup>, and Johann Eder<sup>3</sup>

<sup>1</sup> University of Vienna, Department of Knowledge and Business Engineering

<sup>2</sup> University of Paderborn, Heinz Nixdorf Institute

<sup>3</sup> Alps Adria University Klagenfurt, Department of Informatics Systems

**Abstract.** Medical research is a highly collaborative process in an interdisciplinary environment that may be effectively supported by a Computer Supported Cooperative Work (CSCW) system. Research activities should be traceable in order to allow verification of results, repeatability of experiments and documentation as learning processes. Therefore, by recording the provenance of data together with the collaborative context it is embedded into, novel types of provenance queries may be answered. We designed and implemented a next-generation CSCW system providing both the collaborative functionalities as well as the definition and execution of structured processes. We integrated a data provenance model recording process- and collaboration-related operations automatically and demonstrate the capabilities of the model by answering specific data provenance queries from the biomedical domain. 

## 1 Introduction

Scientific research projects are complex processes. Researchers from different domains (e.g. medical, biological, technical) collaborate by providing their personal expertise, methods and skills. In this highly interdisciplinary environment, the quality of results strongly depends on various factors. On the one hand, data, intermediate results, hypotheses, and conclusions must be shared efficiently. On the other hand, all participants require a basic understanding on the objectives and methodology of the other partners. The former may be accomplished by deploying traditional cooperative systems, which allow data sharing for a project team and tracing data changes. The latter one requires a comprehensible representation of the working processes, which is typically offered by scientific workflow management systems. However, workflow management systems do not cover all aspects necessary for collaborative projects. For instance, medical studies require flexible integration of patient and experiment data and its cooperative annotation. The research methodology of studies must be documented in detail in order to allow verification of results and repeatability of experiments. Therefore, data provenance recording may provide an essential support. Provenance

---

<sup>1</sup> This work was partially supported by the Austrian Ministry of Science and Research; program GEN-AU, project GATIB.

may be defined as *the background knowledge that enables a piece of data to be interpreted and used correctly within context* [9]. It allows to protocol the origin of the data, the way of accessing it, and finally the transformation process of the data. We call this sequence of processing steps a “*construction plan*” of the data object. These processes may be restarted with slightly modified parameters, and, if intermediate results are preserved, re-executions may be optimized [1]. In Stevens et al. [8] *Organizational provenance* is pointed out as an additional type of provenance. Organizational provenance comprises information about who has transformed which data in which context. This kind of provenance is closely related to collaborative work.

With respect to the particular demands of medical research projects, we designed and implemented a next-generation Computer Supported Cooperative work (CSCW) system providing both the collaborative functionalities as well as the definition and execution of structured processes. We integrated a data provenance model recording process- and collaboration-related operations automatically. Project participants are able to trace from a certain result back to the original data that was used to produce it. By combining transformation and versioning paths of data with collaborative information and semantic annotations, new types of data provenance queries can be formulated. Moreover, inter-project collaboration is encouraged by transparently sharing data, results and processes with other research projects and mutually benefiting from expertise.

This paper is organized as follows: in section 2 we give an overview of the main functionality of our CSCW system and systematically elaborate the main requirements for our data provenance model, which is presented in section 3. We entail use cases for data provenance queries in section 4 and show how the queries are answered. Before we investigate similar approaches in section 5 and conclude our results in section 6.

## 2 GAMECS CSCW System

The GAMECS CSCW [15,12] system was developed in the context of the biobank initiative GATiB (Genome Austria Tissue Bank), which is part of the Austrian Genome Program (<http://www.gen-au.at>). The main idea was to establish an information system that is capable of supporting the medical research activities in the environment of the biobank. GAMECS bases on the WasabiBeans framework (<http://www.wasabibeans.de>), a flexible service-oriented architecture developed at the university of Paderborn [11,10]. WasabiBeans is widely used in research projects to share data in distributed heterogenous system infrastructures. An example field of application of WasabiBeans is the integration of a high-tech laboratory and a digital library to provide up to date measurement results of thermal stress test within an university-wide IT infrastructure [14]. It implements the concept of virtual knowledge spaces for a flexible structuring of data and supports users to aware each other for a better cooperative work process. In our project we use WasabiBeans as a middleware integrating distributed data sources and biomedical services. GAMECS facilitates flexible structuring

of data and services in knowledge spaces that are used for cooperations. During the work of medical researches with the system, a set of additional requirements have arisen. Integrating process patterns into the CSCW system and the traceability and reproducibility of research results were requested. Therefore, we developed a semantic model that enhances the capabilities of the CSCW system by integrating data provenance and process definition and enactment. Thus, we were able to cover the following aspects in our CSCW system:

1. *Semantic Annotation*: We encourage collaboration by creating virtual knowledge spaces that are filled with data resources and services. A knowledge space can be created for a particular project or study. We support various ways of annotating contexts semantically. First, a categorized tag repository allows the assignment of categories (and subcategories) that characterize contexts. Second, free text annotations containing textual descriptions. Finally, the content of contexts is searchable. Thus, it is possible to look for all contexts, where the same set (or subset) of data resources or services is used.
2. *Process Definition*: Processes are assigned to collaborative knowledge spaces and can be executed by members of the corresponding knowledge space. Well established procedures can be implemented as standardized processes. Dedicated processes may be provided to untrained persons for improving the learning process. A sandbox like testing environment enables to trace, re-execute and modify processes and immediately view the created results.
3. *Process Traceability*: We designed appropriate data structures for logging the execution of web service and store input parameters and generated data, allowing a detailed view on how data resources were created.
4. *Process Repeatability*: Since the execution sequence of services is stored as well as the parameter settings, it is possible to repeat the execution of processes or even subprocesses. Verification of analysis results may be accomplished as well as process re-execution with slightly modified parameters.

### 3 Data Provenance Model

The data provenance model includes meta data about data resources, web services, processes and collaboration contexts. We designed the provenance model by using OWL-DL [2]. On the one hand OWL-DL is a powerful knowledge representation language that may be used to capture domain expertise, which is particularly complex in the field of medical research. Although various medical classification such as the ICD-O-3 (International classification of diseases for oncology) [7] exist, not all medical terms may be covered by standards. Some elements of the medical terminology require the ability to define and test for restrictions and relationships. OWL-DL allows to model detailed and restrictable relationships and facilitates verification of instance data as well as classifying data by subsumption. On the other hand, our model includes several transitive relationships that may be efficiently handled by reasoners. Moreover, we wanted

**Table 1.** Context Perspective Properties

<code>context_label:{string}</code>	Name of study or project	
<code>context_description:{string}</code>	Detailed context description	
<code>has_user(User)</code>	Assigned user	
<code>has_service(Service)</code>	Assigned web service	
<code>has_resource(Resource)</code>	Assigned resource	
<code>is_subcontext(Context)</code>	Context is embedded into another context.	Trans.
<code>related_context(Context)</code>	Context is related to other context(s).	Trans.

to benefit from the powerful query languages that may be applied to ontologies (e.g. SPARQL). In the following section we present the details of the model structured by its main perspectives.

### 3.1 Context Perspective

As a context we understand virtual knowledge space that are created for sharing data, knowledge and services for a group of cooperating persons. Furthermore, knowledge spaces may be nested into each other to form context hierarchies. Following we notate class names in italic letters, data properties are specified as `data_property_name: {type}`, object properties are notated as `property_name(Class_name)`, and transitive properties are marked as Trans.

### 3.2 Resource Perspective

Various kinds of resources are integrated into knowledge spaces. External data repositories may be included by describing the structure of data and by defining appropriate access patterns. Data may be extracted from semi- or unstructured documents and stored in novel data structures. The type of a resource may be one of the *primitive* types Integer, Float, Bool, Date, DateTime or String, a *file* type or a *complex* data repository type. Primitive types are typically used in order to store the input/output parameter values for web services. File types are assigned to documents that are shared in virtual knowledge spaces, whereas an internal version control management keeps track of all document versions. Complex resources are structured documents (XML or CSV files) or external database tables. The structure of complex types is defined by mapping (database) attributes to resources.

### 3.3 Web Service Perspective

Web services are ideal to facilitate application to application communication. They are highly standardized interfaces enhancing interoperability of applications and overcoming barriers of data exchange format, programming languages and operating systems. Although web services allow the flexible deployment of applications in a distributed environment, they raise novel challenges in the areas of traceability and reproducibility. For instance, an external web service may use a new version of an algorithm delivering slightly different results.

**Table 2.** Web Service Perspective Properties

<code>webservice_id:{string}</code>	Unique URI for identification
<code>webservice_name:{string}</code>	Label used for web service composition
<code>webservice_description:{string}</code>	Description of the functionality of the service
<code>state:{stateless,stateful}</code>	Different or identical results at each run.
<code>runtime:{immediate,short,medium,long}</code>	Runtime categories for web service execution.
<code>record_provenance:{yes,no}</code>	Enable or disable provenance recording.
<code>endpoint_references:{string}</code>	References to physical web service instances
<code>has_input_parameter(Type)</code>	Web service input parameters
<code>has_output_parameter(Type)</code>	Web service output parameters
<code>has_web_service_call(Web_Service_Call)</code>	Service execution entry

A web service may be described conceptually by a separate class *Web\_Service*, defining its input and output parameters as well as a short textual description of its task (see table 2). Similar to the provenance model of Taverna [6], we describe services as *stateless* or *stateful*. Stateless services are assumed to produce the same result for the same input parameters, while stateful services may return different results. The annotation is reasonable for both data quality and optimization aspects. On the one hand, the execution of a process that solely consists of stateless web service calls, is guaranteed to produce identical results in each run. On the other hand, the runtime of a service is an important characteristic. Provenance data of stateless steps may be recorded or re-materialized on demand and thus the the results may be recorded or not depending on the storage requirements and the computational complexity. We suggest to assign runtime annotations to services in order to support optimized process executions. A web service may have several physical instances which are stored in the collection of `endpoint_references`. Each execution of a web service is recorded in a `Web_Service_Call` instance by using predefined properties. The execution time of a service call may be easily calculated as the difference of `end_timestamp` and `start_timestamp`. The actual parameter values are mapped by the corresponding `has_input` and `has_output` properties.

## 4 Querying Provenance Data

Querying transformation paths is a complex task that involves graph traversal and filter operations. Therefore, we can utilize the expressiveness of ontologies and the capabilities of ontology reasoners (e.g. Pellet). An ontology reasoner may infer relationships by combining the structural knowledge of an ontology (T-Box) with the information of actual instances (A-Box). If, for a certain resource (*resourceB*) all services and resources should be detected that have been involved in producing the resource the query should return its detailed transformation path. As the sequence of web service calls is defined by transitive defined *m* : *has\_predecessor* object properties, a reasoner is able to deduce the entire set of predecessors of a certain web service call. Afterwards, all web services and associated input resources may be extracted by the following query:

```

SELECT ?web_service ?resource WHERE {
  ?web_service_call0 rdf:type m:Web_Service_Call ;
  ?web_service_call m:has_output m:resourceB.
  ?web_service_call1 rdf:type m:Web_Service_Call ;
  m:webservice_call_timestamp ?datetime .
  m:service_call0 m:has_predecessor ?web_service_call1 .
  ?web_service rdf:type m:Web_Service;
  m:has_call ?web_service_call.
  ?resource rdf:type m:Resource .
  ?web_service_call1 m:has_input ?resource .
}
ORDER BY ASC(?datetime)

```

The query returns a list of web service - resource pairs ordered by the execution time of web service calls. Starting with the web service call *?web\_service\_call0* that produced the resource *resourceB*, all preceding web service calls are extracted as well as its associated input resources.

In the next example query, all contexts, in which a certain resource *document\_1* is shared, should be returned. However, we are only interested in contexts which have an associated annotation text containing “liver cancer” or which have sub-contexts with this annotation. SPARQL offers path-oriented filter patterns for graphs allowing to select elements along predefined traversal paths. Since contexts may be nested into each other up to an arbitrary depth, all sub-contexts of a context may be identified by the inverted *^m : is\_subcontext\** property pattern. No explicate reasoning step is executed in this case, as only the transitive closure of a single property is required, which may be determined by the lightweight internal reasoner of Jena. Alternatively, an external reasoner can be applied to infer all *is\_subcontext* relationships.

```

SELECT ?context ?annotation_text WHERE {
  ?context ^m:is_subcontext*/m:has_resource m:document_1 .
  ?context m:has_annotation/m:annotation_text ?annotation_text .
  FILTER regex(?annotation_text, "liver cancer", "i")
}

```

## 5 Related Work

Much research has been done in the area of data provenance in recent years. Foster et al. [3] proposed the Chimera virtual data system that is capable of storing and tracing the transformations of scientific data. In contrast to the Chimera approach we enhance our model with aspects of collaborative work and support reasoning on meta data. Another provenance recording component was developed for the Kepler system [1]. Kepler allows the definition of scientific workflows by arranging tasks in sequences. Provenance data is collected for workflow instances. The Kepler system allows to switch on and off the recording of intermediate result. In our model, we distinguish between services that are guaranteed to deliver the same results for given input parameters, and services that can deliver divergent results. The scientific workflow system Taverna

incorporated a data provenance model recently [6]. Two variants of the provenance model are presented: firstly a baseline model that may be used to answer typical data provenance queries. And secondly, an enhanced model that assigns lightweight annotations to the processing steps. The former model represents the data-flow along the executed workflow steps. Data dependencies are collected by recording the input and output data in relations. In the enhanced provenance model, the capabilities of the model are extended by semantic information.

In the work of [4] OWL-DL based meta data is used for the execution of scientific workflows. This approach mainly concentrates on defining constraints for workflow steps. Each step has input and output data types that may be described by semantic annotations. In contrast, our model includes even more fine-grained modeling of data, e.g. structured files or database records. While our efforts are in the areas of process traceability and re-execution, their work enables to validate processes. Rajbhandari et al. [13] presented how data provenance can be captured in service-oriented architectures. The execution of services is accomplished by a workflow engine, whereas each service call is protocoled by a provenance capturing service. Though, the details of the model are not presented. An interesting integration of provenance in workflows was proposed in [5]. The presented architectures provides enhancement of workflows which are based on semantic web services. Since this work was influenced by SOA/Grid research, emphasis is put on definition and execution of workflows on service-oriented architectures. There is an overlap with our data provenance model, since we also capture service execution sequences as well as metadata about the input and output parameters and support queries over both kind data. However, we have integrated information about collaboration acts in the model. Thus, we are able to answer queries that are beyond typical process-oriented views. Further, similar contexts may be detected by taking into account collaboration information.

## 6 Conclusion

Research on data provenance has gained in importance in recent years. Particularly, in IT systems supporting research in natural sciences, the necessity of tracing back data to its origins was recognized. We presented a semantic data provenance model enhancing the capabilities of the medical CSCW system GAMECS. We demonstrated, how the OWL-DL based model may answer complex queries with the support of reasoners and path-traversal query patterns. When combining process-related data with semantic, context-related data, novel types of data provenance queries may be formulated that are beyond the typical generation of data derivation paths. Semantic annotations allow to categorize collaboration contexts, data, processes and services, making them comparable to each other and enabling inference of tacit relationships between them. Using data provenance information during process enactment is a promising approach that may be handled in more detail in future work. For instance, strategies are required determining which kind of intermediate results should be stored in which locations in order to have an optimal execution performance.



## References

1. Altintas, I., Barney, O., Jaeger-Frank, E.: Provenance collection support in the kepler scientific workflow system (2006)
2. World Wide Web Consortium. Owl web ontology language guide (2004)
3. Foster, I., Vöckler, J., Wilde, M., Zhao, Y.: Chimera: A virtual data system for representing, querying, and automating data derivation. In: Proc. of the 14th Conference on Scientific and Statistical Database Management (2002)
4. Kim, J., Gil, Y., Ratnakar, V.: Semantic metadata generation for large scientific workflows. In: Proc. of the 5th International Semantic Web Conference, ISWC-2006, Athens, GA, USA (2006)
5. Liming, C., Xueqiang, Y., Feng, T.: A semantic web service based approach for augmented provenance. In: WI '06: Proc. of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2006)
6. Missier, P., Belhajjame, K., Zhao, J., Roos, M., Goble, C.: Data lineage model for taverna workflows with lightweight annotation requirements (2008)
7. World Health Organization. Who: International classification of diseases for oncology, icd-o-3 (2000)
8. Goble, C., Stevens, R., Zhao, J.: Using provenance to manage knowledge of in silico experiments. *Brief Bioinform.* 8(3) (2007)
9. Ram, S., Liu, J.: 2(3) (2008)
10. Schulte, J., Dopke, I., Keil, R., Stark, K., Eder, J.: Enhanced security management for flexible and dynamic cooperative environments. In: CollaborateCom 2009 Proc. (2009)
11. Schulte, J., Hampel, T., Bopp, T., Hinn, R.: Wasabi framework – an open service infrastructure for collaborative work. In: SKG '07 Proc., Xi'an, China 0-7695-3007-9. IEEE Computer Society, Los Alamitos (2007)
12. Schulte, J., Hampel, T., Stark, K., Eder, J., Schikuta, E.: Towards the next generation of service-oriented flexible collaborative systems - a basic framework applied to medical research. In: ICEIS, vol. (1) (2008)
13. Shrija, R., Walker David, W.: Incorporating provenance in service oriented architecture. In: NWESP '06: Proc. of the International Conference on Next Generation Web Services Practices, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2006)
14. Sommerkamp, H., Schulte, J., Keil, R., Rybka, J., Ferber, F.: Ltm-sola - a service-oriented application to integrate high-tech laboratories and virtual knowledge spaces. In: CollaborateCom 2009 Proc. (2009)
15. Stark, K., Schulte, J., Hampel, T., Schikuta, E., Zatloukal, K., Eder, J.: Gatib-csw, medical research supported by a service-oriented collaborative system. In: Bellahsene, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 148–162. Springer, Heidelberg (2008)

# Optimization of Object-Oriented Queries through Rewriting Compound Weakly Dependent Subqueries

Michał Bleja<sup>1</sup>, Tomasz Kowalski<sup>1,2</sup>, and Kazimierz Subieta<sup>3,4</sup>

<sup>1</sup> Faculty of Mathematics and Computer Science, University of Łódź, Poland  
blejam@math.uni.lodz.pl

<sup>2</sup> Computer Engineering Department, Technical University of Łódź, Poland  
tkowals@kis.p.lodz.pl

<sup>3</sup> Polish-Japanese Institute of Information Technology, Warsaw, Poland

<sup>4</sup> Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland  
subieta@pjwstk.edu.pl

**Abstract.** A new static optimization method for object-oriented queries is presented. We deal with a special class of subqueries of a given query called “compound weakly dependent subqueries”. The dependency is considered in the context of SBQL non-algebraic query operators like selection, projection, join, etc. A subquery is weakly dependent from its nearest non-algebraic operator if it depends only on expressions that can be entirely evaluated on small collections. The subquery is considered compound if the dependency concerns at least two such expressions. The research follows the stack-based approach (SBA) to query languages and its query language SBQL (Stack-Based Query Language). Our optimization method is based on analyzing scoping and binding rules for names occurring in queries.

**Keywords:** query optimization, weakly dependent subqueries, object-oriented database, stack-based approach, SBQL.

## 1 Introduction

Efficient query evaluation is a very desirable and frequently critical feature of database management systems. The performance can be supported by various methods (indices, parallel execution, etc.). In this paper we consider query optimization based on query rewriting. Rewriting means transforming a query  $q_1$  into a semantically equivalent query  $q_2$  ensuring much better performance. It consists in locating parts of a query matching some pattern. Such optimization is a compile time-action performed before a query is executed. It requires performing a special phase called *static analysis* [10].

Our optimization method is based on the Stack-Based Approach (SBA) ([11], [12], [13]). SBA and its query language SBQL are the result of investigation into a uniform

---

<sup>1</sup> The author is a scholarship holder of project entitled “Innovative education ...” supported by European Social Fund.

conceptual platform for an integrated query and programming language for object-oriented databases. Currently a similar approach is developed in the Microsoft LINQ project [8] that integrates a query language with .Net programming languages (and recently with Java). One of the most important concepts of SBA is an *environment stack* (ENVS), known also as *call stack*. The approach respects the *naming-scoping-binding* paradigm, what means that each name in a query or program code is bound to a suitable run-time entity (e.g. object, attribute, variable, procedure, view etc.) depending on the scope for the name.

Analyzing query processing in the Stack-Based Approach it can be noticed that some subqueries are evaluated many times in loops implied by non-algebraic operators despite their results are the same in each loop iteration. This observation is the basis for an important query rewriting technique called the method of independent subqueries ([9]). It is also known from relational systems ([5], [6]) in a less general variant. In SBA this method is generalized for any kind of non-algebraic query operators and for very general object-oriented database model.

In [3] we present the generalization of the independent subqueries method to cases in which the subquery is dependent from its nearest non-algebraic operator, but the dependency is specifically constrained. The dependency concerns only an expression that can be entirely evaluated on a small collection. The rewriting rule in [3] is relevant regardless of whether the values of a small collection are available or unavailable during the compilation time. The values can be also changed after the compilation. If the dependency concerns a name that is typed by enumeration then a simpler rewriting rule may be used (see [2]). This rule is based on a conditional statement using all enumerators that have to be known during the compilation time.

In general, the dependency of a subquery from its nearest non-algebraic operator can concern two or more expressions dependent on small collections. The number of evaluations of such a subquery called “compound weakly dependent subquery” can be limited to the product of the sizes of these collections. The sizes of these collections should be significantly smaller in comparison to the collection size returned by the left subquery of the non-algebraic operator on which the subquery depends on. Comparing the sizes of collections enables the optimizer to check whether rewriting the query would guarantee better evaluation time. It implies introducing an efficient query evaluations cost model.

The rest of the paper is organized as follows. Section 2 describes the general idea of the compound weakly dependent subqueries method. Section 3 presents the corresponding algorithm that we have developed for the system ODRA [1]. Section 4 concludes.

## 2 The Optimization Method

To present SBA examples, we assume the class diagram (schema) presented in Fig.1. The classes *Student*, *Project*, *Emp* and *Dept* model projects implemented by students and supervised by employees working in departments. Names of classes (attributes, links, etc.) are followed by cardinality numbers (cardinality [1..1] is dropped).

Attributes *sex* of *Person* and *job* of *Emp* are of enumerated types. The first one takes values (“male”, “female”), the second one takes values (“analyst”, “programmer”, “tester”).

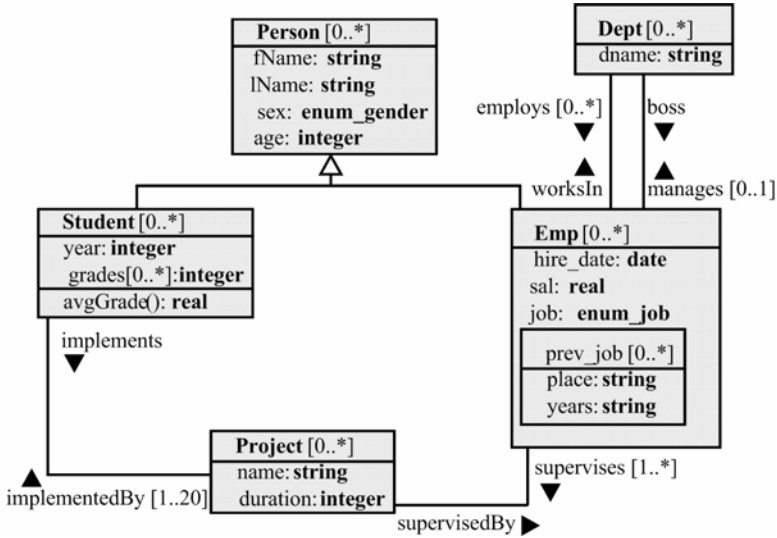


Fig. 1. A schema of an example database

### 2.1 Static Analysis of Queries

In this subsection we briefly present the mechanism of *static analysis* [10] used in our optimization method. It is a compile-time mechanism that performs static type checking ([4], [7]) on the basis of abstract syntax trees (ASTs) of queries. The static analysis uses special data structures: a metabase, a static environment stack *S\_ENVS* and a static query result stack *S\_QRES*. These structures are compile-time equivalents of run-time structures: an object store, an environment stack *ENVS* and a query result stack *QRES*, correspondingly. *S\_ENVS* models bindings (in particular opening new scopes and binding names) that are performed on *ENVS*. The process of accumulating intermediate and final query result on *QRES* is modeled by *S\_QRES*.

The main component of the metabase is a *schema graph* that is generated from a database schema. It contains nodes representing database entities (objects, attributes, classes, links, etc.) and interfaces of methods. The edges represent relationships between nodes. The graph nodes are identified by internal identifiers that are processed on static stacks. In our model the metabase stores also some statistical data. A node of the schema graph is associated with the estimated number of objects in the collection that is represented by the node. We designate it by  $NE(Entity)$ , where *Entity* is a unique node identifier. For clarity, *Entity* will be represented by an object name instead of a node identifier (in general, however, this assumption is not adequate, as node names need not be unique). For instance,  $NE(Emp) = 1000$ ,  $NE(Dept) = 10$ ,  $NE(Student)=2000$ ,  $NE(Project)=100$ ,  $NE(Person) = 3000$ .

## 2.2 General Idea of the Optimization Method Involving Compound Weakly Dependent Subqueries

The following example in SBQL illustrates the general idea of our method. The query gets employees having salary greater than the average salary calculated for employees working in their departments and having the same job. For the query below we determine binding levels for all the names and the numbers of stack sections opened by non-algebraic operators.

$$\begin{array}{l}
 \text{Emp as } e \text{ where } e . \text{sal} > \\
 1 \quad 2 \quad 2 \quad 3 \quad 3 \\
 \quad \text{avg}(e . \text{worksIn} . \text{Dept} . \text{employs} . (\text{Emp where job}=e . \text{job}) . \text{sal}) \quad (1) \\
 \quad 2 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 4 \quad 4 \quad 2 \quad 5 \quad 5 \quad 3 \quad 3
 \end{array}$$

Consider the following subquery of query (1)

$$\text{avg}(e.\text{worksIn}.\text{Dept}.\text{employs}.(Emp \text{ where } job=e.job).\text{sal}) \quad (2)$$

The subquery (2) contains two names  $e$  that are bound in the 2<sup>nd</sup> stack section opened by the first *where* operator : in expression  $e.\text{worksIn}.\text{Dept}$  and in expression  $e.job$ . Hence the method of independent subqueries [9] cannot be applied. The method of queries involving large and small collections [3] cannot be applied too because it resolves dependency of subqueries concerning only one small collection. However the subquery (2) can be evaluated only 30 times instead of 1000 times. Therefore we have to develop a more general rewriting rule. After optimizing (1) it should take the following form:

$$\begin{array}{l}
 (((\text{Dept as } n1 \text{ join bag}(\text{"analyst"}, \text{"programmer"}, \text{"tester"}) \text{ as } n2) \\
 \quad \text{join avg}(n1.\text{employs}.(Emp \text{ where } job=n2).\text{sal}) \text{ as } n3) \text{ group as } aux). \quad (3) \\
 (\text{Emp as } e \text{ where } e.\text{sal} > (aux \text{ where } e.\text{worksIn}.\text{Dept}=n1 \text{ and } e.job=n2).n3)
 \end{array}$$

Names  $n1$ ,  $n2$ ,  $n3$  and  $aux$  are automatically chosen by the optimizer. In the two first lines of (3) before the last *dot* the query returns a bag named  $aux$  consisting of 30 structures. Each structure has three fields: an identifier (of a  $Dept$  object) named  $n1$ , a name of the job named  $n2$  and the average salary (named  $n3$ ) calculated for employees having such job and working in this  $Dept$ . The last *dot* in the second line puts on top of ENVs a binder  $aux$  containing these structures. It is then used to calculate the query in the 3<sup>rd</sup> line. In this way average salaries are calculated three times for each department and they are used in the final query, as required.

Detecting subqueries such as (2) consists in verifying in which ENVs sections the names occurring in a subquery are to be bound. The binding levels for names are compared to the scope numbers of non-algebraic operators. We take into consideration only subqueries (referred to as "compound weakly dependent subqueries") of a given query that depend from their direct non-algebraic operator only on expressions returning small collections. If the number being the product of sizes of these collections is quite a lot smaller than the collection size returned by the left subquery of this non-algebraic operator, then we decide to rewrite such a query. Comparing the sizes of collections is necessary to check whether rewriting the query would ensure better performance. The query (1) involves two subqueries connected by the *where* operator. The left subquery  $Emp \text{ as } e$  returns 1000 elements (according

to the statistical data) and the right subquery (2) depends on the *where* operator on two expressions returning 10 and 3 elements, correspondingly. Hence it makes sense to rewrite (1) to the form (3). In this way the number of evaluations of the query (2) has been reduced to  $NE(Dept) * sizeof(enum\_job) = 30$ .

An essential difficulty of the algorithm consists in detecting specific parts of a compound weakly dependent subquery like *e.worksIn.Dept* and *e.job* for (2). We have chosen these parts because they contain the name *e* that is bound in the scope opened by the *where* operator. Besides these parts depend on small collections only. Other names in a subquery like (2) cannot be bound in the scope of its left-side direct non-algebraic operator. When the expression is typed by an enumeration (as in case of *e.job*) we use an explicit bag containing values of the enumerated type.

To limit the number of evaluations of a subquery like (2) to the number of collection elements returned by the subquery (4)

$$Dept \text{ as } n1 \text{ join bag}(\text{"analyst"}, \text{"programmer"}, \text{"tester"}) \text{ as } n2 \tag{4}$$

we factor a subquery like (5)

$$(Dept \text{ as } n1 \text{ join bag}(\text{"analyst"}, \text{"programmer"}, \text{"tester"}) \text{ as } n2) \tag{5}$$

$$\text{ join avg}(n1.employs.(Emp \text{ where } job=n2).sal) \text{ as } n3$$

out of the first *where* operator in (1). Query (5) is named with the auxiliary name *aux*. Then this name, as well as previously introduced names, are used to rewrite the subquery (2) to the form:

$$(aux \text{ where } e.worksIn.Dept=n1 \text{ and } e.job=n2).n3 \tag{6}$$

The query (6) returns for each employee the average salary calculated for employees working in his/her department and having the same job.

The method is experimentally tested within the ODB system [1]. In the case of the query (1), the gain for a collection of 1000 employee objects is 10 times faster execution and the gain for 10000 employee objects is 118 times faster execution.

### 2.3 More General Case

The presented optimization method makes it possible to rewrite arbitrary compound weakly dependent subqueries. After rewriting such a subquery can be factored out of any non-algebraic operator. Besides, in general a query can contain several weakly dependent subqueries. Recursive application of our method enables rewriting all weakly dependent subqueries. At first, subqueries which are under the scope of the most nested non-algebraic operators are rewritten. The following example illustrates the generality of our method.

$$\exists Student \text{ as } s (\text{count}(s.implements.Project.supervisedBy.(Emp \text{ where } age > 2 \ 1 \ 2 \ 3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 4 \ 4) \text{ avg}(worksIn.Dept.employs.(Emp \text{ where } sex=s.sex).age))) > 0) \tag{7}$$

$$4 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5 \ 6 \ 6 \ 2 \ 7 \ 7 \ 5 \ 5$$

The above query returns *true* if at least one student implemented a project that was supervised by an employee satisfying the following criterion: the age of this employee

must be greater than the average age calculated for employees working in his/her department and having the same sex as the given student.

Note that entire right-hand subquery of the quantifier is weakly dependent from it. Both expressions (*s.implements.Project* and *s.sex*) contain the name *s* that is bound in the 2<sup>nd</sup> stack section opened by the quantifier and depend on small collections. However the right-hand subquery of the quantifier also contains the subquery:

$$\mathbf{avg}(\mathit{worksIn.Dept.employs}(\mathit{Emp \ where \ sex=s.sex}.age)) \quad (8)$$

that is weakly dependent from the first *where* operator. The name *worksIn* in (8) is bound in the scope opened by this operator and denotes a pointer link to an element of a small collection of *Dept* objects. After rewriting the nested weakly dependent query (8) the query (7) takes the following form:

$$\exists \mathit{Student \ as \ s} (\mathbf{count}(\mathit{s.implements.Project.supervisedBy}(\mathbf{((Dept \ as \ n1 \ \mathbf{join} \ \mathbf{avg}(n1.employs}(\mathit{Emp \ where \ sex=s.sex}.age) \ \mathbf{as} \ n2) \ \mathbf{group \ as} \ \mathit{aux1}).}(\mathit{Emp \ where \ age}>(\mathit{aux1 \ where \ worksIn.Dept=n1}.n2))))>0) \quad (9)$$

Now the query (9) contains only one weakly dependent subquery: it is the entire right subquery of the quantifier. Applying the transformation to (9) we obtain the following query:

$$\begin{aligned} &(((\mathit{bag}(\text{"male"}, \text{"female"}) \ \mathbf{as} \ n3 \ \mathbf{join} \ \mathit{Project \ as} \ n4) \ \mathbf{join} \\ & \quad (\mathbf{count}(n4.supervisedBy}(\mathbf{((Dept \ as} \ n1 \ \mathbf{join} \ \mathbf{avg}(n1.employs}(\mathit{Emp \ where} \\ & \quad \quad \mathit{sex=n3}.age) \ \mathbf{as} \ n2) \ \mathbf{group \ as} \ \mathit{aux1}).}(\mathit{Emp \ where \ age}> \\ & \quad (\mathit{aux1 \ where \ worksIn.Dept=n1}.n2))))>0) \ \mathbf{as} \ n5) \ \mathbf{group \ as} \ \mathit{aux2}).(\mathbf{forsome} \\ & \quad (\mathit{Student \ as} \ s) (\mathit{aux2 \ where \ s.implements.Project=n4 \ \mathbf{and} \ s.sex=n3}.n5)) \end{aligned} \quad (10)$$

The form (10) terminates the optimization action – no further optimization by means of our method is possible.

## 2.4 General Rewriting Rule

The general rewriting rule for queries involving compound weakly dependent subqueries can be formulated as follows. Let  $q_1 \theta q_2$  be a query connecting two subqueries by a non-algebraic operator  $\theta$ . Let  $q_2$  has the form:

$q_2 = \alpha_1 \circ \mathit{cwds}(\beta_1(C_1), \beta_2(C_2), \dots, \beta_k(C_k)) \circ \alpha_2$ ;  $k \geq 1$ ,  $\alpha_1$  and  $\alpha_2$  are some parts of  $q_2$  (maybe empty),  $\circ$  is a concatenation of strings,  $\mathit{cwds}(\beta_1(C_1), \beta_2(C_2), \dots, \beta_k(C_k))$  is a compound weakly dependent subquery where each part  $\beta_i$  ( $i=1..k$ ) depends on  $\theta$  only and contains a name  $C_i$  that is bound to an element of a small collection. Each  $\beta_i(C_i)$  must be of the same type as the type of the collection of  $C_i$  elements. Then the query:

$$q_1 \theta \alpha_1 \circ \mathit{cwds}(\beta_1(C_1), \beta_2(C_2), \dots, \beta_k(C_k)) \circ \alpha_2 \quad (11)$$

can be rewritten to:

$$\begin{aligned} &(((C_1 \ \mathbf{as} \ sc_1 \ \mathbf{join} \ C_2 \ \mathbf{as} \ sc_2 \ \mathbf{join} \ \dots \ \mathbf{join} \ C_k \ \mathbf{as} \ sc_k) \\ & \quad \mathbf{join} \ \mathit{cwds}(sc_1, sc_2, \dots, sc_k) \ \mathbf{as} \ \mathit{aux1}) \ \mathbf{group \ as} \ \mathit{aux2}). \\ & \quad (q_1 \theta \alpha_1 \circ ((\mathit{aux2 \ where} \ sc_1 = \beta_1(C_1) \ \mathbf{and} \ sc_2 = \beta_2(C_2) \ \mathbf{and} \ \dots \\ & \quad \quad \mathbf{and} \ sc_k = \beta_k(C_k)).\mathit{aux1}) \circ \alpha_2) \end{aligned} \quad (12)$$

The general idea consists in limiting the number of processings of a compound weakly dependent subquery  $cwds(\beta_1(C_1), \beta_2(C_2), \dots, \beta_k(C_k))$  to the number that is the product of the sizes of collections  $C_i$  ( $i=1..k$ ) occurring in  $\beta_i$ . It is aimed by introducing an additional query with the *join* operator that is independent of  $\theta$ . The entire result of this query is named *aux2*. It is a bag of structures  $struct\{(sc_1(c_1), sc_2(c_2), \dots, sc_k(c_k)), aux2(cw)\}$ , where  $c_i$  ( $i=1..k$ ) is an element of a bag returned by the name  $C_i$  and  $cw$  is an element returned by  $cwds(sc_1, sc_2, \dots, sc_k)$ . The bag is then used (after the *dot*) to construct the query (*aux2 where  $sc_1 = \beta_1(C_1)$  and  $sc_2 = \beta_2(C_2)$  and...and  $sc_k = \beta_k(C_k)$* ).*aux1*. It replaces the compound weakly dependent  $cwds(\beta_1(C_1), \beta_2(C_2), \dots, \beta_k(C_k))$  of (11). If some expression  $\beta_i(C_i)$  is typed by an enumeration then instead of the collection name  $C_i$  the bag that consists of all the values of an enumerated type is used in (12).

### 3 The Rewriting Algorithm

The rewriting based on the rules (11) and (12) is accomplished by five recursive procedures. For the paper space limit we present only their signatures:

- *optimizeQuery*( $q:ASTtype$ ) – it applies the *queriesInvolvingCWDSMethod* procedure to AST node  $q$  as long as  $q$  contains subqueries depending on their non-algebraic operators only on expressions returning small collections.
- *queriesInvolvingCWDSMethod*( $q:ASTtype$ ) – it recursively traverses AST starting from node  $q$  and applies the *applyQueriesInvolvingCWDSMethod* procedure. If the procedure meets a non-algebraic operator then its right and left queries are visited by the same procedure. At first compound weakly dependent subqueries which are under the scope of the most nested non-algebraic operators will be rewritten.
- *applyQueriesInvolvingCWDSMethod*( $\theta:ASTtype$ ) – it transforms according to rewriting rule (12) all right-hand subqueries of non-algebraic operator  $\theta$  that depend on it only on expressions returning collections for which the number being the product of their sizes is small than the collection size returned by the left-hand subquery of the  $\theta$  operator.
- *findCWDS*( $\theta:ASTtype, q:ASTtype$ ): ( $ASTtype, ASTtype$ ) – it applies the *getCWDS* function as long as the function returns a subquery of  $q$  (maybe the whole  $q$ ) that is compound weakly dependent from  $\theta$ .
- *getCWDS*( $\theta:ASTtype, q:ASTtype$ ): ( $ASTtype, ASTtype$ ) – it detects parts of query  $q$  that are dependent from  $\theta$  operator on single names. Other names in the query cannot be in the scope of  $\theta$ . If the dependency concerns expressions returning small collections or typed by enumerations then the function returns the query  $q$  and its dependent parts.

### 4 Conclusions and Future Work

We have presented a query optimization method which was aimed at minimizing the number of evaluations of compound weakly dependent subqueries. Our rewriting rule is very general, it works for any non-algebraic operator and for any data model (assuming that its semantics would be expressed in terms of SBA). The rule makes



also no assumptions concerning what the compound weakly dependent subquery returns: it may return a reference to an object, a single value, a structure, a collection of references, a collection of values, a collection of structures, etc. Finally the rule makes rewrites for arbitrarily complex nested subqueries regardless of their left and right contexts.

The algorithm applied repeatedly detects and resolves all the possible compound weakly dependent subqueries in a query. Besides some subquery can be dependent from several non-algebraic operators, hence in each iteration of the algorithm the subquery is transformed and factored out of a next one. The prototype rewriting algorithm has been implemented by us in the ODRA system. We have to perform many experimental tests to confirm the efficiency of this algorithm. We also plan to implement another optimization variant that does not depend on a cost model.

## References

1. Adamus, R., et al.: Overview of the Project ODRA. In: Proc. 1st ICOODB Conf., pp. 179–197 (2008) ISBN 078-7399-412-9
2. Bleja, M., Kowalski, T., Adamus, R., Subieta, K.: Optimization of Object-Oriented Queries Involving Weakly Dependent Subqueries. In: Proc. 2nd ICOODB Conf., Zurich, Switzerland, pp. 77–94 ISBN 978-3-909386-95-6
3. Bleja, M., Stencel, K., Subieta, K.: Optimization of Object-Oriented Queries Addressing Large and Small Collections. In: Proc. of the International Multiconference on Computer Science and Information Technology, Mrągowo, Poland, pp. 643–650 ISBN 978-83-60810-22-4, ISSN 1896-7094
4. Hryniów, R., et al.: Types and Type Checking in Stack-Based Query Languages. Institute of Computer Science PAS Report 984, Warszawa (March 2005), <http://www.si.pjwstk.edu.pl/publications/en/publications-2005.html> ISSN 0138-0648
5. Ioannidis, Y.E.: Y.E., Query Optimization. *Computing Surveys* 28(1), 121–123 (1996)
6. Jarke, M., Koch, J.: Query Optimization in Database Systems. *ACM Computing Surveys* 16(2), 111–152 (1984)
7. Lentner, M., Stencel, K., Subieta, K.: Semi-strong Static Type Checking of Object-Oriented Query Languages. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 399–408. Springer, Heidelberg (2006)
8. Official Microsoft LINQ Project, <http://msdn.microsoft.com/en-us/netframework/aa904594.aspx>
9. Płodzień, J., Kraken, A.: Object Query Optimization through Detecting Independent Subqueries. *Information Systems* 25(8), 467–490 (2000)
10. Płodzień, J., Subieta, K.: Static Analysis of Queries as a Tool for Static Optimization. In: Proc. IDEAS Conf., pp. 117–122. IEEE Computer Society, Los Alamitos (2001)
11. Subieta, K., Beeri, C., Matthes, F., Schmidt, J.W.: A Stack Based Approach to Query Languages. In: Proc. of 2nd Intl. East-West Database Workshop, Springer Workshop in Computing, Klagenfurt, Austria, September 1994, pp. 159–180 (1994)
12. Subieta, K.: Stack-Based Approach (SBA) and Stack-Based Query Language, SBQL (2008), <http://www.sbql.pl>
13. Subieta, K.: Stack-based Query Language. In: *Encyclopedia of Database Systems 2009*, pp. 2771–2772. Springer, US (2009) ISBN 978-0-387-35544-3, 978-0-387-39940-9

# A Utilization of Schema Constraints to Transform Predicates in XPath Query

Dung Xuan Thi Le<sup>1</sup>, Stéphane Bressan<sup>2</sup>, Eric Pardede<sup>1</sup>,  
David Taniar<sup>3</sup>, and Wenny Rahayu<sup>1</sup>

<sup>1</sup> La Trobe University, Australia  
{dx11le@students., w.rahayu@, e.pardede@}latrobe.edu.au

<sup>2</sup> National University of Singapore  
steph@nus.edu.sg

<sup>3</sup> Monash University, Australia  
David.Taniar@infotech.monash.edu.au

**Abstract.** A predicate in an XPath query expresses a condition to be fulfilled in addition to the structural constraint imposed by the path itself. The condition is a Boolean expression. It may involve comparisons between elements and values, path expressions denoting elements to be compared as well as further path expressions. In this paper, we are concerned with the semantic transformation of such predicates in the presence of a schema for the XML data. The goal of this semantic transformation is to eliminate the predicates from the XPath query if possible in order to avoid the early, unnecessary query processing. Otherwise, if predicates are retained, we show how they can be semantically transformed to boost efficiency and reduce resource utilization. An algorithm is proposed to determine whether predicates should be eliminated or retained. We implement the proposed transformations and empirically evaluate their efficiency and effectiveness as semantic query optimization devices.

**Keywords:** Semantic Transformation, XML Query Processing, XPath.

## 1 Introduction and Motivation

A predicate in an XPath query expresses a condition to be fulfilled in addition to the structural constraint imposed by the path itself. The condition is a Boolean expression. It may involve comparisons between elements and values, path expressions denoting elements to be compared as well as further path expressions. Let us consider the following XPath query:

```
//contract[supervisor = /company//perm/@id and age <=52]
```

Assuming self-explanatory XML data with a clear intended meaning of the element names and structure, this XPath query lists “*all contract details where each contract has a supervisor who is also a permanent employee and contract age is not more than 52*”. The elementary predicate involves a comparison of an element ‘supervisor’ and an absolute path ‘/company//perm/@id’ as well as a comparison of

an element 'age' with a constant value range 52. The multiple conditions are connected with a conjunction 'AND'.

The XPath query in the example above can be transformed, under the knowledge of different constraints, for example *structure*, *cardinality*, *foreign key and inclusion*, in the XML schema. Semantic query transformation precisely takes advantage of such structural and explicit constraints defined in the XML schema to transform the query.

One of the benefits of semantic transformation is to detect and remove any redundancy in the query that may impact upon performance. Assume that the constraints in the schema allow us to eliminate the predicate [supervisor = 'company//perm/@id' and age < =52] from the XPath query, which now is simplified to '//contract'. The actual performance improvement between this query '//contract' and the original query //contract[supervisor = /company//perm/@id and age <=52] is demonstrated later in this paper.

In the above example, the predicate has been completely removed. This is known as predicate elimination, which is a common and practical semantic query optimization for relational databases. However, in XML an query, the structure of the data must be taken into account: the semantic transformation involves decisions about both the tree pattern structure and the values of the data.

Our motivation is to leverage the schema of XML data to propose a semantic transformation typology to deal with complex Boolean predicates in the XPath query. The fragment of XPath query studied in this work includes ("?", "/", "[]" or "\*"). The semantic transformation takes into consideration of that the two XPath queries are equivalent if and only if they yield the same result set.

The semantic transformation aims to improve the performance of query evaluation, which can be achieved by either removing the predicate if verification of constraints used by the predicate is qualified or by eliminating some redundancies in the predicate. We then evaluate the effectiveness and efficiency of the resulting semantic transformation framework for XPath queries using five XML data sets of different sizes and associated schema. We adopt a representative XML-enabled (we use the vendor extension) off-the-shelf commercial relational database management system to perform our experimentation.

**Roadmap.** Following the motivation and introduction, in Section 2, we provide an overview of related works. Our semantic transformation including a status determination function and its algorithm and rules is presented in Section 3. We discuss the implementation setting and analyze the result in Section 4. We conclude this paper in Section 5.

## 2 Related Work

We are clearly concerned with semantic query optimization [1, 5, 10, 12] and the satisfiability problem [2, 3, 11, 7, 8]. The authors of [9] consider a predicate that covers mostly simple elements with values. The 'Raindrop' project [13, 14] demonstrated the concept of rewriting XQuery, but it overlooked semantic rewriting, as binding variables in XQuery must be handled properly prior to detecting the path for writing. This contributes to the delay of the rewriting time. As far as predicate rewriting

is concerned, this paper proposes conditional elements with a single value, while the conjunctive or disjunctive merging of multiple conditions is not addressed. Previous work, such as [2, 17], is limited to *inclusion*, *exclusion* and *enumeration* of XML schema constraints. We now explore opportunities to work on a complex type of predicate, such as merging conditions.

In relation to a schema-based approach, the schema information is also used to study satisfiability problems. The work in [6, 7, 8, 17] uses the information in the schema to eliminate the wildcard in the XPath query. These approaches focused on the replacement of the wildcard by a specific nodes test and by eliminating the recursive axis. This can be done when the path can be easily mapped to a schema path, and the set of schema paths is computed through the XSchema-XPath evaluator.

Although these proposed techniques do not directly discuss XML query semantic transformation, they have a direction similar to that of our work. The difference between our technique and the existing work is that while we pre-process the XML schema and derive all available schema paths, the existing work [2, 6, 7, 8, 16] processed schema paths only when a user XPath query is given for constraint checking. This means that there is potentially a long delay, as the schema needs to be processed as a whole each time there is a new query requirement. Our technique may initially take longer; however, the schema pre-processing is done only once, and hence, the transformation process for any XML query will be much quicker.

The introduction of schema constraints utilization [4, 15] provides a set of rules for transforming the *structural* types of queries to the equivalent ones with the aim of achieving a better performance. ‘Structural’ query type means a query tree such as an XPath query that allows no predicate; hence, no predicate was addressed in this work [4].

The paper first explored a typology of structural transformations that utilize a structural constraint such as *unique path* [4]. This constraint is commonly seen and expressed in any given XML schema. Our work leverages the task of transformation so that semantic transformation can be achieved with highly effective results. We focus on the semantic transformation work for transforming predicates in an XPath query.

### 3 Semantic Transformations

Semantic transformation for XPath predicates requires multiple constraints including structure and various leaf-node element constraints. It is imperative that the awarding of status (e.g. ‘full-qualifier’ or ‘partial-qualified’) is accurate.

#### 3.1 Status Determination for a Predicate

We propose a function, see Algorithm 1 for full details, to determine the status of a predicate in XPath before it is transformed.

$$S \rightarrow (Q, C) \\ \varepsilon(1R) \xrightarrow{\downarrow} R_S$$

The function  $\mathcal{E}$ , accepts the input  $lR$  and transforms it to  $R_S$  by using information  $S(Q, C)$ .  $R$  is a predicate, and  $R = [r_1. (\perp) r_2 (\perp) \dots (\perp)r_n]$ , where  $r_1, r_2, r_3, \dots, r_n$  is a series of restriction and  $\perp$  is the logical operator AND/OR;  $l$  is an outer-focus<sup>1</sup> of a predicate;  $Q$  is a list of unique paths, each of which is a structural constraint;  $C$  is a list of schema constraints that contains a set of elements and their associated constraints. Both  $Q$  and  $C$  are derived from schema  $S$ . They are the global data structured lists, which are constructed by a function called `pre_processing_schema`[4]. This function processes all constraints/semantics defined in the given schemas and stores them in  $Q$  and  $C$ .

The function  $\mathcal{E}$  returns a status such as conflict-qualifier, full-qualifier or partial-qualifier.

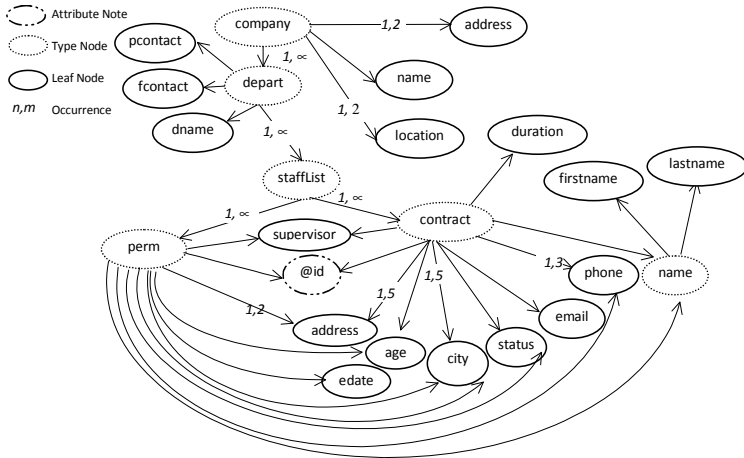


Fig. 1. An outline of XML Schema Fragment for a *Company* Structure in a Graphical View

**Example 1.** Below, we demonstrate an XPath query (refer to Figure 1)  $p = //contract[supervisor = \text{'}/company//perm/@id']/name$ . In this example, we demonstrate the predicate  $R = [supervisor = \text{'}/company//perm/@id']$  where its status needs to be identified as a partial-qualifier or a full-qualifier predicate.

As shown, predicate  $\mathcal{R}$  contains a restriction  $r$  which is a comparison of the value of the "supervisor" element with an absolute path  $\text{'}/company//perm/@id'$ . By applying the function  $\mathcal{E}$ , it first verifies if  $\text{'}/contract/supervisor'$  is a subset of a unique path in list  $Q$ . Second, the function also verifies whether  $\text{'supervisor'}$  is referenced to 'permkey' in constraint list  $C$ . As the result, function  $\mathcal{E}$  finds that  $\text{'}/contract/supervisor'$  is a subset of the unique path  $\text{'company/depart/staffList/contract/supervisor'}$  in list  $Q$  and  $\text{'supervisor'}$  refers to 'permkey', an ID of a permanent employee. The algorithm also evaluates  $\text{'}/company//perm/@id'$  to ensure the correctness of the path expression. As a result, it awards the restriction  $\text{'FQ'}$  status, hence  $R$  is a 'full-qualifier'.

<sup>1</sup> An outer-focus is an element which externally associates with the predicate.

**Algorithm 1. Status Determination for a Predicate**

```

01: INPUT:  $\perp R$ ,  $C$ ,  $Q$ 
02: OUTPUT:  $sR$ 
03: BEGIN
04: Let  $CF$  be 'conflict',  $FQ$  be 'full-qualifier',  $PQ$  be 'partial-qualifier',
     $\lambda$  be a comparison operator ( $<$ ,  $>$ ,  $=$ ,  $\geq$ ),  $q$  be a unique path in  $Q$  list,  $tempv$  is a temp variable.
05: For each  $lr$  not yet being evaluated in  $\perp R$ 
06: IF  $semantic\_path\_match(lr) \not\subseteq q$  exists in  $Q$  THEN
07:    $lr := 'CF'$ 
08: ELSE
09:   CASE  $lr$  contains no  $v$  ; set  $lr := 'FQ'$ 
10:   CASE  $lr$  contain  $v$ 
11:     Match  $v$  of  $e$  in  $lr$  to  $v$  of  $e$  in  $C$ 
12:     IF  $v$  of  $e$  in  $lr$  do not match  $\forall v$  of  $e$  in  $C$  THEN set  $r := 'CF'$ 
13:     ELSE IF  $v$  of  $e$  in  $lr$  matches  $\forall v$  of same  $e$  in  $C$  THEN set  $r := 'FQ'$ 
14:     ELSE IF  $v$  of  $e$  in  $lr$  matches  $\exists v$  of same  $e$  in  $C$  THEN  $lr := 'PQ'$ ;  $lr := 'checked'$ ;
         $tempv := v$ 
15:     For  $lr$  not yet processed in  $\perp R$ 
16:       IF found  $lr$  has the same partial-path as  $tempv$  has THEN
17:          $tempv := v \cup tempv$ ;  $lr := 'checked'$ 
18:         IF  $tempv$  of  $e$  matches  $\forall v$  of same  $e$  in  $C$  AND exist  $\lambda = 'OR'$  THEN
19:           Replace all 'checked' with 'FQ'
20:           ELSE IF  $tempv$  of  $e$  matches  $\exists v$  of same  $e$  in  $C$  AND (exist  $\lambda = 'OR'$  or no  $\lambda$ ) THEN
21:             Replace all 'checked' with 'PQ'
22:             ELSE Replace all 'checked' with 'CF'
23:         CASE  $lr$  contain  $v$  AND  $v$  is a path expression
24:           IF ( $v \subseteq q$  in  $Q$ ) AND  $v$  of  $e$  in  $lr$  matches  $\forall v$  of same  $e$  in  $C$  THEN
25:              $r := 'FQ'$ 
26:           ELSE  $lr := 'CF'$ 
27: IF exist only 'PQ' AND exist  $\perp$  or not exist  $\perp$  in  $R$  THEN  $sR := 'PQ'$ 
28: ELSE IF exist only 'FQ' AND exist  $\perp$  or not exist  $\perp$  in  $R$  THEN  $sR := 'FQ'$ 
29: ELSE IF exist 'PQ', 'FQ', 'CF' AND exist  $\perp$  in  $R$  THEN
30:   DO WHILE exist next_status in  $R$ 
31:     result_status := 'FQ' IF status is 'FQ' AND next_status is 'CF' AND  $\perp$  is 'OR'
32:     result_status := 'PQ' IF status is 'PQ' AND next_status is 'CF' AND  $\perp$  is 'OR'
33:     result_status := 'CQ' IF status is 'FQ' or 'PQ' AND next_status is 'CF' AND  $\perp$  is 'AND'
34:     status is = result_status; next_status is the status after next_status
35:   LOOP
36:  $sR := result\_status$ 

```

**3.2 Semantic Transformation Rule After Status Determination**

A predicate with 'FQ' status may be retained in the XPath query if it does not satisfy certain rules (presented below). A predicate with 'PQ' status cannot be removed; however, we can semantically transform it to another expression [4] where possible, to boost the query processing efficiency. We propose a rule to finalize the predicate, based on the status below.

**Rule 1. Semantic transformation for predicate.** Let  $R$  be a predicate in XPath query  $\mathcal{P}$ ,

1. **Predicate elimination:** If a predicate  $R$  is awarded FQ status and the comparison element in each restriction also exists in the schema constraint

list C. and the comparison element has a minimal occurrence value of at least 1, then R is removed from P.

2. **Predicate transformation:** If R is awarded either PQ or FQ status where a comparison element of each restriction in R has a minimal cardinality at least 0, R cannot be removed but can be further transformed [4]

## 4 Implementation Overview and Empirical Evaluation

This section describes the implementation and empirical evaluation.

### 4.1 Implementation

We describe the implementation in terms of *hardware* and *software*: (1) the *hardware* includes a machine that has a configuration of AMD Athlon 64 3200+, 2300 MHz and 3.0 GB of RAM; and (2) the *software* includes Windows XP Professional OS and Java VM 1.6. We use a commercial relational database system that is XML-enabled and use their provided database connection driver to connect to our algorithm modules. We use five synthetic datasets (compliant with the schema shown in Figure 1) of varying sizes: 20, 40, 60, 80 and 100 megabytes. We run a series of queries for each data set.

Our actual workload is composed of XPath queries. For each category, we select one or two queries to illustrate the result. We show the original query and the result of query rewriting after our semantic transformation algorithms transform the query.

```
Original Query 1 //depart/staffList[perm/phone or perm/age < 53]
Transformed Query 1 //depart/staffList
Original Query 2 //contract[supervisor = '/company//perm/@id']/name
Transformed Query 2 //contract/name
Original Query 3 company/depart[//perm/age = 35]/name
Transformed Query 3 company/depart[staffList/perm/age = 35]/name')
Original Query 4 company/depart/staffList/contract[duration = 2 and age >
18]/name
Transformed Query 4 /company/depart/staffList/contract/name
```

As mentioned in the previous section, the schema has been pre-processed prior to the queries being issued. The first four XPath proved valid, therefore, there are semantic XPath queries produced. The fifth query is an invalid type, which results in no semantic query being produced and hence, there is no need to access the database.

### 4.2 Empirical Evaluation

The results of the queries above are graphed in Figure 2. Each graph shows the results of two queries: the first result is the time that is taken to access the database by the original query and the second result is the total of the transformation time and accessing the database time by its associated semantic XPath.

In the **comparison of partial Path with constant & connective (Query 1)** we demonstrate the multiple conditions joined by a ‘or’ connective. The predicate is

completely eliminated from the XPath query. The semantic query performance is slightly improved between 7 and 10%, compared to the performance of the original XPath. We also note that the gain increases as data size increases.

In **Query 4** we demonstrate the connective for this XPath query is 'and'. The semantic query performance is not significantly improved but does not indicate a downgrade of performance.

In the **comparison of partial path with absolute path (Query 2)**, we demonstrate the predicate that has a restricted element 'supervisor' of contract employees, who must be a permanent employee. The verification confirms the predicate has full-qualifier status. The predicate has been completely removed due to the minimal cardinality of 'supervisor' being 1. The performance is improved by between 45 and 79%, compared to the performance of the original XPath. We also note that the gain increases significantly as data size increases.

In the **comparison of partial path with constant & no connective (Query 3)**, unlike query 1 and 2, we demonstrate the descendant-or-self '/' hierarchical relationship with no connectives in the predicate. In the predicate, only employees who are 35 years old are needed. This means that a value of 35 is only part of the restricted values.

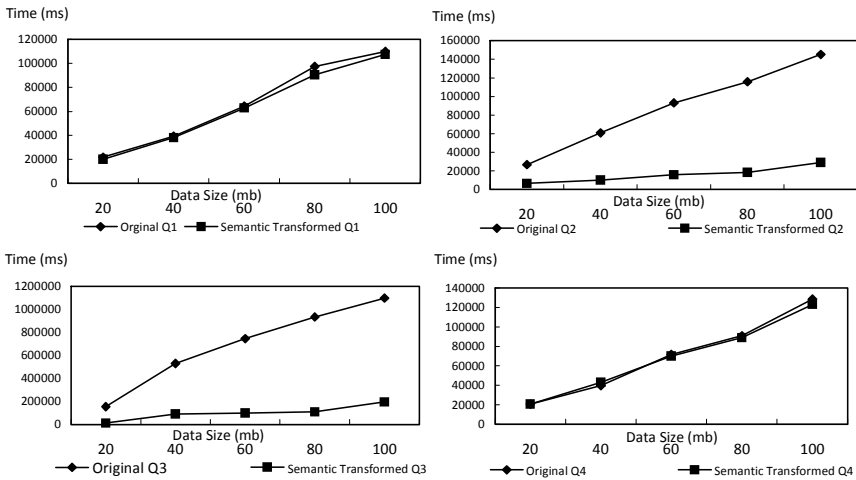


Fig. 2. Query Performance

The semantic query performance is significantly improved by between 40 and 78% (the gain significantly increases as the size increases), compared to the performance of the original XPath.

## 5 Conclusion and Future Work

In this paper, we propose a semantic query transformation for predicates. We focus on predicates that not only have multiple conditions but also are such that each condition



has path and values combined. We show empirically that the proposed transformations provide effective and promising opportunities to improve the performance of XPath query evaluation. Our objective is to ultimately offer a comprehensive semantic query optimization framework for XML databases. This is necessary since vendors are proposing versatile and opaque solutions that do not allow intrusive or even fine-grain optimization techniques. Our vision is one of the loosely-coupled semantic optimizations that respect vendors' autonomy while bringing significant benefits to the user.

## References

1. Charkravarthy, U.S., Grant, J., Minker, J.: Logic-Based Approach to Semantic Query Optimization. *ACM Transactions on Database Systems* 15(2), 162–207 (1990)
2. Chan, Y., Fan, W., Zeng, Y.: Taming XPath Queries by Minimizing Wildcard Steps. In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pp. 156–167 (2004)
3. Che, D., Aberer, K., Özsu, M.T.: Query optimization in XML structured-document databases. *The VLDB Journal The International Journal on Very Large Data Bases* 15(3), 263–289 (2006)
4. Le, D., Bressan, S., Taniar, D., Rahayu, W.: Semantic XPath Query Transformation: Opportunities and Performance. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) *DASFAA 2007*. LNCS, vol. 4443, pp. 994–1000. Springer, Heidelberg (2007)
5. Le, D., Pardede, E.: On Using Semantic Transformation Algorithms for XML Safe Update. In: *8th International Conference on Information Systems Technology and its Applications, ISTA 2009*, pp. 367–378 (2009)
6. Hammer, M., Jdondik, S.B.: Knowledge-based processing. In: *Proceedings of the 6th Very Large Databases (VLDB) Conference, Montreal*, pp. 137–146. IEEE, Los Alamitos (1980)
7. Groppe, S., Groppe, J.: A Prototype of a Schema-Based XPath Satisfiability Tester. In: Bressan, S., Küng, J., Wagner, R. (eds.) *DEXA 2006*. LNCS, vol. 4080, pp. 93–103. Springer, Heidelberg (2006)
8. Groppe, J., Groppe, S.: Satisfiability-Test, Rewriting and Refinement of Users' XPath Queries According to XML Schema Definitions. In: Manolopoulos, Y., Pokorný, J., Sellis, T.K. (eds.) *ADBIS 2006*. LNCS, vol. 4152, pp. 22–38. Springer, Heidelberg (2006)
9. Groppe, J., Groppe, S.: Filtering Unsatisfiable XPATh Queries. In: *Proc of the 8th Intl. Conf. on Enterprise Information Systems: Databases & Information Syst Integration, ICEIS 2006*, pp. 157–162 (2006)
10. Gupta, K.A., Suciu, D.: Stream Processing of XPath Queries with Predicates. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 419–430 (2003)
11. King, J.: Quist: A system for semantic query optimization in relational databases. In: *Very Large Database (VLDB)*, pp. 510–517. IEEE Computer Society, Los Alamitos (1981)
12. Ramanan, P.: Efficient algorithms for minimizing tree pattern queries. In: *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 299–309 (2002)
13. Shenoy, S.T., Ozsoyoglu, Z.M.: Design and Implementation of a Semantic Query Optimizer. *IEEE Transactions on Knowledge and Data Engineering* 1(3), 344–361 (1987)

14. Su, H., Murali, M., Rundensteiner, E.: Semantic Query Optimization in an Automata Algebra Combined XQuery Engine over XML Streams. In: Proceedings of the 30th Very Large Data Bases (VLDB) Conference, Toronto, Canada, pp. 1293–1296 (2004)
15. Su, H., Rundensteiner, E., Mani, M.: Semantic Query Optimization for XQuery over XML Streams. In: Proceedings of the 31st Intl. Conference on Very Large Data Bases (VLDB), pp. 277–282 (2005)
16. Sun, W., Liu, D.: Using Ontologies for Semantic Query Optimization of XML Databases. In: Nayak, R., Zaki, M.J. (eds.) KDXD 2006. LNCS, vol. 3915, pp. 64–73. Springer, Heidelberg (2006)
17. Wang, G., Liu, M., Yu, J.: Effective Schema-Based XML Query Optimization Techniques. In: Proceedings of the 7th Intl. Database Engineering and Application Symposium (IDEAS), pp. 1–6 (2003)

# Merging Views Containing Outer Joins in the Teradata DBMS

Ahmad Ghazal, Dawit Seid, and Alain Crotte

Teradata Corporation  
100 N. Sepulveda Blvd. El Segundo, CA, 90245  
{ahmad.ghazal,dawit.seid,alain.crotte}@teradata.com

**Abstract.** Query rewrite (QRW) optimizations apply algebraic transformations to a SQL query  $Q$  producing another SQL query  $Q'$  such that  $Q$  and  $Q'$  are semantically equivalent (i.e. produce the same result) but  $Q'$  can be executed more efficiently than  $Q$ . Merging views (as well as derived tables) to their parent SQL block is an important part of QRW which creates more opportunities for numerous other query optimizations like optimal join order and enabling other rewrites like redundant join elimination. This paper presents novel and practical techniques of view merging in the presence of outer joins which are implemented in the Teradata 12.0 DBMS. We also present experimental results that demonstrate performance improvements achieved using these techniques.

## 1 Introduction

The growing sophistication of business intelligence applications developed on top of relational DBMSs is tremendously increasing the complexity of automatically generated queries submitted 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 applications. For example, in data warehousing, numerous views and derived tables can be used for security reasons, for implementing business rules and semantic layers or for capturing complex intermediate results.

Merging views and derived tables, which we will henceforth simply call views, to their referencing SQL blocks is an important query rewrite (QRW) optimization. View merging reduces query complexity and enables more efficient global join ordering of the relations within the view and the referencing query; view merging allows the query optimizer to pick a join order that does not have to first join the relations inside the view, if doing so turned out to be non-optimal. There are numerous other benefits of view merging such as enabling other rewrites, such as redundant join elimination, and reducing the number of query blocks thereby eliminating the need for predicate move around and associated overhead of inter-block optimizations.

The basic philosophy in Teradata is that view merging rewrite should be rule based and must be exploited whenever semantically possible. This approach is based on the observation that, from the overall query optimization perspective, the search space

that includes spooled views is a subset of the search space where views are merged. In the Teradata DBMS, views with no aggregation, statistical window functions or sampling are always merged regardless of the structure of the view referencing block. In some cases, views with aggregations, statistical window functions or sampling are also merged. The Teradata DBMS also merges majority of views containing outer joins in either their own definition or in their referencing query. This is an interesting capability because merging views involving outer joins poses a number of challenges including how to correctly merge the WHERE clause of a view. This requires determining whether to merge the WHERE clause of a view to the containing block's WHERE clause or one of its ON clauses. Such issues are particularly intricate when a view is on the null-supplying side of an outer join, and the challenges are further compounded when such a view contains one or more null-sensitive expressions (NSE's), which will be defined below. In this paper, we focus on merging such views by identifying the semantic requirements that may force view materialization and devising SQL rewrites to merge them without violating these requirements.

## 2 Problem Definition

As mentioned in the previous section, the focus of this paper is on view merging in the presence of outer joins where the null supplying side is a view with NSE's. Before addressing this case, we describe how view merging is performed in the presence of outer joins using a simple example illustrated by query Q1 below. This query is based on the TPC-H data model which we will be using throughout the paper. The TPC-H data model consists of 8 tables covering eight years of data from a retail application (see [11] for full description). The main tables are `orders` and its child table `lineitem`. The `orders` table has the column `o_custkey` referencing the customer table and although `o_custkey` is not nullable, some customers do not have orders.

```
Q1: SELECT DT.y
      FROM customer LEFT JOIN
           ( SELECT o_custkey, o_orderkey
             FROM ordertbl
           ) DT(x,y)
      ON c_custkey=DT.x;
```

Q1 is a very simple query where the view is the null supplying side of the outer join and it consists of a single table. Merging DT to the main query produces Q1' below:

```
Q1': SELECT o_orderkey as y
       FROM customer LEFT JOIN ordertbl
       ON c_custkey=o_custkey;
```

Q1' is produced by "gluing" the definition of the view to the main query. In this case the view is a retrieve from a single table and the view reference is replaced by the table reference. Q2 below illustrates a more complex case of view folding where

the view has outer joins and a WHERE clause. Q2' is the rewritten Q2 after applying view folding.

```
Q2: SELECT DT.y
      FROM customer LEFT JOIN
        ( SELECT o_custkey, o_orderkey
          FROM ordertbl LEFT JOIN lineitem ON l_orderkey = o_orderkey
          WHERE o_comment like '%urgent%'
        ) DT(x,y)
      ON c_custkey=DT.x;
```

```
Q2': SELECT o_orderkey as y
       FROM customer LEFT JOIN
         (ordertbl LEFT JOIN lineitem ON l_orderkey = o_orderkey)
         ON c_custkey=o_custkey AND o_comment like '%urgent%';
```

In Q2', DT's definition is also glued to the main query by merging the join clause of the view to the join clause of the referencing query. Note that the WHERE clause of DT is merged to the ON clause of the referencing query. The reason is that, semantically, a predicate from an ON clause can be applied on the null supplying side of the outer join before executing the outer join. Hence, the result in Q2' is semantically equivalent to executing the WHERE clause before the outer join between DT and customer, as specified in Q2. Note that if DT was not on the null supplying side, then the WHERE clause of DT would merge with the WHERE clause of the referencing query instead of the ON clause of the referencing query.

Although the above examples we used to illustrate the semantics of view merging in the presence of outer joins only included simple views, note that view merging in Teradata covers more complex cases where the view and/or query have aggregations, window statistical functions, and so on. The detail of such cases is outside the scope of this paper.

Next we show that if the view has NSE's and the view is on the null supplying side of an outer join, the aforementioned merging logic may not work correctly. Consider Q3 and Q4 below which illustrate this semantic problem:

```
Q3: SELECT DT.y
      FROM customer LEFT JOIN
        ( SELECT o_custkey, 'const' FROM ordertbl
        ) DT(x,y)
      ON c_custkey=DT.x;
```

```
Q4: SELECT DT.y
      FROM nation LEFT JOIN
        ( SELECT c_nationkey, COALESCE(o_clerk,'AAA')
          FROM customer LEFT JOIN ordertbl ON c_custkey=o_custkey
        ) DT(x,y)
      ON n_nationkey=DT.x;
```

The straightforward merging of the view DT in Q3 and Q4 produces Q3' and Q4' below:

```
Q3':SELECT 'const'AS y
      FROM customer LEFT JOIN ordertbl ON c_custkey = o_custkey;
```

```

Q4':SELECT COALESCE (o_clerk,'AAA') AS y
      FROM (customer LEFT JOIN ordertbl ON c_custkey= o_custkey)
           LEFT JOIN nation ON n_nationkey =c_nationkey;

```

Q3' is an incorrect rewrite since it may produce different results than Q3 for the constant expression; Q3 requires the constant expression to have nulls in the final result for un-matching rows of ordertbl but Q3' will produce constants for such rows. Likewise in Q4, the rewrite in Q4' is an incorrect rewrite since it may produce different results than Q4 for the coalesce expression. We refer to constant expressions and expressions like coalesce as *null sensitive expressions* (NSE's).

One may consider avoiding the semantic problem caused by the NSE in queries like Q4' by disallowing the join order that first joins nation and customer and force the join planner to evaluate NSEs at specific points and materialize the results. However, this solution is not transparent to the join planner and it also defeats the purpose of merging DT.

### 3 Teradata's Solution

The solution involves picking a base relation among the relations in the view that represents the view, which will disappear after merging, and associating the NSE's to this relation. Specifically, we encapsulate NSE's by a CASE expression in the merged query that returns nulls for the view's non-matching rows and returns the NSE itself for the matching rows. For example, Q3 can be rewritten as:

```

Q3'': SELECT CASE WHEN ordertbl.rowid IS NULL
                 THEN NULL ELSE 'const' END as y
      FROM customer LEFT JOIN ordertbl ON c_custkey=o_custkey;

```

Similarly, Q4 can be rewritten as:

```

Q4'': SELECT CASE WHEN customer.rowid IS NULL THEN NULL
                 ELSE COALESCE(o_clerk,'AAA') END AS y
      FROM nation LEFT JOIN
           (customer LEFT JOIN ordertbl ON c_custkey=o_custkey)
           ON n_nationkey=c_nationkey;

```

In general, the case expressions is rewritten as *CASE WHEN <,relation>.non-nullable-field IS NULL THEN (NULL) ELSE (NSE) END*). The rewrite basically checks whether a non-nullable field (a field that is defined to not have nulls before the outer join) is null after the outer join. If it is null for a row, then it means the row is part of the non-matching result and the NULL result is produced in that case. Otherwise, it is a matching row and the original NSE is used for the result. We used rowid (the relation row identifier) as the non-nullable field since it is always not null.

Next we briefly discuss the basic strategy for picking the relation that represents the view in the CASE expression once the view is merged. In Q3, ordertbl table is used to represent the view and it is obvious that the join between customer and ordertbl is non-matching if and only if the join between customer and DT is non-matching. For Q4, customer table is picked to represent DT since it is not a null

supplying relation of any join in the view and therefore can represent the view. If the view has a full outer join (at the highest level) then both relations of the full outer join need to be used in the case expression. For example, consider Q5, a variant of Q4 with a full outer join instead of the left outer join in DT. The merging of DT in Q5 below results in Q5' where both `customer` and `ordertbl` are considered outer most relations.

```
Q5: SELECT DT.y FROM nation LEFT JOIN
      (SELECT c_nationkey, COALESCE(o_clerk, 'AAA') FROM customer
       FULL JOIN ordertbl ON c_custkey=o_custkey) DT(x,y)
      ON n_nationkey=DT.x;

Q5': SELECT CASE
        WHEN customer.rowid IS NULL AND ordertbl.rowid IS NULL
        THEN NULL
        ELSE COALESCE(o_clerk, 'AAA')
      END AS y
FROM   nation LEFT JOIN
      (customer FULL JOIN ordertbl ON c_custkey=o_custkey)
      ON n_nationkey=c_nationkey;
```

If the outermost join in the view is an inner join, then either side can be used to represent the view in the CASE expression. We use Q6 and Q6' below to show such cases. Q6 is similar to Q4 and Q5 but with an inner join in the view definition. Either `ordertbl` or `customer` table can be used as the outer most table in DT; we picked the `ordertbl` table in Q6':

```
Q6: SELECT DT.y
      FROM nation LEFT JOIN
           ( SELECT c_nationkey, COALESCE(o_clerk, 'AAA')
           FROM customer INNER JOIN
                ordertbl ON c_custkey=o_custkey
           ) DT(x,y)
      ON n_nationkey=DT.x;

Q6': SELECT CASE
        WHEN ordertbl.rowid IS NULL THEN NULL
        ELSE COALESCE(o_clerk, 'AAA')
      END AS y
FROM   nation LEFT JOIN
      (customer INNER JOIN ordertbl ON c_custkey=o_custkey)
      ON n_nationkey=c_nationkey;
```

## 4 Related Work

The research literature on view merging is limited to relatively straightforward cases [1,2,3] where neither the views nor the referencing queries contain outer joins. Also, a large class of mergeable views in general, and views containing outer joins and NSEs in particular, are currently materialized in many commercial DBMS products [4,5,6]. To the best of our knowledge, ours is the first work to present techniques to fold views containing NSEs. Note that rewrites involving materialized view with outer joins like those in [10] are different issue than ours.

There are a few papers that showed how to exploit view merging to enable further optimization. In [7], a technique is presented to leverage view merging to eliminate redundant joins including inner joins involving views that are defined on the same set of base tables, between views and their underlying base tables or self-joins of same relations. In all these cases, redundant joins that would have to be executed if the views were materialized were avoided by merging the views. The join elimination is based on the functional dependencies of the base relations that become visible to the query optimizer after view merging. In [8], the technique from [7] is extended for a nested relational data model which is composed of tuples, sets and lists. In our previous work [9], we have addressed how view merging interacts with other query rewrite rules.

## 5 Experimental Validation

To illustrate the effect of view merging we performed a set of experiments using the TPC-H data model simulating a situation often encountered in customer situations with inconsistent data. These experiments were conducted on a Teradata appliance system at volume points 10GB, 100GB, 300GB and 1 TB. In real-life the most common issue are missing values in fields or records missing which manifest in the data as NULL values. In order to simulate this phenomenon, we modified the `orders` table so that approximately 1% of the records end up with a `customer` field equal to NULL.

To conduct our experiments, we defined a view left joining `orders` with `customers` and replacing the status of the order with 'UNKNOWN' for the orders which have no corresponding customer. The view definition is shown below. Note that `ostatus` is an NSE in the view `orderstatus` which requires our solution to merge the view when referenced in queries:

```
CREATE VIEW orderstatus(orderkey, ordstatus, customer_name,
                       mktsegment) AS
SELECT o_orderkey, COALESCE(o_orderstatus, 'UNKNOWN') AS ostatus,
       c_name, c_mktsegment
FROM ordertbl LEFT OUTER JOIN customer ON o_custkey = c_custkey;
```

We then ran two queries using the view and the `lineitem` table. The queries perform a left outer join between `lineitem` and the view and they project the order key, the status of the order and the revenue associated with the line items shipped within a particular time period. The first query (R1) generated a report for a period of one month:

```
R1:SELECT l_orderkey, ordstatus,
        SUM(l_extendedprice*(1-l_discount)) AS revenue
FROM lineitem LEFT OUTER JOIN orderstatus
ON l_orderkey = orderkey
WHERE l_shipdate BETWEEN DATE '1993-01-01' AND
      DATE'1993-01-01' + INTERVAL '1' YEAR
GROUP BY l_orderkey, ordstatus;
```



The second query (R2) generated a report for a period of 5 years and for the market segment “HOUSEHOLD”:

```
R2: SELECT l_orderkey, ordstatus,
        SUM(l_extendedprice*(1-l_discount)) AS revenue
FROM lineitem LEFT OUTER JOIN orderstatus
on l_orderkey = orderkey
WHERE l_shipdate BETWEEN DATE '1993-01-01' AND
      DATE'1993-01-01' + INTERVAL '5' YEAR
      AND mktsegment = 'HOUSEHOLD'
GROUP BY l_orderkey, ordstatus;
```

The results for the elapsed times in seconds are presented in the graphs below. In R1, view merging allowed the join of lineitem to a one-month constraint to be done first. This caused early elimination of most of the rows yielding a very low elapsed time as compared to the case where the view is spooled. In the latter case the join between customer and orders occurred first and the subsequent join to lineitem last. For R2, while merging the view resulted in the same join order as the case where the view is spooled (optimal plan involves joining orders and customer tables first), as the result shows, there was little overhead incurred by our CASE expression re-write.

These results show an excellent consistency across all data sizes. For the first report R1, view merging provides an improvement between 6X and 10X. For R2 the view merging overhead is less then 1% of the execution time which suggests that our solution works for all cases regardless of the final join order.

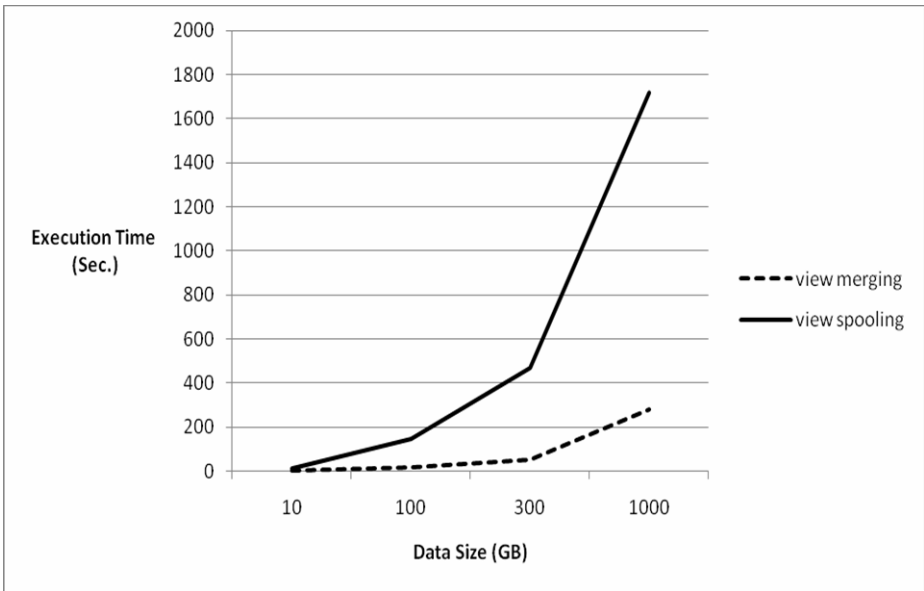


Fig. 1. R1's Execution time for view merging versus view spooling

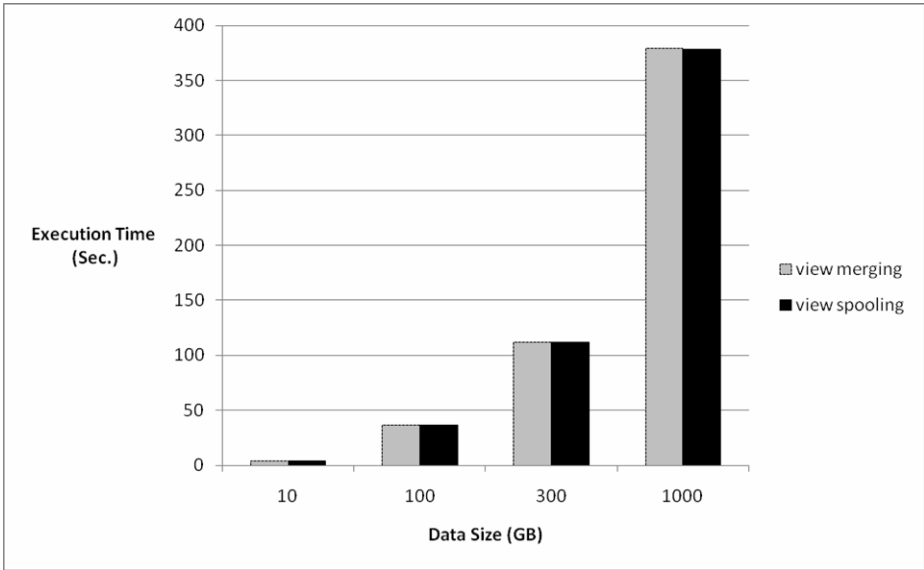


Fig. 2. R2's Execution time for view merging versus view spooling

## 6 Conclusion

This paper provides the first view merging algorithm for views containing null-sensitive expressions (NSEs) that are involved in outer joins. Such queries arise naturally in many business intelligence applications and merging the views plays a critical role in executing such queries efficiently. The challenge in merging views containing NSEs is that outer joins make it impossible to distinguish null values from NSEs from null values produced by outer joins once the views are merged. The algorithm proposed in this paper associates a non-nullable field to the view and ensures that rows from the view remain distinguishable in the merged view. Experimental results on a few queries show substantial improvements in query performance where view merging is useful. Another set of experiments were also conducted on queries for which view merging did not improve their execution time and the results show that very little overhead is incurred by our re-write.

## References

- [1] Pirahesh, H., Hellerstein, J.M., Hasan, W.: Extensible/rule Based Query Rewrite Optimization in Starburst. In: SIGMOD, pp. 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] Ahmed, R., Lee, A., Witkowski, A.: Cost-Based Query Transformation in Oracle. In: VLDB, pp. 1026–1036 (2007)
- [4] IBM Corp., DB2 Version 9.1 for z/OS: Performance Monitoring and Tuning Guide, 6th edn. (April 2009)

- [5] Oracle Corp., Oracle Database 11g Release 2 (11.2) Documentation
- [6] Microsoft Corp, SQL Server 2008 Product Documentation (2009), <http://msdn.microsoft.com/en-us/library/bb418440SQL.10.aspx>
- [7] Ott, N., Horländer, K.: Removing redundant join operations in queries involving views. *Inf. Syst.* 10(3), 279–288 (1985)
- [8] Südkamp, N., Linnemann, V.: Elimination of views and redundant variables in an SQL-like database language for extended NF2 structures. In: VLDB, pp.302–313 (1990)
- [9] Ghazal, A., Seid, D., Crolotte, A., McKenna, B.: Exploiting Interactions among Query Rewrite Rules in the Teradata DBMS. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 596–609. Springer, Heidelberg (2008)
- [10] Larson, P., Zhou, J.: View matching for outer-join views. *The VLDB Journal* 16(1), 29–53 (2007)
- [11] TPC-H specification – Transaction Performance Council, <http://www.tpc.org>

# A Hybrid Index Structure for Set-Valued Attributes Using Itemset Tree and Inverted List\*

Shahriyar Hossain and Hasan Jamil

Department of Computer Science  
Wayne State University, Michigan, USA  
shah\_h@wayne.edu, jamil@cs.wayne.edu

**Abstract.** The use of set-valued objects is becoming increasingly commonplace in modern application domains, multimedia, genetics, the stock market, etc. Recent research on set indexing has focused mainly on containment joins and data mining without considering basic set operations on set-valued attributes. In this paper, we propose a novel indexing scheme for processing *superset*, *subset* and *equality* queries on set-valued attributes. The proposed index structure is a hybrid of itemset-transaction set tree of “frequent items” and an inverted list of “infrequent items” that take advantage of the developments in itemset research in data mining. In this hybrid scheme, the expectation is that basic set operations with frequent low cardinality sets will yield superior retrieval performance and avoid the high costs of construction and maintenance of item-set tree for infrequent large item-sets. We demonstrate, through extensive experiments, that the proposed method performs as expected, and yields superior overall performance compared to the state of the art indexing scheme for set-valued attributes, i.e., inverted lists.

## 1 Introduction

The need for containment queries involving set-valued attributes is found in a variety of application areas, ranging from scientific databases to XML documents, annotation databases, market basket analysis, production models, multimedia [4,6], bio-molecular databases [1], etc. Containment queries span a wide range of query families, from simple existence queries to composite similarity, pattern matching, or graph isomorphism queries. For example, against a movie annotation database we could ask “*find all the movies where Tom Hanks and Meryl Streep both performed*”, or “*find all users who visited sports or finance pages but nothing else*” in a database of internet usage.

To evaluate containment queries, namely, *subset*, *superset* and *equality* efficiently, we need targeted access to data using index structures. The state of the art method for indexing set-valued databases is inverted list [12]. The problem with inverted lists, however, is that for frequent items, they grow quite large. As

---

\* This research was partially supported by National Science Foundation grants CNS 0521454 and IIS 0612203.

frequent items are queried frequently and regularly, the performance for query evaluation will degrade for very large databases with items of skewed distribution. In order to avoid the intersection of large inverted lists, we can actually devise an index structure where some such intersections are pre-computed and stored in memory or secondary storage. The idea of pre-computation is borrowed from existing research in spatial databases [11], and data mining [10].

Our contribution is summarized in the following way:

- We propose an index structure for set-valued attributes, called Mixed Inverted List Itemset-Tidset index or *MixIIT*. This is a hybrid of concept lattice and inverted lists which is kept in secondary storage. However, in *MixIIT*, we augment the concept lattice with itemset-tidset lists.
- We develop novel and efficient algorithms for computing basic set operations such as subset, superset and equality that leverages the proposed *MixIIT* structure. The details of the algorithms may be found in [13].
- We experimentally compare the performance of *MixIIT* with inverted list and show that not only does *MixIIT* perform superior to inverted lists, the low memory requirement of *MixIIT* makes it possible to exploit this structure for practical applications today.

The rest of the paper is organized as follows. We briefly discuss recent and relevant research in section [1]. In Section 2, we present the *MixIIT* index structure and in section 3, we describe the query evaluation algorithms. An extensive experimental evaluation is discussed in section 4 that compares *MixIIT* with the inverted lists index. Finally, section 5 summarizes our research.

## 1.1 Related Research

Reported research on the evaluation of basic containment queries with set-valued predicates are few and far between. So far, database research has mostly focused on similarity [17] and join queries [16]. The research on set containment queries can be divided into three major categories. First, we have signature based approaches where the transactions of varying lengths are converted into signature bit-strings of fixed length [7]. The greatest advantage of signature based methods is that set comparison operations such as subset, superset checking get reduced to simple bitwise operation. Signatures can be organized into various indexing structures such as Sequential Signature Files [15], Bit-Slice Signature file (BSSF), Multilevel Signature file, Compressed Multi Framed Signature file, S-Tree and its variants, Signature Graph and Signature tree [5]. The common problem with any signature based index is that signatures can only generate the candidate set of transactions which covers the query result. As a result, there will always be some false positives.

The second approach in indexing set-valued databases originated from information retrieval research. Inverted files or list consist of a directory containing all distinct items that can be searched for, and a list for each distinct item which contains the transactions where the item appeared [3]. As shown in [12], inverted

list based indexing techniques perform better than signature file based ones for different database, set, vocabulary and query cardinality. But, transaction lists for frequent items may grow quite large which may eventually slow down the query evaluation process.

In [11], Hellerstein et. al. proposed a R-tree [9] based indexing method for set-valued databases. Computing bounding box for itemsets is not as simple as that for spatial data points. Also, storing bounding box of unequal lengths creates maintainability issues. In [19], Terrovitis et al proposed an indexing technique which combines the advantages of inverted lists with the power of in-memory index for frequent itemset. However, it is not possible to update the FP-tree [10] based *access tree* online.

## 2 Index Structure

In this section, we introduce a hybrid index that combines a main memory itemset-tidset representation of concept lattice with an inverted list residing in secondary storage. First, we give background information for inverted lists and itemset-tidsets and explain their benefits and drawbacks. Then, we show how these indexing schemes are combined in the set indexing system.

**Table 1.** Example Database

(a) Vocabulary	(b) Transaction Database																				
<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border: 1px solid black; padding: 2px;">Items</td> <td style="border: 1px solid black; padding: 2px;">A</td> <td style="border: 1px solid black; padding: 2px;">C</td> <td style="border: 1px solid black; padding: 2px;">D</td> <td style="border: 1px solid black; padding: 2px;">T</td> <td style="border: 1px solid black; padding: 2px;">W</td> </tr> </table>	Items	A	C	D	T	W	<table style="border-collapse: collapse; width: 100%;"> <tr> <td style="border: 1px solid black; padding: 2px;">Transaction</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">2</td> <td style="border: 1px solid black; padding: 2px;">3</td> <td style="border: 1px solid black; padding: 2px;">4</td> <td style="border: 1px solid black; padding: 2px;">5</td> <td style="border: 1px solid black; padding: 2px;">6</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">Items</td> <td style="border: 1px solid black; padding: 2px;">ACTW</td> <td style="border: 1px solid black; padding: 2px;">CDW</td> <td style="border: 1px solid black; padding: 2px;">ACTW</td> <td style="border: 1px solid black; padding: 2px;">ACDW</td> <td style="border: 1px solid black; padding: 2px;">ACDTW</td> <td style="border: 1px solid black; padding: 2px;">CDT</td> </tr> </table>	Transaction	1	2	3	4	5	6	Items	ACTW	CDW	ACTW	ACDW	ACDTW	CDT
Items	A	C	D	T	W																
Transaction	1	2	3	4	5	6															
Items	ACTW	CDW	ACTW	ACDW	ACDTW	CDT															

### 2.1 Inverted File Index

The inverted file index is essentially a two dimensional list. The first dimension is a list of all the distinct items appearing in the database. Each node in the vocabulary list points to a list of transactions where the item appeared. For containment query evaluation, one needs to store the length of every transaction along with the transaction id.

The evaluation of set based queries on inverted file index may be trivial but maintaining an inverted list is complicated. The lists may be huge for large databases as the id of a transaction is inserted as many times as the number of items it contains. As a result, theoretically, the size of the inverted file could be similar to the size of the transaction collection or even larger. Due to their size, the inverted lists are stored in secondary storage. Therefore, the larger these inverted lists are, the more pages have to be retrieved from the disk for evaluating a query. Moreover, the most frequent items will have the longest inverted lists. This is particularly damaging for the evaluation of set-valued queried, as the topmost frequent items are usually the ones most frequently queried.

## 2.2 Itemset-tidset Index

Concept lattices are used in many application areas to represent conceptual hierarchies among objects in the underlying data. The field of Formal Concept Analysis [8] has grown to a powerful theory for data analysis, information retrieval and knowledge discovery. There is an increasing interest in the application of concept lattices for data mining, especially for mining association rules [14] and generating frequent itemset [20,23]. Despite their numerous applications, concept lattices have never been employed for indexing set valued databases. In this section, we will present the conceptual model of lattice based index structure. We begin the discussion by defining some of the topics for formal concept analysis from [21] in the light of set valued databases.

**Definition 1.** A formal concept is a pair  $(A, B)$  with  $A \subseteq G$ ,  $B \subseteq M$ ,  $A' = B$  and  $B' = A$ . (This is equivalent to  $A \subseteq G$  and  $B \subseteq M$  being maximal with  $A \times B \subseteq I$ .)  $A$  is called extent and  $B$  is called intent of the concept.

It is easy to see that even for a small vocabulary the number of concepts can grow very large. One possible way of combating this growth is to store only the most frequently occurring elements as in closed itemsets [22]. In the context of concept lattices, and for real-life databases, an iceberg [18] type solution seems appropriate where only the frequent items are stored in the lattice and less frequent items are still kept in an inverted list like structure so that the membership of the items in these two structures can be adjusted (following database updates) when the items become less or more frequent. The index structure discussed next based on concept lattice and inverted list captures this spirit.

## 2.3 Mixed Inverted List Itemset-Tidset (*MixIIT*) Index

In *MixIIT* index structure we store the frequent items in the database in the Itemset-Tidset search tree built essentially using the algorithm in [22]. However, we deviate from the algorithm in the following principal ways:

- To facilitate subset queries, all the edges are considered bidirectional.
- We avoid strict support based pruning of the itemset-tidset as proposed in [22]. For example, let item  $A$  and  $B$  passes the support threshold but, itemset  $AB$  falls below the cutoff. If we remove itemset  $AB$  as proposed in [22], then we will not be able to evaluate queries containing  $AB$  correctly. So, in the lattice structure, we store *all* the concepts which can be constructed with the frequent items, not just the ones which pass the support threshold.
- Finally, we define a special concept, called *sink*, which contains all the frequent items in its intent. The sink concept is connected with all the leaf nodes of the concept tree. This arrangement is necessary for the evaluation of the subset queries where we need to start from the most general intent and subsequently look for the specific one.

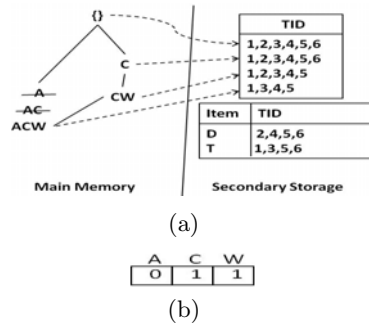
As the extent of every concept can grow quite large, we store them in a secondary storage, which can serve as the file system or an SQL database. Itemset-tidset tree of only frequent items will generate incorrect answers for subset and superset queries with infrequent items in the query set. We get around this issue by maintaining the inverted list of transactions for the infrequent items in the secondary storage. In Figure 1(a), we show the *MixIIT* index for the example database in Table 1. We create an itemset-tidset tree for the top three frequent items. The three items, namely *A*, *D* and *T* equally deserve the position for third frequent item. We made a random choice *A*. Also, *ACW* is the closure of *A* and *AC*. So, using the closure properties described in [22], we can replace *A* and *AC* from the tree with their closure. Finally, we added links from the sink concept *ACW* to all the leaves of the tree which, in this example, is only *CW*. The inverted lists of infrequent items *D* and *T* are stored in the secondary storage.

The intents are stored in bit string format as shown in Figure 1(b). This representation is different from signature as the transformation between itemset and its corresponding bit string is one to one. With bitstring representation, we can compute subset or superset with a bitwise comparison instead of set manipulation. For example, if *a* and *b* are the bit string representation of itemset *A* and *B*, respectively and if  $A \subseteq B$  then  $a \wedge b = a$ .

### 3 Query Processing Using MixIIT

In this section, we present the generic approach taken for evaluating the three types of queries we are interested in: subset, equality and superset. In [13], we present a more detailed discussion on the evaluation process.

The evaluation algorithms for all three types of queries have two main stages: (a) evaluation in the itemset-tidset tree, and (b) evaluation in the inverted file. The frequent items from the query set are used to traverse the concepts in the itemset-tidset tree. A set of candidate transaction ids is created in the process. Next, the infrequent items are used to extract transactions ids from the inverted list. In the final stage, two lists of transaction ids are combined based on the nature of the query. The basic idea is that we use the intent of the concepts in the itemset-tidset tree to quickly trace



**Fig. 1.** (a) *MixIIT* index for the database in Table 1 (b) Bit String representation of the itemset *CW*.

a candidate answer to the query. The benefit is quite significant since we can avoid costly union and intersection operations between large transaction sets.



## 4 Experimental Evaluation

We decided to compare the performance of *MixIIT* with inverted list based indices because in [12], the authors have shown that inverted lists perform better than signature based index structures for low cardinality set valued attributes, and even outperforms B-trees for containment queries in RDBMS [24].

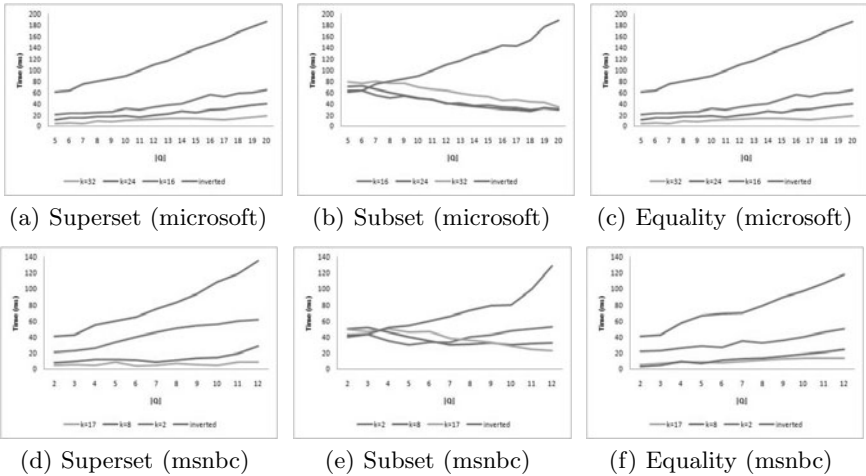
### 4.1 Data Sets and Performance Evaluation

For evaluation purposes, we use two real data sets from UCI KDD archive [2]. The first data set is a one-week log tracing the virtual areas that users visited in the web portal [www.microsoft.com](http://www.microsoft.com). Each record corresponds to a user session and the set value comprises the areas visited. There are 32k records and the vocabulary of the data set contains 294 distinct items (areas). The distribution of the items in the records is skewed and the average size of the record is 3 items. on the web portal of [msnbc.com](http://msnbc.com) taken from the UCI KDD archive as well. The vocabulary here is very limited, comprising only 17 distinct items and unlike the previous one, the distribution of the items is relatively uniform. The average size of the record is 5.7 items. The total number of transactions for this data set is 989818. As a result, we end up with long inverted lists.

**Query Evaluation with microsoft data.** We created 15 queries with lengths varying from 5 to 20. The items from query were selected randomly from the vocabulary. In figure 2(a) through 2(c), we present the performance graph for subset, superset and equality query evaluation for `microsoft` data set. The data points in the plot depict average time required for queries of different lengths. For equality and superset queries, *MixIIT* outperforms inverted lists by a significant factor. The performance gain increases with the increase of  $|Q|$ . *MixIIT* structure performs better with the increase of frequent items in the Itemset-Tidset tree, as expected. We have mixed results for subset queries (figure 2(b)). For small query sets, *MixIIT* is not as efficient as inverted list and with the increase of frequent items,  $k$ , the performance degrades. As we discuss in [13], smaller query sets result in more choices for the recursive call. More frequent items in the lattice increases the number of leaf nodes and consequently, the number of upward links from the sink node increases. Hence, the performance degrades in both the cases. an increase in the number of frequent items stored,  $k$  from 16 to 32 there is an increase in the number of concepts in the itemset-tidset tree. There is also an increase in the number of leaves in the itemset-tidset tree from 1698 to 5678. So, there are more choices going upwards from the sink concept *ACDTW* and more paths to traverse for any given subset query. As a result, with the increase in  $k$ , the performance will most likely degrade.

**Query Evaluation with msnbc data.** We created 10 random queries with lengths varying from 2 to 12. The threshold value for the memory resident lattice was varied from 2 to 8 and 17, which covers all the items in the vocabulary. The main contribution of this experiment is that we could evaluate the *MixIIT*

structure against a small yet real life data set which can be indexed solely with in-memory lattice. The findings are presented in figure 2(d) through 2(f). One of the notable difference from the *microsoft* data is the behavior of  $k = 2$  threshold. At such a low threshold, the combination of itemset-tidset tree and inverted list is dominated by the performance of the inverted list. Another interesting observation is that the performance graphs of the inverted list as well as top 2 and 8 item *MixIIT* structures show an upward trend for large queries. This data set contains longer inverted lists for infrequent items compared to *microsoft* data set as there are more transactions for smaller vocabulary. As a result, the performance of the index structures, which depends on the inverted list begins to deteriorate with increasing query size.



**Fig. 2.** Performance comparison of *MixIIT* index with inverted list. (a-c): for *microsoft* data. (d-e): for *msnbc* data. Inverted list performance is shown as the top most curve in each graph.

## 5 Summary

In this paper, we have proposed a novel indexing scheme, *MixIIT*, which combines a main memory resident itemset-tidset tree with an inverted file for infrequent items, kept in secondary storage. We also introduced novel evaluation algorithms for subset, superset and equality queries using the *MixIIT* index. Finally, we performed an experimental study comparing *MixIIT* with inverted lists for these queries. We found that in the case of superset and equality queries, *MixIIT* clearly outperforms the inverted lists with reasonable main-memory overhead. Although much lower than inverted lists, the costs for subset queries in *MixIIT* appears to be a bit higher for smaller  $k$ , and showed trends of improvement as  $k$  grew larger. The primary goal of subset query is to find the minimal coverage of the frequent items in the query set. As a future work, we

wish to use a hash or  $B^+$ -tree based secondary structure which will help us find the minimal coverage without traversing the tree. We will also investigate how other forms of queries such as set intersection, union, join etc. can be computed with *MixIIT* structure. Finally based on the experimental results, it is apparent that relaxing the threshold for items in the lattice will not always guarantee better performance. It may even get worse, as we discovered for the subset queries. We plan to establish the relationship among the size of the possible set elements, number of records in the database, and the size of the sets allowed in a record so that an optimum threshold can be selected for best performance.

## References

1. Bairoch, A., Apweiler, R.: The swiss-prot protein sequence data bank and its supplement trembl. *Nucleic Acids Res.* 27, 49–54 (1997)
2. Bay, S.D., Kibler, D., Pazzani, M.J., Smyth, P.: The uci kdd archive of large data sets for data mining research and experimentation. *SIGKDD Explor. Newsl.* 2(2), 81–85 (2000)
3. Bertino, E., Tan, C.K.-L., Ooi, B.C., Sacks-Davis, R., Zobel, J., Shidlovsky, B.: *Indexing Techniques for Advanced Database Systems*, pp. 151–184. Kluwer, Dordrecht (1997)
4. Böhm, K., Rakow, T.C.: Metadata for multimedia documents. *SIGMOD Record* 23, 21–26 (1994)
5. Chen, Y.: On the signature tree construction and analysis. *TKDE* 18(9), 1207–1224 (2006)
6. Jain, R., Hampapur, A.: Metadata in Video Databases. *SIGMOD Record* 23(4), 27–33 (1994)
7. Faloutsos, C.: Signature files. In: *Information Retrieval: Data Structures & Algorithms*, pp. 44–65 (1992)
8. Ganter, B., Stumme, G., Wille, R.: Formal concept analysis: Theory and applications 10(8), 926–926 (2004)
9. Guttman, A.: *R-trees: a dynamic index structure for spatial searching*. Morgan Kaufmann Publishers Inc., San Francisco (1988)
10. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *DMKD* 8(1), 53–87 (2004)
11. Hellerstein, J.M., Pfeffer, A.: The RD-tree: an index structure for sets. Technical Report 1252, University of Wisconsin at Madison (1994)
12. Helmer, S., Moerkotte, G.: A performance study of four index structures for set-valued attributes of low cardinality. *The VLDB Journal* 12(3), 244–261 (2003)
13. Hossain, S., Jamil, H.: *MixIIT: A hybrid index structure for set-valued attributes*. Technical report, Wayne State University, USA (2010)
14. Hu, K., Lu, Y., Shi, C.: Incremental discovering association rules: A concept lattice approach. In: Zhong, N., Zhou, L. (eds.) *PAKDD 1999*. LNCS (LNAI), vol. 1574, pp. 109–113. Springer, Heidelberg (1999)
15. Ishikawa, Y., Kitagawa, H., Ohbo, N.: Evaluation of signature files as set access facilities in oodbs. In: *SIGMOD*, pp. 247–256. ACM Press, New York (1993)
16. Mamoulis, N.: Efficient processing of joins on set-valued attributes. In: *SIGMOD*, pp. 157–168 (2003)
17. Mamoulis, N., Cheung, D., Lian, W.: Similarity search in sets and categorical data using the signature tree. In: *ICDE*, March 2003, pp. 75–86 (2003)

18. Stumme, G., Taouil, R., Bastide, Y., Lakhal, L.: Conceptual clustering with iceberg concept lattices. In: Proc. of GI-Fachgruppentreffen Maschinelles Lernen'01, Universität Dortmund (2001)
19. Terrovitis, M., Passas, S., Vassiliadis, P., Sellis, T.: A combination of trie-trees and inverted files for the indexing of set-valued attributes. In: CIKM, pp. 728–737 (2006)
20. Valtchev, P., Missaoui, R., Godin, R.: A framework for incremental generation of closed itemsets. *Discrete Appl. Math.* 156(6), 924–949 (2008)
21. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In: Rival, I. (ed.) *Ordered Sets*, September 1981. NATO Advanced Study Institute, vol. 83, pp. 445–470 (1981)
22. Zaki, M.J., Hsiao, C.-J.: Charm: An efficient algorithm for closed itemset mining. In: *SDM* (2002)
23. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: *KDD*, pp. 283–286. AAAI Press, Menlo Park (1997)
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*, pp. 425–436 (2001)

# Optimization of Disk Accesses for Multidimensional Range Queries<sup>\*</sup>

Peter Chovanec, Michal Krátký, and Radim Bača

Department of Computer Science  
VŠB–Technical University of Ostrava, Czech Republic  
{peter.chovanec,michal.kratky,radim.baca}@vsb.cz

**Abstract.** Multidimensional data structures have become very popular in recent years. Their importance lies in efficient indexing of data, which have naturally multidimensional characteristics like navigation data, drawing specifications etc. The R-tree is a well-known structure based on the bounding of spatial near points by rectangles. Although efficient query processing of multidimensional data is requested, the R-tree has been shown to be inefficient in many cases. From the disk access cost point of view, the main issue of range query processing is the expensive cost of random accesses during the tree traversal. In the case of queries with low selectivity, the sequential scan of all tuples may be more efficient than the range query processing. We focus on efficiency of the disk access cost and we present an optimization of the disk access cost during range query processing. Our method focuses on a leaf node retrieval and it can be simply adopted by any tree. We put forward our tests using the R-tree since it is the most common multidimensional data structure.

**Keywords:** R-tree, multidimensional data structures, multidimensional range query, disk random accesses, skip-sequential prefetch.

## 1 Introduction

Spatial data can be commonly found in various application areas including medicine, cartography, computer vision, molecular biology and many others. Query processing in high-dimensional spaces has therefore been a very prominent research area over the last few years. A number of new index structures and algorithms have been proposed [18].

There are two major approaches to multi-dimensional indexing [21]: data structures for indexing metric spaces and data structures for indexing vector spaces. The first approach includes, for example,  $n$ -dimensional B-tree [8], R-tree [9], R\*-tree [2], Signature R-tree [11], X-tree [3], UB-tree [1], and BUB-tree [7]. The second one includes M-tree [6], for example. A multi-dimensional data structure supports the following query types [21]: range/window queries,  $k$ -NN queries and so on. The range query retrieves all tuples of a multidimensional space matched by a query rectangle.

---

<sup>\*</sup> Work is partially supported by Grant of GACR No. P202/10/0573 and SGS, VŠB–Technical University of Ostrava, No. SP/2010138, Czech Republic.

As far as the tree-based multidimensional data structures are concerned, random disk accesses are one of their major weaknesses during the range query processing. Some optimization techniques for disk accesses have been primarily developed for the B-tree and other indices used in relational DBMS. In [16], a B-tree with multi-page disk reads was introduced. This structure introduced a support of multi-page disk read/write access for long sequential range retrievals. Another storage technique is tree-structured placement [4], that explicitly accounts for the mismatch between tree-structured data and disk drive characteristics. This technique uses the recently proposed idea of semi-sequential disk access to place the tree structure. In [5], an index clustering technique called the segment-page clustering (SP-clustering) was introduced. The SP-clustering avoids the scattering by storing the relevant nodes contiguously in a segment that contains a sequence of contiguous disk pages and improves the query performance by offering sequential disk access within a segment.

A modified storage layout of tree-based indices has been proposed in [20]. This method is based on efficient access to repeatedly read nodes at the top levels of the tree by storing tree nodes of the same level together. After an object has been inserted such that node splits occur, the swap of nodes is executed. This approach improves the query processing; however, the node swapping significantly affects the build time (up to 40%). The Multiresolution File Scan (MFS) technique has been presented in [17] as another manner to optimize random reads. MFS is based on a selection of flat files, so-called views, that represent the data set at multiple resolutions. It is a simple structure with around 5% of the data size storage overhead. Each of the files is accessed using a pseudo-optimal schedule which takes the transfer and positioning time of the disk into account.

In this work, we introduce an optimization of disk accesses for a multidimensional range query and we show that it is often possible to replace random accesses by the sequential read during query processing. Instead of developing another sophisticated data structure, our goal is to optimize disk accesses using the simplest possible way, which is applicable for any tree data structure. Since the R-tree is the most popular multidimensional data structure, we present the optimization of the R-tree.

R-tree [9] can be thought of as an extension of B-trees in a multi-dimensional space. It corresponds to a hierarchy of nested  $n$ -dimensional MBRs (see [9] for detail). R-tree performance is usually measured with respect to the retrieval cost (in terms of DAC) of queries. Variants of R-trees differ in the way they perform the split algorithm. The well-known R-tree variants include  $R^*$ -trees and  $R^+$ -trees. In [15], we can find a more detailed description as well as a depiction of other R-tree variants. Bulk-load techniques for the R-tree [13] try to minimize the number of pages and speed up the inserts of many records. Bulk-load is often based on a preordering of inserted items.

The R-tree is appropriate for our method since it contains overlapping minimal bounding boxes and many leaf node accesses can occur during the range query. We adopt a method called skip-sequential prefetch often used in RDBMS [12,14] and our approach is enhanced by the read buffer. The outline of the paper is

as follows: In Section 2, we introduce index organization and disk access optimization used in RDBMS. Section 4 presents our improved approach for efficient query processing in the R-tree. In Section 5, we put forward experimental results. Finally, we outline possible areas of our future work and conclude the paper.

## 2 Index Organization in Relational DBMS

### 2.1 DBMS Structures and I/O Activity

An index row is a useful concept for fast access to the table according to a key. Each index record contains a key and a pointer to the corresponding row in the table. The index records and the table rows are grouped into pages, where their size is usually 2kB, 4kB or 8kB [12]. The most common index structure used in DBMS is a B-tree and its variants (usually a B<sup>+</sup>-tree or a redundant B-tree).

DBMS usually defines the so-called clustering factor of an index which specifies how much the order of the records in the index corresponds to the order of the rows in the table. A smaller clustering factor is more efficient. If the clustering factor is high and we process a range query using the index, then we randomly access the table rows as we sequentially read the records from the index. Those random accesses to table rows have a very negative impact on the query processing efficiency.

Every RDBMS contains the area in the memory into which the index and the table pages are read from the secondary storage. This area is known as the buffer pool (or the data cache). The I/O activity is a transfer of a page from the secondary storage into a buffer pool; the page is read into the buffer pool. Every database has one or more buffer pools used to minimize a disk activity and avoid the necessity of frequent accesses to the secondary storage [19][12].

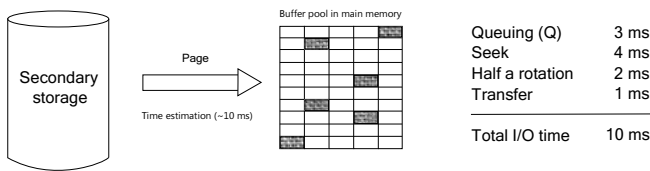


Fig. 1. Random reads from disk drive

Figure 1 shows the example of the enormous time cost involved with random page reading from the secondary storage into the buffer pool. The time to read one page from the disk takes roughly 10 ms, which means 100 disk accesses per second. Let us consider pages of 2kB in size. We obtain 200 kB/s instead of approximately 40 MB/s in the case of sequential accesses. The Solid-State Drives (SSD) provides the low read time and latency. The main advantage of SSD disks is a fast random access due to the fact there is no read/write head. In Section 4, we show that our method is applicable even in the case of SSD.

### 2.2 Skip-Sequential Prefetch

Reading more pages together means that read time is reduced. The read time of one page may be as low as 0.1 ms. Because RDBMSs know in advance which pages will be required, the reads can be performed before the pages are required. This process is called skip sequential prefetch (or sequential prefetch) [12], it reads the rows in one direction even when the clustering factor is high.

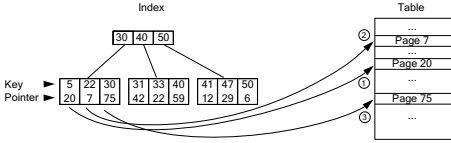


Fig. 2. Conventional random reads

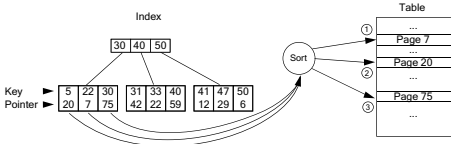


Fig. 3. Skip sequential prefetch reads

Figure 2 shows the conventional random reads from the secondary storage. Pages of the table are accessed randomly as they are found in the index.

In contrast, skip sequential prefetch sorts pages before they are accessed in the table (see Figure 3). Reading the pages in their physical order reduces seeking and consequently increases the efficiency of the range query processing.

## 3 Disk Access Optimization for Range Query

### 3.1 Range Query Processing

The range query algorithm traverses the tree from the root node and it follows only relevant records in each node. The record is relevant if its MBR is intersected by the query rectangle. The algorithm searches all subtrees recursively and it is finished after all relevant subtrees are processed. The R-tree has some features which can significantly slow down the range query processing. MBR of nodes often overlap and cover the dead space (the space without tuples). It can lead to the retrieval of leaf nodes with zero or few relevant items during the query processing. Similar false hits occur in inner nodes as well. Therefore, we often read more nodes than necessary. This problem escalates when the dimension of the data increases and it is known as the curse of dimensionality [21].

### 3.2 Disk Access Optimization

The creation of new nodes during the split operation over a node results in nodes not being sorted in the file according to the order in which they are retrieved by the range query algorithm. This issue leads to random accesses to the secondary storage during the query processing. An example of a small subtree is presented in Figure 4. Obviously, nodes are not sorted according to the pre-order traversal.

Let us take a range query defined by two points Ql and Qh. Regions R3, R4, and R5 are intersected by the query box and they are searched.



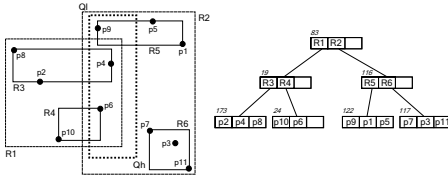


Fig. 4. Range query in the R-tree

They are read as they are found by the algorithm. The order in which the pages are read is 173, 24, and 122. These random accesses significantly increase the query processing time.

In this paper, we solve this problem by adopting a well known technique called skip sequential prefetch and we extend this method using a read buffer. The new method is called buffered prefetch. The basic idea of the skip sequential prefetch method can be seen in Algorithm 1. Relevant leaf nodes are not immediately read and searched, but their indices (pointers) are stored in a `LeafIndices` array until all relevant leaf nodes are found. We do not invoke the `BufferedPrefetch` method here (Line 16). We only sort the `LeafIndices` array, read the leaf nodes from the secondary storage, and find the result set records satisfying the range query.

The smaller skips between the read pages decrease the seek time and results in a more efficient read time. Moreover, current disks include their own cache (for example 8MB) which enables more efficient reading of the sorted pages.

---

**Algorithm 1.** Range query algorithm using prefetch techniques

---

**Input** : Range query `QB`  
**output**: Result set `ResultSet`

```

1 LeafIndices = empty array of leaf node indices
2 Z = stack of tree nodes
3 N = the root node
4 while ¬Z.IsEmpty() do
5     if there is the next record (MBR, Pchild) in N where the MBR overlap QB then
6         if Pchild node is a leaf node then
7             LeafIndices = LeafIndices ∪ Pchild;
8         else
9             Z.Push(N);
10            N = Pchild node;
11        end
12    else
13        N = Z.Pop();
14    end
15 end
16 ResultSet = BufferedPrefetch(LeafIndices, QB);

```

---

The buffered prefetch method uses the read buffer to further optimize the node retrieval. We use the following technique (see Algorithm 2):

1. We find a group of leaf nodes which are close to each other in the file (the nodes have close indexes).

**Algorithm 2.** BufferedPrefetch method

---

```

Input : Array of leaf node indices LeafIndices, Range query QB
output: Result set ResultSet

1 LeafIndices.Sort;
2 StartIndex = 0;
3 Start = LeafIndices[StartIndex ];
4 for  $i \leftarrow 0$  to LeafIndices.Count do
5   End = LeafIndices[ $i+1$ ];
6   if End - Start > Treshold then
7     LargeBlock = read a block from Start to LeafIndices[ $i$ ];
8     for  $j \leftarrow$  StartIndex to  $i$  do
9       LN = read the node LeafIndices[ $j$ ] from LargeBlock and store
          it in the buffer pool;
10      if LN contains points in QB then
11        Add such points into ResultSet;
12      end
13    end
14    StartIndex =  $i+1$ ;
15    Start = LeafIndices[StartIndex ];
16  end
17 end
18 return ResultSet;

```

---

2. We read the large block from the secondary storage containing this group of leaf nodes (Line 7).
3. We pick these leaf nodes from the read block in the buffer pool and the rest of the block is discarded (Line 9).

We work with a heuristic that the sequential disk accesses are up-to  $200\times$  more efficient than the random accesses. Nodes within a specific distance (so-called *Threshold*) are read together. The threshold value depends on several parameters like density of data set, selectivity of queries and so on. However, it is mainly influenced by a difference between the sequential and random search. The threshold can be easily estimated experimentally. We use a greedy approach to select the groups of nodes, which are sequentially read in one block. This makes the group selection very fast and we are able to do that in one scan through the array of the leaf node indices (Lines 4–17).

We only focus on leaf nodes, since they represent 90% of the tree size and they can rarely fit into the buffer pool [14]. The technique can also be successfully used in other levels of the tree; however, it is usually not necessary and the improvement is not so significant. It is clear that the buffered prefetch technique is simply extensible in a concurrent environment; concurrent threads of query processing can share the buffer including node indices.

## 4 Experimental Results

In our tests<sup>1</sup>, we compare query processing in the R\*-tree, an R\*-tree with skip sequential prefetch, and an R\*-tree with buffered prefetch. Two real collections have been chosen for these tests. The first collection, titled as XML, represents a set of paths in an XML document [10]. We selected a subset generated from the XMark collection<sup>2</sup> including 10-dimensional tuples. The second collection, titled as TIGER<sup>3</sup>, is a standard spatial data set. We choose the type 2 of the Wyoming data set from 2006, which includes 2D points without topological information. In Table 1, we see basic characteristics of created R-trees. In all experiments, we turn off the OS's disk read cache to prevent the OS from file caching.

**Table 1.** Basic characteristics of created R-trees

Data Collections	XML		TIGER	
Inserted items	1,031,080		5,889,786	
Page size [B]	2,048		512	
	Build insert	Bulkload	Build insert	Bulk-load
Height	4	4	5	4
Inner nodes	2,292	1,022	15,381	6,101
Leaf nodes	32,780	22,416	228,168	140,234
Index size [MB]	71.83	48	124.70	74.93

Efficiency of the range query processing was measured by Disk Access Cost (DAC), read time, and complete query processing time. We have used 15 various range queries from the result size 1 to the selectivity 52%, each query was processed 10× and the results were averaged. Disk access cost is measured by the volume of data read (*DAC*) and the number of disk accesses (*Disk accesses*).

Table 2 presents the average results of the range query processing with several threshold values. *Read pages* means the average number of pages read by one disk access. It seems that the threshold value 64 is the most efficient for queries on the XML collection. In this case, we get the maximum performance of the combination of sequential and random reading. The major issue of magnetic hard drives, as was mentioned above, is the high seek time in the case of random accesses. This fact can be seen in comparison of the R-tree and the bulk-loaded R-tree. Nodes with similar items are stored close to each other in the bulk-loaded index file. Therefore, the read time is significantly lower, although more than the twice number of disk accesses has been performed in the case of the bulk-loaded index.

The higher efficiency of Solid-State drives<sup>4</sup> is shown in Table 3. SSD provides up-to 80× more efficient random accesses compared to the magnetic hard drives. Therefore, the query processing time for the R-tree and SSD is approximately 3.2× more efficient than in the case of the conventional HDD.

<sup>1</sup> The experiments were executed on an AMD Opteron 865 1.8Ghz, 2.0 MB L2 cache; 2GB of DDR333; Windows 2008 Server.

<sup>2</sup> <http://monetdb.cwi.nl/xml/>

<sup>3</sup> <http://www.census.gov/geo/www/tiger/>

<sup>4</sup> The experiments were executed with Imation S-Class Solid State Drive, 32 GB capacity, Sequential Read: 130MB/s, Sequential Read: IOPS: 83,000.

**Table 2.** XML Collection: Summary table

Technique/Index	Average Results				
	DAC [MB]	Disk accesses	Read pages	Time [s]	Read time [s]
R-tree	8.12	2,256	1	<b>3.75</b>	3.15
Bulk-loaded R-tree	9.62	4,694	1	1.99	1.40
Sequential prefetch	8.12	2,256	1	2.03	1.51
Buf. prefetch (T = 32)	12.61	230	26	1.59	1.15
Buf. prefetch (T = 64)	13.85	133	50	<b>1.49</b>	1.03
Buf. prefetch (T = 128)	15.29	77	97	1.53	1.06
Buf. prefetch (T = 256)	17.48	45	190	1.63	1.17

**Table 3.** XML Collection: Summary table for SSD disk

Technique/Index	Average Results				
	DAC [MB]	Disk accesses	Read pages	Time [s]	Read time [s]
R-tree	8.12	2,256	1	<b>1.17</b>	0.77
Bulk-loaded R-tree	9.62	4,694	1	1.33	0.81
Sequential prefetch	8.12	2,256	1	0.89	0.49
Buf. prefetch (T = 32)	12.61	230	26	0.65	0.27
Buf. prefetch (T = 64)	13.85	133	50	<b>0.66</b>	0.27
Buf. prefetch (T = 128)	15.29	77	97	0.66	0.28
Buf. prefetch (T = 256)	17.48	45	190	0.77	0.31

Our experiments support the proposed assumptions. Sequential prefetch is approximately  $1.9\times$  more efficient than the standard R-tree. Moreover, buffered prefetch is approximately  $2.5\times$  more efficient than the standard R-tree. The bulk-loaded R-tree provides more efficient query processing since it sorts pages during the tree building. It means that a query in the bulk-loaded R-tree is basically processed by sequential prefetch. Although SSD provides more efficient random accesses than conventional disks, we see that our techniques improve the query processing time up-to  $1.8\times$ .

Table 4 presents average results of the range query processing for the TIGER collection. The result sets are significantly bigger than in the case of the XML collection. Therefore, buffered prefetch with higher Threshold turned out as the best one. We can see that our approach is approximately  $3.8\times$  more efficient than the R-tree,  $2.3\times$  more efficient than the bulk-loaded R-tree and it saves up to 42% of query processing time in comparison with skip sequential prefetch.

In the case of queries with a large result set, a large number of leaf nodes is searched. It means the array with node indices may take a lot of memory. Consequently, we can not consider unlimited memory. In Table 5, it is possible to see the influence of the memory limitation on the efficiency of our approach. The average number of pages read by one disk access decreases when the buffer size increases. Therefore, the number of disk accesses and read times are higher. We can see, even if a very small buffer (16 kB) is utilized, our approach saves up to 64% of the query processing time.

Another technique to optimize random accesses has been presented in [20]. Authors propose that their method improves the query processing time up to  $2\times$ ; however, they introduced a modification of the tree build algorithm which affects the build time up to 40%. Our method is more efficient since we apply buffered

**Table 4.** Collection TIGER: Summary table

Technique/Index	Average Results				
	DAC [MB]	Disk accesses	Read pages	Time [s]	Read time [s]
R-tree	19.71	27,424	1	<b>42.07</b>	36.64
Bulkloaded R-tree	28.19	55,260	1	25.03	18.20
Sequential prefetch	19.71	27,424	1	20.02	15.34
Buf. prefetch (T = 32)	39.67	2,705	29	14.87	10.92
Buf. prefetch (T = 64)	42.39	1,484	57	12.82	8.99
Buf. prefetch (T = 128)	45.39	815	111	<b>11.79</b>	8.11
Buf. prefetch (T = 256)	48.75	449	216	13.16	9.27

**Table 5.** Collection TIGER: Memory Limitations

Threshold/Memory	Average Results				
	Disk accesses	Read pages	Number of sorting	Time [s]	Read time [s]
128/Unlimited	815	111	1	11.79	8.11
32/16 kB	5,035	21	10	15.00	11.19
128/64 kB	1,368	100	3	12.91	9.06

prefetch on leaf nodes which occupy 90% of the tree size, whereas the method proposed in [20] is pointed to the top level nodes.

## 5 Conclusion

In this article, we presented an improvement of range query processing in the R\*-tree by an optimization of disk accesses. Our approach avoids frequent random disk accesses. We adopt skip sequential prefetch used in RDBMS. Leaf node indices are sorted before the leaf nodes are read from the secondary storage. Moreover, we enhanced this approach by the read buffer used for the transfer of several nodes into the main memory. In our experiments, we tested range queries and compare the efficiency of the R\*-tree, an R\*-tree with sequential prefetch, and an R\*-tree with buffered prefetch. From DAC point of view, costs are significantly higher in the case of buffered prefetch. However, the query processing time of the R\*-tree with buffered prefetch is up to  $3.8\times$  more efficient than in the case of the R\*-tree and it saves approximately 30% of the query processing time compared to sequential prefetch.

## References

1. Bayer, R.: The Universal B-Tree for Multidimensional Indexing: General Concepts. In: Masuda, T., Tsukamoto, M., Masunaga, Y. (eds.) WWCA 1997. LNCS, vol. 1274. Springer, Heidelberg (1997)
2. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R\*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proceedings of ACM SIGMOD 1990, pp. 322–331 (1990)
3. Berchtold, S., Keim, D.A., Kriegel, H.-P.: The X-Tree: An Index Structure for High-Dimensional Data. In: Proceedings of VLDB 1996, USA., pp. 28–39 (1996)

4. Bhadkamkar, M., Farfan, F., Hristidis, V., Rangaswami, R.: Storing Semi-Structured Data on Disk Drives. *ACM Transactions on Storage* 5(2) (2009)
5. Cha, G.-H.: Index Clustering for High-Performance Sequential Index Access. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) *DASFAA 2004*. LNCS, vol. 2973, pp. 39–51. Springer, Heidelberg (2004)
6. Ciaccia, P., Pattela, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: *Proceedings of VLDB 1997*, Athens, Greece, pp. 426–435 (1997)
7. Fenk, R.: The BUB-Tree. In: *Proceedings of VLDB 2002*, Hongkong, China (2002)
8. Freeston, M.: A General Solution of the  $n$ -dimensional B-tree Problem. In: *Proceedings of SIGMOD International Conference*, San Jose, USA (1995)
9. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: *Proceedings of ACM SIGMOD 1984*, USA, pp. 47–57. ACM Press, New York (1984)
10. Krátký, M., Bača, R., Snášel, V.: On the Efficient Processing Regular Path Expressions of an Enormous Volume of XML Data. In: Wagner, R., Revell, N., Pernul, G. (eds.) *DEXA 2007*. LNCS, vol. 4653, pp. 1–12. Springer, Heidelberg (2007)
11. Krátký, M., Snášel, V., Zezula, P., Pokorný, J.: Efficient Processing of Narrow Range Queries in the R-Tree. In: *Proceedings of IDEAS 2006*. IEEE CS Press, Los Alamitos (2006)
12. Lahdenmäki, T., Leach, M.: *Relational Database Index Design and the Optimizers*. John Wiley and Sons, New Jersey (2005)
13. Arge, L., Hinrichs, K.H., Vahrenhold, J., Vitter, J.S.: Efficient Bulk Operations on Dynamic R-Trees. *Algorithmica* 33(1) (2002)
14. Lightstone, S.S., Teorey, T.J., Nadeau, T.: *Physical Database Design: the Database Professional's Guide*. Morgan Kaufmann, San Francisco (2007)
15. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A.N., Theodoridis, Y.: *R-Trees: Theory and Applications*. Springer, Heidelberg (2005)
16. O'Neil, P.E.: The SB-tree: An Index-Sequential Structure for High-Performance Sequential Access. *Acta Informatica* 29(3) (1992)
17. Riedewald, M., Agrawal, D., Abbadi, A.E., Korn, F.: Accessing Scientific Data: Simpler is Better. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) *SSTD 2003*. LNCS, vol. 2750, pp. 214–232. Springer, Heidelberg (2003)
18. Samet, H.: *Foundations of Multidimensional and Metric Data Structures*, 1st edn. Morgan Kaufmann Series in Computer Graphics (2006)
19. Shasha, D., Bonnet, P.: *Database Tuning: Principles, Experiments, and Troubleshooting Techniques*. Morgan Kaufmann, San Francisco (2002)
20. Skopal, T., Hoksza, D., Pokorný, J.: Construction of Tree-Based Indexes for Level-Contiguous Buffering Support. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) *DASFAA 2007*. LNCS, vol. 4443, pp. 361–373. Springer, Heidelberg (2007)
21. Yu, C. (ed.): *High-Dimensional Indexing*. LNCS, vol. 2341, pp. 9–35. Springer, Heidelberg (2002)

# A Privacy-Enhancing Content-Based Publish/Subscribe System Using Scalar Product Preserving Transformations

Sunoh Choi<sup>1</sup>, Gabriel Ghinita<sup>2</sup>, and Elisa Bertino<sup>2</sup>

<sup>1</sup> Department of Electrical and Computer Engineering, Purdue University, USA  
choi39@purdue.edu

<sup>2</sup> Department of Computer Science, Purdue University, USA  
{gghinita, bertino}@cs.purdue.edu

**Abstract.** Users of content-based publish/subscribe systems (CBPS) are interested in receiving data items with values that satisfy certain conditions. Each user submits a list of subscription specifications to a broker, which routes data items from publishers to users. When a broker receives a notification that contains a value from a publisher, it forwards it only to the subscribers whose requests match the value. However, in many applications, the data published are confidential, and their contents must not be revealed to brokers. Furthermore, a user's subscription may contain sensitive information that must be protected from brokers. Therefore, a difficult challenge arises: how to route publisher data to the appropriate subscribers without the intermediate brokers learning the plain text values of the notifications and subscriptions. To that extent, brokers must be able to perform operations on top of the encrypted contents of subscriptions and notifications. Such operations may be as simple as equality match, but often require more complex operations such as determining inclusion of data in a value interval. Previous work attempted to solve this problem by using one-way data mappings or specialized encryption functions that allow evaluation of conditions on ciphertexts. However, such operations are computationally expensive, and the resulting CBPS lack scalability. As fast dissemination is an important requirement in many applications, we focus on a new data transformation method called Asymmetric Scalar-product Preserving Encryption (ASPE) [1]. We devise methods that build upon ASPE to support private evaluation of several types of conditions. We also suggest techniques for secure aggregation of notifications, supporting functions such as sum, minimum, maximum and count. Our experimental evaluation shows that ASPE-based CBPS incurs 65% less overhead for exact-match filtering and 50% less overhead for range filtering compared to the state-of-the-art.

**Keywords:** Publish/Subscribe Systems, Privacy, Confidentiality, Security.

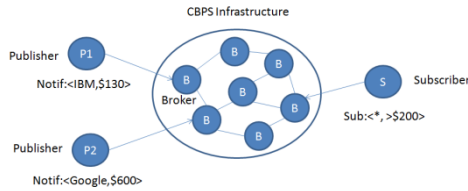
## 1 Introduction

In Content-based Publish/Subscribe Systems (CBPS), data sources (or publishers) disseminate contents to users (or subscribers) using an infrastructure of intermediate

routing entities called *brokers*. Each user creates a subscription that specifies constraints on the data items that s/he is interested to receive. Users register subscriptions with brokers. When a broker receives a notification from a data source, it examines which subscriptions satisfy the notification, and routes the data item to the appropriate users.

A typical application of CBPS is the dissemination of stock market quotes. Several large stock market exchanges act as data publishers, and generate high-rate data streams with prices of individual stocks. Each stock price update, e.g., “*Google, \$600*” represents a notification. Users interested in an individual stock may formulate subscriptions with conditions such as “*Google, ≥ \$550*”, stating that only updates on the stock price of Google that are above \$550 should be sent to the user. Figure 1 illustrates this scenario. When a broker receives a notification, it should be able to determine which subscriptions satisfy the notification and forward it accordingly. In order to do so, a broker has to support various evaluation functions, such as exact match (or equality) filtering, inequality filtering (e.g., “*>*” or “*≤*”), range filtering, etc.

On the other hand, it is not always feasible to deploy a trusted, secure broker infrastructure dedicated to each CBPS application. Instead, an existing content distribution network such as Akamai [6], or an ad-hoc P2P/Grid computing environment may be used, in order to reduce operation and maintenance costs. In this scenario, brokers can no longer be trusted. Therefore, the contents of subscriptions and notifications should no longer be revealed in plaintext to the brokers.



**Fig. 1.** Content-Based Publish/Subscribe System

A non-trusted broker that has access to plaintext subscriptions and notifications may cause significant damage to the CBPS participants. For instance, stock quotes dissemination is typically an expensive service to maintain, and users must pay a subscription fee. If the brokers do not route the data properly, unauthorized users may gain data access, causing financial losses to the publishers. On the other hand, subscription data may also be sensitive. Brokers may collect data subscriptions and infer certain sensitive information: for instance, untrustworthy brokers who notice that a large number of users are interested in a given stock at a particular price may use this knowledge to place unfair market orders at the disadvantage of other participants. In other application areas, subscription contents may disclose private details about users, such as shopping habits, political or religious affiliations, etc.

The conventional method for preventing unauthorized disclosures in a context like CBPS is to apply encryption on the plaintext data and to allow only authorized parties



to perform decryption. Unauthorized parties, such as the provider of the routing infrastructure, are unable to gain access to the plaintext data even though they have access to the ciphertext. However, encryption greatly complicates the process of data filtering. A naïve solution, based on encryption, would require broadcasting all the encrypted data from publishers to all subscribers. But, such a solution is not scalable, as the amount of transferred data is huge, and would quickly congest communication networks. Moreover, a user must have access only to the part of the data s/he has subscribed for. To this end, data items must be encrypted with different keys, and an alternate channel used to send appropriate encryption keys to each user is required. This way, each subscriber can only decrypt the data items s/he is authorized to. Still, managing such a large number of keys may also pose serious scalability concerns.

In order to achieve notification and subscription confidentiality in CBPS, a broker should be allowed to access only encrypted data. Several techniques that address this problem have been proposed previously. For instance, the method in [5] uses a partitioning method and builds an index of encrypted subscriptions. Certain types of conditions can be evaluated on values encrypted in this fashion. However, the method incurs false positives. The solution in [14] uses properties similar to those of homomorphic encryption [21] to evaluate conditions on encrypted data. However, the cost incurred may be large.

In this paper we propose a novel approach to encryption-based access control for CBPS. Our methods support efficient and precise evaluation of conditions based on the ciphertexts of subscriptions and notifications. Our solution relies on Asymmetric Scalar-Product Preserving Encryption (ASPE) [1], a geometric transformation that supports comparisons between pairs of data items. ASPE has been proposed in [1] in the context of evaluating nearest-neighbor queries. We adapt ASPE to support a broad range of conditions evaluations, such as exact match, inequality, range, as well as conjunction of single-attribute conditions. Our specific contributions are:

(i) We propose a novel secure CBPS method for condition evaluations using encrypted values. Our solution has a reduced computational overhead and does not incur false positives.

(ii) We outline a mechanism for secure CBPS aggregation under functions such as sum, minimum, maximum and count.

(iii) We developed a prototype of the proposed method on top of the well-established CBPS system SIENA [4], and we performed an extensive experimental evaluation in comparison with the state-of-the-art in secure CBPS [5]. The experimental results show that our proposed method clearly outperforms existing work in terms of overhead, while offering similar security features.

The remainder of the paper is organized as follows: Section 2 presents background information and surveys related work. Section 3 presents our solution for secure CBPS, whereas Section 4 discusses the case of attacker collusion. In Section 5, we outline a method for secure aggregation in CBPS. Section 6 presents experimental results, and Section 7 concludes with directions for future work.

## 2 Background

### 2.1 CBPS Overview

The main characteristic of CBPS is the loose coupling between publishers and subscribers, which allows for high scalability. The participants in a CBPS system are publishers (or data producers), subscribers (or data consumers), and an infrastructure of brokers that route the data from publishers to subscribers.

Publishers continuously generate data items and forward them to data brokers. A data item, also called a notification, is a set of attributes, where an attribute is a triple: (type, name, value) [4]. After the publisher sends a notification to a broker, it is no longer concerned with the process through which notifications reach subscribers.

A subscriber has an ability to express its interest in a particular data item by generating a subscription. Sometimes subscriptions are also referred to as filters. A subscription is a set of constraints, where a constraint is as follows: (type, name, operator1, value1, operator2, value2). The operators include all the common equality and ordering relations (=, ≠, <, >, etc). In order to express its interest, a subscriber creates a subscription and registers it with a broker. Subsequently, the subscriber does not need to be concerned about how messages will be routed.

Brokers route notification and subscription messages, and *match* notifications against subscriptions. An attribute  $a=(types_a, name_a, value_a)$  matches an attribute constraint  $\phi=(types_\phi, name_\phi, operator1_\phi, value1_\phi, operator2_\phi, value2_\phi)$  iff.  $types_a=types_\phi \wedge name_a=name_\phi \wedge operator1_\phi(value_a, value1_\phi) \wedge operator2_\phi(value_a, value2_\phi)$ . We denote the fact that an attribute  $a$  matches an attribute constraint  $\phi$  by  $a < \phi$ . All such constraints must be matched. Then, we say that a notification  $n$  matches a filter  $f$ . ( $n < f$  for short) if:

$$n < f \Leftrightarrow \forall \phi \in f: \exists a \in n : a < \phi$$

A subscription  $s_1$  is said to *cover* another subscription  $s_2$  when the set of notifications matched by  $s_2$  is a subset of the set of notifications matched by  $s_1$ . Deciding whether a subscription covers another can reduce the amount of match operations performed by brokers, as well as the network traffic overhead.

$$s_2 < s_1 \Leftrightarrow \forall n: n < s_2 \Rightarrow n < s_1$$

There are several kinds of subscriptions. For instance, an equality filter specifies that an exact value must be matched (e.g., “price = 150”). The inequality filter requires a notification to have a value that is greater (conversely, less) than the value of the subscription (e.g., “amount < 300”). Similarly, a range filter specifies a value range (e.g., “100 < price < 200”). In addition, a subscriber may want to receive a notification that satisfies simultaneously several conditions on distinct attributes (e.g., “price < 100 and amount > 300”). We refer to such a subscription as conjunction filter.

We assume that brokers are curious but honest. In the worst case, a malicious broker could simply refuse to forward notifications, staging a denial-of-service attack. Protection against such threats can be achieved if several alternate paths between each publisher and subscriber pair are enforced in the routing infrastructure. However, this

**Table 1.** Examples of  $\prec$ 

Notification	Match	Subscription
<i>integer price = 150</i>	$\prec$	<i>integer price = 150</i>
<i>integer amount = 500</i>	$\nprec$	<i>integer amount &lt; 300</i>
<i>integer price = 150</i>	$\prec$	<i>integer price &gt; 100</i> <i>integer price &lt; 200</i>
<i>integer price = 100</i> <i>integer amount = 500</i>	$\nprec$	<i>integer price &lt; 100</i> <i>integer amount &gt; 300</i>

is outside our scope: we focus on protecting against disclosure of notification and subscription information to brokers. The problem of malicious brokers has been addressed in previous work under some restricted scenarios [10, 12].

## 2.2 Related Work

Our research is related to previous work on outsourced databases (ODB) [8]. In ODB, a data owner (or publisher) stores a database at a service provider (SP) site. Users (i.e., data consumers) send their queries to the SP which processes queries and returns the results to the user. However, the SP is not trusted, therefore the data owner must upload an encrypted version of the database to the service provider. Processing queries on top of encrypted data poses similar challenges to our problem of matching notifications to subscriptions without accessing the plaintext versions of either. In addition, ODB are also concerned with ensuring the completeness (i.e., integrity) of query results. We do not address integrity; however, previous results that address integrity in data streams [20] can be adapted to our scenario.

The Order Preserving Encryption Scheme (OPES) applies an encryption function to an ordinal domain, such that  $E(x) < E(y)$  if  $x < y$  [2]. Thus, OPES converts a distribution of values to another distribution of values, possibly in a distinct domain. So, we can use OPES for inequality filtering and range filtering. However, OPES requires the distribution of the data in advance, whereas data are dynamically generated in CBPS.

On the other hand, we can encrypt a database by splitting the data domain into several partitions and assigning indices to these partitions [5, 7, 19]. In this case, since a service provider may return a superset of the query result, the data owner should perform a post-processing to filter false positives. This can be a problem in CBPS because a subscriber should not be able to receive data that s/he does not have authorization for. Such approaches violate the confidentiality of notifications. The work in [19] assumes that subscribers know the secure index. However, this can give users extra information which may result in compromising confidentiality.

The work in [15] discusses security requirements in CBPS. However, no specific methods for notification and subscription confidentiality are given. Similar to our work, in [16] the brokers are also not trusted. However, it is required that the part of notifications used for routing is not encrypted. Hence, confidentiality is compromised. The work in [18] assumes that the brokers are trusted and are allowed to decrypt a notification for routing. Such an assumption is not reasonable in our model.

The work of Raiciu et al. [5] provides notification and subscription confidentiality for equality, inequality, and range filtering. However, it incurs false positives in inequality filtering and range filtering. Moreover, in order to reduce the ratio of false positives, it has to use more partitions in a domain. Hence, it will take more time for a broker to match a notification against subscriptions. In addition, since a notification should have information about all partitions for range filtering, it will not be practical in a real application. The work in [14] achieves privacy-preserving filtering and covering in CBPS, but it relies on cryptographic elements that incur high computational overhead.

The work in [11] used an additively homomorphic public-key cryptosystem to protect confidential data from intermediate aggregation nodes. The work in [10] verified the integrity of the sum by leveraging a homomorphic MAC scheme based on the discrete logarithm property. However, only secure additive aggregation is considered. We consider in addition methods for sum, min, max and count functions for secure aggregation supporting equality, inequality and range filtering.

### 2.3 Asymmetric Scalar-Product Preserving Encryption (ASPE)

Distance-preserving-transformations (DPT) [13] are an appealing construction to hide the data while allowing at the same time to perform certain operations on top of the ciphertexts. DPT are a subset of the more general Distance Recoverable Encryption (DRE). However, as shown in [1], DRE-based techniques are vulnerable to attacks when the adversary knows some of the plaintext data, or the mapping between some plaintexts and ciphertexts [1, 9]. In our work, we employ a superior transformation, ASPE, which is not distance-recoverable [1].

The weakness of DRE comes from the fact that the attacker is able to recover distance information from the encrypted data [1]. In contrast, ASPE does not reveal distance information. Instead, it only provides means for distance comparison. Given two points  $p_1$  and  $p_2$ , it decides which of the two points is nearer to a query point  $q$ .

$$\frac{D(p_1, q) \geq D(p_2, q)}{\sqrt{\|p_1\|^2 - 2p_1 \cdot q + \|q\|^2} \geq \sqrt{\|p_2\|^2 - 2p_2 \cdot q + \|q\|^2}} \\ \|p_1\|^2 - \|p_2\|^2 + 2(p_2 - p_1) \cdot q \geq 0$$

The inequality is decomposed to a number of scalar product computations. There are three types of scalar products: (type-1) scalar product of a data point with itself, (type-2) scalar product of a data point with the query point, and (type-3) scalar product of two different data points  $p_1$  and  $p_2$ . If an encryption preserves only type-1 and type-2 products but not type-3 products, which is essential in DRE, then we can compare the distances  $d(p_1, q)$  and  $d(p_2, q)$  by the above equation without the vulnerabilities of DRE.

The scalar product of  $p$  and  $q$  can be represented as  $p^T I q$ .  $I$  is the unit matrix, and can be decomposed to  $MM^T$  for any invertible matrix  $M$ . If we set  $p' = M^T p$  and  $q' = M^{-1} q$ , it is not computationally feasible for an adversary to determine the value of  $p$  and  $q$  from  $p'$  and  $q'$  without knowing  $M$ . Also,  $p'^T q' = p^T M M^{-1} q = p^T q$ , i.e., type-2 scalar product is preserved. But,  $p_1'^T p_2' = p_1^T M M^T p_2$  is not equal to  $p_1^T p_2$  in general. The Asymmetric Scalar-product Preserving Encryption (ASPE) uses  $M$  and  $M^{-1}$  as the transformations for data points and queries respectively.

If the type-1 product  $\|p\|^2$  is revealed to the attacker, he knows that  $p$  lies on a hypersphere that is centered at the origin with a radius  $\|p\|$ . It can partially compromise security. We need to hide this information by hiding the value of  $\|p\|^2$ . That is, the value of  $-0.5\|p_1\|^2$  is treated as the  $(d+1)$ -st dimension of the point  $p$  since it will be used to get the distance difference. Given a  $d$ -dimensional data point  $p$ , a  $(d+1)$ -dimensional point  $p^*$  is created. Similarly, we need to extend a query  $q$  to a  $(d+1)$ -dimensional point  $q^*$ . The simplest way is to set the  $(d+1)$ -st dimension of  $q^*$  to 1. For security, we generate a random number  $r > 0$  and scale  $q^*$  by  $r$ .

Let  $p'_1, p'_2$ , and  $q'$  be the encrypted points of the data points  $p_1, p_2$  and the query point  $q$ . We can determine whether  $p_1$  is closer to  $q$  than  $p_2$  by evaluating whether  $(p'_1 - p'_2) \cdot q' > 0$ . Note that

$$\begin{aligned} (p'_1 - p'_2) \cdot q' &= (p'_1 - p'_2)^T q' = (M^T p_1^* - M^T p_2^*)^T M^{-1} q^* = (p_1^* - p_2^*)^T q^* \\ &= (p_1 - p_2)^T (rq) + (-0.5\|p_1\|^2 + 0.5\|p_2\|^2)r \\ &= 0.5r(\|p_2\|^2 - \|p_1\|^2 + 2(p_1 - p_2)^T q) \\ &= 0.5r(D(p_2, q) - D(p_1, q)) \end{aligned}$$

So, the condition is equivalent to

$$0.5r(D(p_2, q) - D(p_1, q)) > 0 \Leftrightarrow D(p_2, q) > D(p_1, q).$$

### 3 Secure CBPS Using ASPE

By using ASPE, we can compare the distance between a data point  $p_1$  and a query point  $q$  with the distance between another data point  $p_2$  and the same query point  $q$ . In this paper, we propose a CBPS framework for matching subscriptions and notifications encrypted with an ASPE-like technique. In the CBPS setting, the data point  $p$  corresponds to a notification, and the query point  $q$  to a subscription. Specifically, for each subscription the users generate a set of reference values encrypted according to transformation  $M$ , whereas the publishers transform notifications according to transformation  $M^{-1}$ .

There are several kinds of subscription conditions in CBPS. 1) In the equality filtering, a subscriber wants to receive a notification which is equal in value to its subscription. 2) In the inequality filtering, a subscriber wants to receive a notification which is larger or less than its subscription. 3) In the range filtering, a subscriber wants to get a notification which is in a certain range. 4) In the conjunction filtering, a subscription consists of several conditions expressed as multiple attribute constraints and a subscriber wants to receive a notification which satisfies all the conditions.

In addition, the *covering* operation allows a broker to determine if, given two subscriptions, one of them will always match a strict subset of the notifications matched by the other. Next, we show how ASPE can be used to support all the above operations while maintaining confidentiality of subscriptions and notifications.

#### 3.1 Equality Filtering

Consider for example that a subscriber wants to receive a notification which is equal to some value  $a$ . The subscriber sends to a broker the encrypted subscription  $E(a) = a'$ .

When a publisher wants to disseminate a data item with value  $x$ , it sends to a broker an encrypted notification  $E(x) = x'$ . When a broker receives  $a'$  and  $x'$ , it must be able to determine whether  $x$  is equal to  $a$ . But, the broker should not learn what are the values of  $x$  and  $a$ .

Our secure matching method works as shown in Fig 2. The subscriber selects  $c$  and  $d$  where  $c = a - s$  and  $d = a + s$ , i.e.,  $a$  is the middle of the interval  $[c, d]$  ( $s > 0$  is a random number) and sends  $c' = M^T c^* = M^T(c, -0.5\|c\|^2)^T$  and  $d' = M^T d^* = M^T(d, -0.5\|d\|^2)^T$  as a subscription to the broker. On the other hand, the publisher sends a notification  $x' = M^{-1}x^* = M^{-1}r(x, 1)^T$ .

As explained in Section 2.3, a broker can evaluate based on the encrypted values which one of  $distance(c, x)$  and  $distance(d, x)$  is larger.

$$\begin{aligned}
(c' - d') \cdot x' &= (c' - d')^T x' = (M^T c^* - M^T d^*)^T M^{-1} x^* \\
&= (c^* - d^*)^T x^* \\
&= (c - d)^T (rx) + (-0.5\|c\|^2 + 0.5\|d\|^2)r \\
&= 0.5r(\|d\|^2 - \|c\|^2 + 2(c - d)^T x) \\
&= 0.5r(D(d, x) - D(c, x))
\end{aligned} \tag{1}$$

If the difference is equal to 0, the broker can determine that  $x$  is equal to  $a$ . Otherwise, the values must be different. However, since the broker receives  $x', c'$ , and  $d'$  instead of  $x$  and  $a$ , it can not know what  $x$  and  $a$  are.

**Theorem 1.** The equality matching using ASPE is secure.

Proof. Suppose that a broker wants to compute  $a$  using  $x', c'$ , and  $d'$  when  $x$  is equal to  $a$ . If the broker can get  $(c + d)^T x = 2\|a\|^2$ , since  $c + d$  is equal to  $2a$ , the equality filtering is not secure. We show that the broker cannot obtain  $(c + d)^T x$  from  $(c' + d')^T x'$ .

$$\begin{aligned}
(c' + d')^T x' &= (M^T c^* + M^T d^*)^T M^{-1} x^* \\
&= (c^* + d^*)^T x^* \\
&= (c + d)^T (rx) + (-0.5\|c\|^2 - 0.5\|d\|^2)r \\
&= 0.5r(2(c + d)^T x - \|c\|^2 - \|d\|^2) \\
&\neq (c + d)^T x
\end{aligned} \tag{2}$$

Since the  $(d+1)$ -st dimensions of  $c^*$  and  $d^*$  are  $-0.5\|c\|^2$  and  $-0.5\|d\|^2$  respectively, and there is a random number  $r$ , the broker can't get  $(c + d)^T x$  by computing  $(c' + d')^T x'$  when  $x$  is equal to  $a$ . Therefore, the equality test using ASPE is secure.

### 3.2 Inequality Filtering

In the inequality filtering, the broker should be able to determine whether the notification value  $x$  is greater than subscription value  $a$ . As in the case of equality filtering using ASPE, the subscriber selects  $c$  and  $d$  where  $c = a - s$  and  $d = a + s$  ( $s > 0$  is a random number) and sends  $c' = M^T c^* = M^T(c, -0.5\|c\|^2)^T$  and  $d' = M^T d^* = M^T(d, -0.5\|d\|^2)^T$  as a subscription to the broker. The publisher sends a notification  $x' = M^{-1}x^* = M^{-1}r(x, 1)^T$ . At that time, the broker can compute a

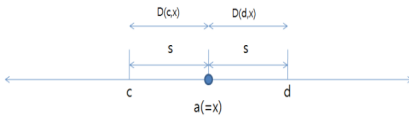


Fig. 2. Equality Filtering using ASPE

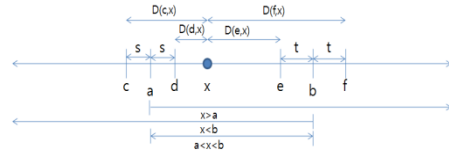


Fig. 3. Range Filtering using ASPE

difference between  $distance(c,x)$  and  $distance(d,x)$ . If the difference is greater than 0, the broker can know that  $x$  is greater than  $a$ . But, the broker can't know what the values of  $x$  and  $a$  are. The security proof is similar to the proof of Theorem 1.

$$D(c, x) - D(d, x) > 0 \Leftrightarrow (d' - c') \cdot x' > 0 \tag{3}$$

### 3.3 Range Filtering

When a subscriber wants to receive a value  $x$  between  $a$  and  $b$ , s/he sends to the broker two pairs of values. The first pair  $c' = M^T c^* = M^T(c, -0.5\|c\|^2)^T$  and  $d' = M^T d^* = M^T(d, -0.5\|d\|^2)^T$  encodes the value of  $a'$ , whereas the second pair  $e' = M^T e^* = M^T(e, -0.5\|e\|^2)^T$  and  $f' = M^T f^* = M^T(f, -0.5\|f\|^2)^T$  encodes the value of  $b'$ , where  $e = b - t$  and  $f = b + t$  ( $t > 0$  is a random number), as shown in Figure 3. By using  $x', c', d', e', f'$ , the broker can compute a difference  $p$  between  $distance(c,x)$  and  $distance(d,x)$  and a difference  $q$  between  $distance(e,x)$  and  $distance(f,x)$ . If  $p$  is greater than 0,  $x$  is greater than  $a$  and if  $q$  is less than 0,  $x$  is less than  $b$ . Therefore, the broker can determine whether  $x$  is between  $a$  and  $b$  as follows:

$$\begin{aligned} D(c, x) - D(d, x) > 0 \ \&\& \ D(e, x) - D(f, x) < 0 \\ \Leftrightarrow (d' - c') \cdot x' > 0 \ \&\& \ (f' - e') \cdot x' < 0 \end{aligned} \tag{4}$$

In [2], it is suggested that the Order Preserving Encryption Scheme (OPES) can be used for inequality filtering and range filtering on encrypted data. However, OPES requires to know the data distribution in advance. But, in CBPS, since data are dynamically generated, it may be difficult to know the data distribution, especially when there can be several publishers. So, OPES is not suitable for CBPS. In [3], the ciphertext size of range filtering is  $O(\sqrt{n})$ . The value  $n$  is the number of data points in a domain. Such an approach would not scale well. Note that, the space complexity of range filtering using ASPE is  $O(1)$ .

### 3.4 Covering

Suppose that there are two subscribers. When the first subscriber wants to get a value greater than  $a$  and the second subscriber wants to receive a value greater than  $b$ , the broker should be able to determine that the set of notifications matched by one of the subscriptions will always be a subset of the notifications matched by the other. For instance, if  $a$  is less than  $b$ , the broker can send only (the encrypted)  $a$  to a parent broker. Also, when a parent broker evaluates the matching condition, it may only

need to evaluate a single condition rather than two, saving computation cost. Therefore, the covering operation helps to achieve scalability in CBPS.

Our idea is illustrated in Figure 4. Each subscriber selects a random number. Thus, the first subscriber chooses a random number  $y$  ( $y > a + s = d$ ) and the second subscriber picks up a random number  $z$  ( $z > b + t = f$ ). The first subscriber sends to the broker the triplet  $c' = M^T c^* = M^T(c, -0.5\|c\|^2)^T$ ,  $d' = M^T d^* = M^T(d, -0.5\|d\|^2)^T$ ,  $y' = M^{-1}y^* = M^{-1}(y, 1)^T u$  (where  $u$  is also a random number) to encode the value of  $a'$  and the second subscriber sends  $e' = M^T e^* = M^T(e, -0.5\|e\|^2)^T$ ,  $f' = M^T f^* = M^T(f, -0.5\|f\|^2)^T$ ,  $z' = M^{-1}z^* = M^{-1}(z, 1)^T v$  (where  $v$  is a random number) to encode the value of  $b'$ .

By computing the difference between  $distance(e, y)$  and  $distance(f, y)$ , if the difference is greater than 0, the broker can determine whether  $y$  is greater than  $b$ . If  $y$  is less than  $b$ , the broker can know that  $b$  is greater than  $a$  since  $y$  is greater than  $a + s (= d)$ .

If  $y$  is greater than  $b$  and condition (5) is satisfied, the broker can determine that  $b$  is greater than  $a$ . However, the broker can't know what  $a$  and  $b$  are. On the other hand, in condition (5), we can use  $z'$  instead of  $y'$ .

$$\begin{aligned}
 D(a, y) > D(b, y) &\Leftrightarrow D(c, y) + D(d, y) > D(e, y) + D(f, y) \\
 &\Leftrightarrow (D(c, y) - D(e, y)) + (D(d, y) - D(f, y)) > 0 \\
 &\Leftrightarrow (e' - c') \cdot y' + (f' - d') \cdot y' > 0
 \end{aligned}
 \tag{5}$$

For simplicity, the condition (5) and Figure 4 show covering for the inequality filtering. We can extend this easily to covering for range filtering. In this case, a subscriber sends  $c', d', i', j', y'$  in order to get the values between  $a$  and  $g$  and the other subscriber sends  $e', f', k', l', z'$  in order to get the values between  $b$  and  $h$ . If condition (6) is satisfied, we can say that the first subscription covers the second subscription. Thus, if  $x$  is between  $b$  and  $h$ , then  $x$  is between  $a$  and  $g$  (e.g.,  $b < x < h \rightarrow a < x < g$ ).

$$\begin{aligned}
 D(a, y) > D(b, y) \ \&\& \ D(g, y) < D(h, y) \\
 &\Leftrightarrow D(c, y) + D(d, y) > D(e, y) + D(f, y) \\
 &\ \&\& \ D(i, y) + D(j, y) < D(k, y) + D(l, y) \\
 &\Leftrightarrow (D(c, y) - D(e, y)) + (D(d, y) - D(f, y)) > 0 \\
 &\ \&\& \ (D(i, y) - D(k, y)) + (D(j, y) - D(l, y)) > 0 \\
 &\Leftrightarrow (e' - c') \cdot y' + (f' - d') \cdot y' > 0 \\
 &\ \&\& \ (k' - i') \cdot y' + (l' - j') \cdot y' > 0
 \end{aligned}
 \tag{6}$$

### 3.5 Conjunction Filtering

Suppose that a publisher sends a value  $m$  and a value  $n$  for two distinct attributes in a single notification. In this case, the publisher sends a column vector  $X' = M^{-1}X = M^{-1}r(X, 1)^T$  to the broker(s). The column vector  $X$  contains the original values  $m$



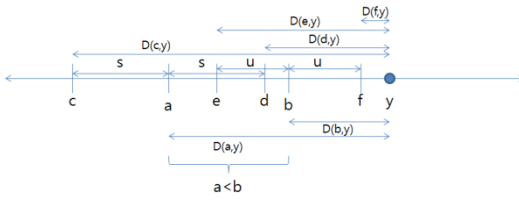


Fig. 4. Covering for Range Query using ASPE

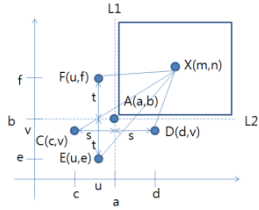


Fig. 5. Conjunctive Query using ASPE

and  $n$ . Suppose that a subscriber wants to receive a notification in which the value of the first attribute is greater than  $a$  and the value of the second attribute is greater than  $b$ . Then, the subscriber sends  $C' = M^T C^* = M^T(C, -0.5\|C\|^2)^T$ ,  $D' = M^T D^* = M^T(D, -0.5\|D\|^2)^T$ ,  $E' = M^T E^* = M^T(E, -0.5\|E\|^2)^T$ ,  $F' = M^T F^* = M^T(F, -0.5\|F\|^2)^T$  as a subscription instead of  $A' = M^T A^* = M^T(A, -0.5\|A\|^2)^T$ .  $A$  is a column vector which contains the values  $a$  and  $b$ . As shown in Figure 5,  $C$  and  $D$  have the same distance from the line  $L1$ .  $E$  and  $F$  have the same distance from the line  $L2$ . If a condition (7) is satisfied, the broker can determine whether  $m$  is greater than  $a$  and  $n$  is greater than  $b$ . Thus, the broker can know that  $X$  is inside the rectangle specified by coordinates  $L1$  and  $L2$ . But, the broker can't know what  $a$ ,  $b$ ,  $m$ , and  $n$  are. We can extend this method to a hyper-rectangle which has multiple attributes to handle multi-dimensional spaces.

$$\begin{aligned}
 &D(C, X) > D(D, X) \ \&\& \ D(E, X) > D(F, X) \\
 \Leftrightarrow &(D' - C') \cdot X' > 0 \ \&\& \ (F' - E') \cdot X' > 0
 \end{aligned}
 \tag{7}$$

In [3,17], the complexity of conjunctive condition evaluations is  $O(nd)$ . The value  $n$  is the number of the data points and the value  $d$  is the number of dimensions. In contrast, the complexity of conjunctive conditions using ASPE is only  $O(d)$ .

### 4 Preventing Collusion Attacks

In ASPE, a subscriber knows the matrix  $M$ . So, if the subscriber is malicious, the system may not be secure since the broker can decode the notification using the matrix learnt from the subscriber. In order to solve this problem, we can employ a trusted security manager component. Instead of creating their own encrypted subscriptions, users send plaintext subscriptions to the security manager. For example, the subscriber sends an original subscription  $x$  to the security manager, and the latter assembles the encrypted subscription.  $y' = M^T y^* = M^T(y, -0.5\|y\|^2)^T$ ,  $z' = M^T z^* = M^T(z, -0.5\|z\|^2)^T$  which can be then sent to a broker. This way, the subscriber does not learn the transformation matrix. The security manager architecture is depicted in Figure 6.

Since the subscriber does not know the matrix, it can't collude with a broker. Also, the subscriber can't get the matrix from the original subscription  $x$  and the encrypted subscription  $y'$  and  $z'$  by Theorem 1. Even if the subscriber can get the value  $a$  from the notification  $a'$  by decoding it using a secret key, since the value  $a$  is not a column vector but an integer value, the subscriber can't learn the matrix.

Moreover, if subscribers want to send too many subscriptions, it can make the CBPS system insecure. If a subscriber or a broker knows all the subscriptions, it may attempt to guess the matrix used for ASPE. In this case, the security manager may restrict the rate of subscriptions in order to protect against this sort of attacks.

In addition, we can use an access control method at the security manager. A subscriber should send a plaintext subscription to the security manager through a secure channel. If the subscriber has an access right, it can receive the encrypted subscription from a broker by forwarding the encrypted subscription.

When a subscriber leaves its subscribing group, the security manager makes a publisher change the matrix which is used for notification encryption or sends a new matrix to the publisher such that the subscriber can't receive notifications any more. The notification made by the new matrix can't be filtered by the prior subscription. So, a subscriber which lost an access right can't receive a notification.

Also, the broker can manage the subscriptions using a soft-state method. Thus, it enables access control by deleting a subscription which expired. In order to prevent the subscription from expiring, a subscriber should contact the security manager to renew an access right and forward a new encrypted subscription to a broker.

Changing the matrix whenever a subscriber leaves the system can be a burden to a secure CBPS. If subscribers do not leave frequently, changing the matrix is reasonable. Also, using soft-state in the brokers can reduce the frequency of changing the matrix.

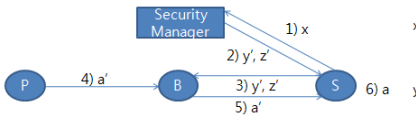


Fig. 6. Preventing collusion

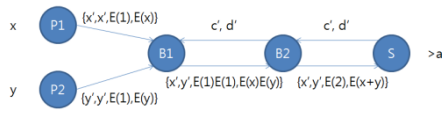


Fig. 7. Secure Aggregation using ASPE

## 5 Secure Aggregation Using ASPE

In this section, we suggest a method for secure aggregation in CBPS using ASPE in conjunction with homomorphic encryption. Specifically, we employ an additively homomorphic public-key cryptosystem for sum, such as the one in [21]. When there is a notification whose value is  $a$  and another notification whose value is  $b$ , two publishers can send the encrypted notification  $E(a)$  and  $E(b)$  by using the homomorphic encryption function. Then a broker computes the sum of the two values as follows:  $E(a + b) = E(a) \cdot E(b)$

Note that, this encryption is applied to the notification content. The part of the notification that is used for routing will also be encrypted with the ASPE function, and this section of the notification is used by the brokers to make forwarding decisions. The format of the notification is the following:

$$Notification = \langle min, max, count, sum \rangle$$

In order to do range filtering, a notification has an encrypted minimum value and an encrypted maximum value by using ASPE. For additive aggregation, it has a sum by

using homomorphic function. For example, a notification whose value is  $x$  is represented as  $\langle x', x', E(1), E(x) \rangle$ . Since we used ASPE,  $x'$  is equal to  $M^T x^* = M^T r(x, 1)^T$ . Because the notification is a sum of only one notification, its count value is equal to  $E(1)$ . Figure 7 illustrates this process.

When a broker receives two notifications  $x'$  and  $y'$ , if a subscriber wants to get a sum of notifications whose values are greater than  $a$ , the broker should check whether  $x$  and  $y$  are greater than  $a$ . As we mentioned in Section 3.2, we can use the inequality filtering method based on ASPE. The following condition should be examined where  $c = a - s$  and  $d = a + s$ . ( $s$  is a random number.) First, if the following condition (8) is satisfied, we can verify whether  $x$  and  $y$  are greater than  $a$ . Then, the sum of the two notifications is computed as  $E(x + y) = E(x) \cdot E(y)$ .

$$\begin{aligned} D(c, x) - D(d, x) > 0 \ \&\& \ D(c, y) - D(d, y) > 0 \\ \Leftrightarrow (d' - c') \cdot x' > 0 \ \&\& \ (d' - c') \cdot y' > 0 \end{aligned} \quad (8)$$

Second, we have to determine which one is greater between the two values in order to get the minimum value and the maximum value. If the condition (9) is satisfied, we can know that  $x$  is less than  $y$ . Because  $x$  and  $y$  are greater than  $a$  in the first step,  $x$  and  $y$  are also greater than  $c$ .

$$D(x, c) < D(y, c) \Leftrightarrow (y' - x') \cdot c' < 0 \quad (9)$$

If we assume that  $x$  is less than  $y$ , the minimum value is  $x$  and the maximum value is  $y$ . So, the broker produces a new notification which has the following values.

$$\langle \min, \max, \text{count}, \text{sum} \rangle = \langle x', y', E(2), E(x + y) \rangle$$

## 6 Experimental Evaluation

We implemented the proposed secure CBPS methods using ASPE in SIENA[4], a wide-area event notification service. SIENA is a content-based publish/subscribe infrastructure where brokers are vertices in a connected overlay acyclic graph. However, SIENA does not provide any security features. We evaluate the performance of our method in comparison with the C-CBPS solution proposed in [5].

As discussed in Section 2.2, C-CBPS supports secure equality filtering and range filtering using a re-mapping of subscriptions and notifications. It adapts a scheme from Song et al. [22] to support equality filtering and defines two schemes for inequality filtering and range filtering.

For equality filtering, C-CBPS computes the hidden value of an attribute by passing its plaintext value to a pseudorandom function with the secret key. The encrypted subscription is the hidden value of the plaintext. The encrypted notification has a random nonce  $r$  and the result of feeding the nonce  $r$  to a pseudorandom function. When the broker computes the value of feeding the nonce  $r$  to a pseudorandom function with the encrypted subscription, if the value is the same with the result contained in the notification, the notification satisfies the subscription.

For inequality filtering, C-CBPS chooses  $l$  points,  $p_1, \dots, p_l$  as reference points and considers the following dictionary:  $\{>p_1, >p_2, \dots, >p_l, <p_1, <p_2, \dots, <p_l\}$ . Subscriptions will be approximated with one of these constraints. Each

notification is considered to be a document containing the words in the dictionary. For range filtering, in order to support  $l_b < N < u_b$  subscriptions, C-CBPS has the publishers and subscribers agree on a partitioning  $P = \{p_1, \dots, p_l\}$ . The publisher encrypts the index of the subset which  $N$  belongs to by using equality filtering. The subscribers include as subscriptions encrypted versions of the indexes of the subsets in the partition they are interested in (i.e. all  $p_i \in P$  such that  $p_i \cap (l_b, u_b) \neq \emptyset$ ).

The security features of our method and C-CBPS are similar. However, our results show that the overhead incurred at the brokers by our method is clearly superior to the work in [5].

### 6.1 Evaluation Methodology

All the data used for testing are generated uniformly at random. In all the tests, a single instance of the enhanced SIENA matching engine was evaluated. All experiments were run on a 2.1Ghz Intel Core2 Duo CPU with 3GB of RAM running Windows Vista and Sun’s JDK 1.6. Time is measured by using the function *System.nanoTime()*.

Matching time is measured as the time that the broker spends to identify the set of matching subscribers, when a notification is given. We measure the average matching time required to match a notification against 1000 subscriptions. Subscription and notification sizes are measured by total network bytes sent including SIENA’s protocol overhead. We consider types of subscriptions which filter numeric attributes using arithmetic operators (i.e., =, <, >). We use the schemes with subscription covering enabled.

### 6.2 Matching Time Measurements

Notifications are integers generated uniformly at random from [0,10000]. To test *equality filtering*, we select subscriptions uniformly at random from [0,10000]. Figure 8(a) shows the results for equality filtering time. Our method using ASPE shows the similar performance with SIENA which uses the plaintext subscriptions and notifications. On the other hand, ASPE takes about 65% less time than C-CBPS for equality filtering. ASPE has a reference matching time of 2.7ms.

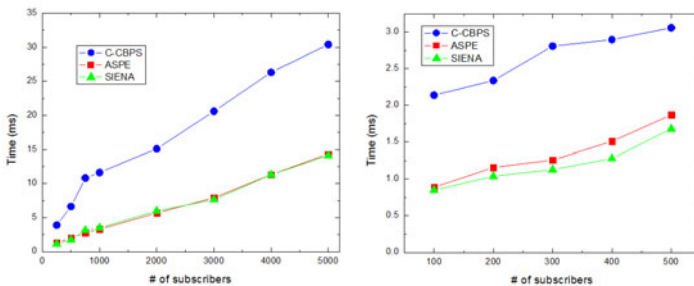


Fig. 8. (a) Equality filtering time (b) Range filtering time

Figure 8(b) shows the result for *range filtering*. ASPE takes about 50% less time than C-CBPS and shows similar performance with SIENA. C-CBPS uses subsets for range filtering. For example, each subset can have a range as follows: (0,50), (10,60), (20,70), etc. It used overlapped partitions for subscription confidentiality. Originally C-CBPS used 890 subsets which have values from [0,1000]. So, it had a false matching rate of 5%. There are about 1000000 subsets which have values from [0,10000]. A partitioning scheme with zero false matches for any range subscription has  $|D|^2$  points, being quite expensive. So, it is not reasonable to use only 890 subsets. In order to compare the performance of C-CBPS with ASPE more precisely, we made C-CBPS use about 15000 subsets which have values from [0,10000]. However, it still has false positives.

Also, in C-CBPS, as there are more subsets, the size of a subscription is larger because the subscription should contain the information about the subsets when covering is enabled. When there are 890 subsets, the size of a subscription is 775 bytes. When we use about 15000 subsets, the size is 10728 bytes. So, when there are more than 500 subscribers in C-CBPS, the system runs out of memory. ASPE is scalable and does not incur false positives.

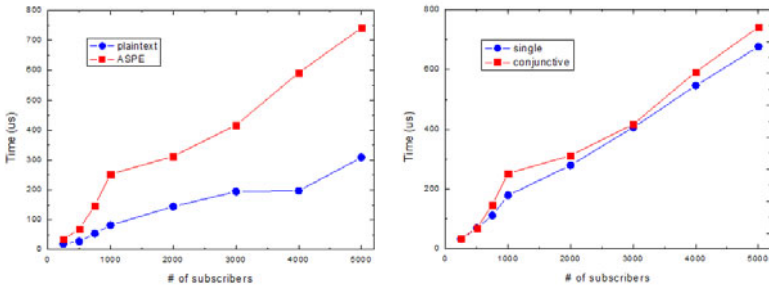


Fig. 9. (a) Conjunction filtering time (b) Single vs conjunctive condition filtering

Since SIENA and the C-CBPS do not support the conjunction filtering, we compared ASPE with plaintext filtering. Figure 9(a) shows the performance of ASPE which runs the conjunction filtering having two attributes. ASPE is about 2.5 times as expensive compared to plaintext filtering. But, since the reference matching time of ASPE for the conjunctive query is 0.25ms, it is still reasonable in practice.

Finally, Figure 9(b) gives the performance of the single condition filtering and conjunctive condition filtering using ASPE. Note that, the conjunctive condition filtering takes only about 10% more time than the single condition filtering. It shows that the conjunction filtering using ASPE is practical and scalable.

## 7 Conclusion

In this paper, we proposed a secure CBPS system based on Asymmetric Scalar-product Preserving Encryption in order to provide notification and subscription confidentiality and to reduce matching complexity. Our methods support equality filtering, inequality filtering, range filtering, covering, and conjunction filtering which

are essential in CBPS. In addition, our solution does not incur false positives, in contrast to existing work such as C-CBPS. Moreover, we suggested a new method for secure aggregation using ASPE and homomorphic functions. We can support sum, min, max and count functions for equality, inequality, and range filtering.

The experiment results show that our methods take about 65% less time in equality filtering and about 50% less time in range filtering than C-CBPS. Moreover, our secure conjunction filtering method incurs reasonable overhead when compared to plaintext conjunction filtering and single condition filtering. In future work, we intend to develop secure mechanisms for other types of subscriptions with more complex conditions.

## Acknowledgments

The work reported in this paper has been partially supported by MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

## References

1. Wong, W.K., Cheung, D.W., Kao, B., Mamoulis, N.: Secure kNN Computation on Encrypted Databases. In: ACM SIGMOD International Conference on Management of Data, pp. 139–152 (2009)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order Preserving Encryption for Numeric Data. In: ACM SIGMOD International Conference on Management of Data, pp. 563–574 (2004)
3. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Theory of Cryptography Conference, pp. 535–554 (2007)
4. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems* 19(3), 332–383 (2001)
5. Raiciu, C., Rosenblum, D.S.: Enabling Confidentiality in Content-Based Publish/Subscribe Infrastructures. In: International Conference on Security and Privacy in Communication Networks, pp. 1–11 (2006)
6. Akamai, <http://www.akamai.com>
7. Hacigumus, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: ACM SIGMOD International Conference on Management of Data, pp. 216–227 (2002)
8. Hacigumus, H., Iyer, B., Mehrotra, S.: Providing database as a service. In: ICDE International Conference on Data Engineering, pp. 29–38 (2002)
9. Liu, K., Giannella, C., Kargupta, H.: An attacker's view of distance preserving maps for privacy preserving data mining. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 297–308. Springer, Heidelberg (2006)
10. Minami, K., Lee, A.J., Winslett, M., Borisov, N.: Secure Aggregation in a Publish-Subscribe System. In: ACM Workshop on Privacy in the Electronic Society, pp. 95–104 (2008)
11. Ahmed, W., Khokhar, A.: Secure aggregation in large scale overlay networks. In: Global Telecommunications Conference, pp. 1–5 (2006)
12. Srivatsa, M., Liu, L.: Securing Publish-Subscribe Overlay Services with EventGuard. In: ACM Conference on Computer and Communications Security, pp. 289–298 (2005)

13. Oliveira, S.R.M., Zaiane, O.R.: Privacy preserving clustering by data transformation. In: 18th Brazillian Symposium on Databases, pp. 304–318 (2003)
14. Nabeel, M., Shang, N., Bertino, E.: Privacy-Preserving Filtering and Covering in Content-Based Publish Subscribe Systems. CERIAS Tech. Report 2009-15, Purdue University, West Lafayette, IN
15. Wang, C., Carzaniga, A., Evans, D., Wolf, A.L.: Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems. In: Hawaii International Conference on System Sciences, pp. 303–310 (2002)
16. Khurana, H.: Scalable Security and Accounting Services for Content-based Publish/Subscribe Systems. In: ACM Symposium on Applied Computing, pp. 801–807 (2005)
17. Shi, E., Bethenourt, J., Hubert Chan, T.-H., Song, D., Perrig, A.: Multi-Dimensional Range Query over Encrypted Data. In: IEEE Symposium on Security and Privacy, pp. 350–364 (2007)
18. Pesonen, L.I.W., Evers, D.M., Bacon, J.: Encryption-Enforced Access Control in Dynamic Multi-Domain Publish/Subscribe Networks. In: International Conference on Distributed Event-Based Systems, pp. 104–115 (2007)
19. Hore, B., Mehrotra, S., Tsudik, G.: A Privacy-Preserving Index for Range Queries. In: International Conference on Very Large Data Bases, pp. 720–731 (2004)
20. Papadopoulos, et al.: Continuous Authentication on Data Streams. In: International Conference on Very Large Data Bases, pp. 135–146 (2007)
21. Paillier, P.: Public-key cryptosystem based on composite degree residuosity classes. In: Advances in Cryptology, pp. 223–238 (1999)
22. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy, pp. 44–55 (2000)

# Synthesizing: Art of Anonymization

Jun Gu, Yuexian Chen, Junning Fu, Huanchun Peng, and Xiaojun Ye

School of Software, Tsinghua University  
j-gu07@mails.tsinghua.edu.cn, yexj@mail.tsinghua.edu.cn

**Abstract.** Although there are a number of anonymization techniques in the microdata publication, two problems remain: (1) the privacy breaches with auxiliary knowledge; (2) the large information losses during the anonymization. We establish the requirement of *presence anonymity* and propose the two-step process of *synthesizing*, consisting of learning a model from the original data, and then sampling a published version with it, which has the similar statistical characteristics and includes fake records. The advantage is that it prevents the auxiliary knowledge attacks as well as enables researchers get correct or approximately correct conclusions. Furthermore, its effectiveness is proved through extensive experiments.

**Keywords:** privacy, anonymization, synthesizing.

## 1 Introduction

The privacy preserving has received considerable attentions from the database, statistics and cryptograph communities in the past few years. Specifically, the microdata  $T$  is a table containing useful information about a group of individuals. Each record in the table corresponds to a unique individual and contains both some sensitive attributes(e.g., disease, salary) and non-sensitive ones(e.g., address, gender, age). The objective of microdata publication is to release a sanitized version  $T^*$  of  $T$  such that forbids potential adversaries from inferring the sensitive values of any individuals from it; at the same time allows legal researchers to reach useful conclusions.

The identifying attributes such as name and social security number are removed before publication. This simple method, however, is insufficient due to the possibility of linking attacks[1]. It is easy to implement the complete privacy by not publishing both quasi-identifier attributes and sensitive attributes. This trivial sanitization provides the maximum privacy, while merely leaves any utility to the legal researchers. Conclusions such as “people who smokes a lot are easy to get the lung cancer” are impossible to be deduced from the trivial sanitized table. Some research in database community applies the domain-specific generalization and suppression to quasi-identifier attributes. These methods are guided by the anonymization principle of  $k$ -anonymity[2], which defines constrains on the quasi-identifier attributes. Another permutation-based method called anatomy[3] clusters records into buckets and publishes quasi-identifiers and sensitive attributes separately. This method follows the privacy



schema of  $\ell$ -diversity [3]. Koudas etc. propose the “distribution-based microdata anonymization” [4]. The authors combine sensitive attribute permutation and generalization with the addition of fake sensitive attribute values. In this approach, the requirement that groups of sensitive attribute values follow specified distributions, is defined in  $t$ -closeness [5].

## 1.1 Background

The challenge of microdata publication is how to enable researchers reach correct conclusions without corrupting the privacy of any individuals in the data.

**Attack with auxiliary knowledge.** In 2002, Sweeney reports that 87% of the population of the United States could be uniquely identified by the seemingly innocuous attributes: gender, date of birth and 5-digit zip code [1]. Such attributes are called the quasi-identifier(QI) attributes. It is called the linking attack that linking the published records with individuals by QI values. Since  $k$ -anonymity, it has always been the focus of research on privacy to resist such linking attacks.

Apart from QI values, more information can be linked with published records, which makes the problem more complicated. Now the greatest threat to microdata is the auxiliary information(also called background knowledge) that users collect from other sources. Although several stringent anonymization schemas are proposed, none of them succeeds in protecting the data privacy when considering the arbitrary auxiliary knowledge. For example, the composition attack proposed by Ganta etc. makes use of the auxiliary information from other anonymized data [6]. Their experiments show that almost all the partitioning-based schemas including  $k$ -anonymity,  $\ell$ -diversity and  $t$ -closeness are vulnerable to the composition attack. Other schemas such as  $m$ -invariance [7],  $(c, k)$ -safety [8], skyline privacy [9] plan to make some assumptions about adversaries’ auxiliary knowledge and then provide the corresponding protection strategies. Their problem is that the auxiliary knowledge comes from various sources, such as the common sense, adversaries’ personal knowledge and information from other published data. It is difficult to define the auxiliary knowledge of all adversaries; therefore the completeness of privacy is impossible to be guaranteed.

**Information Loss.** Some anonymization schemas may cause large information losses; and thus the query or mining results from the anonymized data may be quite different from the original data. For example, the result of Brickell’s experiment shows that in most cases, the trivial sanitization provides equivalent utility and better privacy than  $k$ -anonymity and other similar methods based on QI generalization and suppression [10]. The bucket-based methods such as anatomy, are excellent if we are just interested in the distributions of attributes or correlations among QI attributes. However, they also show their weakness in some workloads. These methods changes the one-to-one mapping between QI and sensitive values to the multiple-to-multiple mapping. If the query conditions contain both QI and sensitive attributes, the results are usually uncorrect.

## 1.2 Contribution

In this paper, we propose a novel methodology in the microdata publication, which is able to resist the attacks with auxiliary knowledge as well as provide considerable data utility.

First of all, we establish the schema of *presence anonymity* as the guideline of privacy preserving. We propose the idea of incorporating the authenticity uncertainty into the published data to reach the goal of auxiliary independent anonymization.

Additionally, we use *synthesizing* as the implementation of *presence anonymity*. The method retains the statistical features of the original data as well as add acceptable amount of noise into the published data.

Finally, we conduct comprehensive experiments on real data to verify the advantages of the proposed technique. The experimental results show that the resulting data of *synthesizing* can be used in common analysis tasks with reasonable correctness.

The rest of the paper is organized as follows. Section 2 discusses the basic ideas of privacy and utility. Section 3 introduces the auxiliary independent anonymization by an example. Section 4 gives the algorithm of *synthesizing*. Experimental results are reported in Section 5. Section 6 concludes the paper.

## 2 Basic Idea

### 2.1 Presence Privacy

Most schemas in the microdata publication are partitioning-based, which divide records in the data sets into disjoint groups satisfying certain criteria. Their resulting data is easy to be attacked because of: (1) exact sensitive value disclosure: the sensitive value in each group is published exactly; (2) locatability: by comparing the QI values one can locate the group in which the target record has been put in [6]. Based on these properties, an adversary can narrow down the set of possible sensitive values for an individual by the sets of sensitive values present in the corresponding groups and his/her auxiliary knowledge.

It is the basic requirement of microdata publication to guarantee the correctness of sensitive values, so the anonymization can only be accomplished by limiting the locatability. A simple way to realize that is to incorporate the authenticity uncertainty into the procedure of publication. If an adversary is uncertainty about whether records in the published data are also included in the original data, his/her confidence of locating the groups of target records in the published data shall be reduced. Based on the assumption, the conclusions inferred through combining the auxiliary knowledge with sensitive values in the corresponding groups will be un-creditable.

We call it the *presence anonymity* that no records in the published data exists in the original data with high probability. The definition is as follows:

**Definition 1.**  $T^*$  is a published version of  $T$ , and  $t \in T^*$  is an record in  $T^*$ .  $T^*$  is said to be privacy preserving, iff for  $\forall t \in T^*$ :

$$p(t \in T | t \in T^*) \leq \alpha \quad (1)$$

As the privacy threshold,  $\alpha$  defines the maximum probability that records in the published data exist in the original data. Under the constrain of *presence anonymity*, even if the adversary learns the exact sensitive values of individuals through the anonymized data and his/her auxiliary knowledge, their credibility will be influenced by the authenticity of published records. It is worth mentioning that the *presence anonymity* is not a partitioning-based schema, but randomization-based schema. The success of differential privacy [11] indicates that randomization is a reliable way to resist attacks with auxiliary knowledge.

## 2.2 Utility Measurement

One important reason why some anonymization schemas cause large information losses is that their utility measurements hardly reflect the effects of data usage. In most partitioning-based schemas such as  $k$ -anonymity and  $\ell$ -diversity, the utility of anonymized data is measured by the steps of generalization or sizes of bucket [3, 12]. Such syntactic utility measurements seem unreasonable, since it often happens that the anonymized data with high syntactic utility is useless in the data mining tasks [10].

Two aspects should be considered to carry out a complete assessment of data utility: (1) the information loss measurement from the data publisher's perspective. One of the main goals of data mining and statistics is to make statements about probability distribution of data - this is certainly true of classification, parameter estimation, hypothesis testing and regression [13]. In this spirit, we recommend to measure the information losses by the statistical divergence between the original data and anonymized data. The more similar their probability distributions are, the more same conclusions we may draw from them in most data mining tasks. (2) The usage of published data from the data user's perspective is measured by the difference between the mining results of the original data and anonymized data. Especially, the correlations between QI values and sensitive values should be verified, as it is the main reason why sensitive values are released.

## 3 Auxiliary Independent Anonymization

To consider the worst case, we usually make two assumptions: (1) the adversary knows that his/her target's record is included in the original data; (2) the adversary knows his/her target's QI values well. The anonymized data is supposed to guarantee the privacy of individuals in the presence of arbitrary auxiliary information. However, such auxiliary independent anonymization is merely met in most microdata publications.

We show the advantages of *presence anonymity* by a simple example. Table 1 includes some medical records that a hospital intends to release, in which the attribute of disease is sensitive. The adversary Alice attempts to locate Dave's record in the published data, and from other sources she gets to know that Dave is a 55-year old man, lives in zip code 100080 and recently visited that hospital.

Under the direction of  $k$ -anonymity, the hospital performs some pre-treatments on these records, and table 2 is the 2-anonymizations result. Just from the published data, Alice could hardly find out whether Dave has the lung cancer or ovarian cancer, as both tuple 7 and 8 satisfy his QI values well. However, based on the common sense that men are impossible to have the ovarian cancer, Alice could infer Dave's disease is the lung cancer easily.

Table 1. Original Table

Name	Zip	Age	Sex	Disease
Bob	100084	23	M	Flu
Charlie	100080	44	M	Flu
Dave	100080	55	M	Lung Cancer
Mark	100084	67	M	Lung Cancer
Frank	100084	31	M	Mumps
Gloria	100080	21	F	Flu
Angela	100080	42	F	Flu
Andy	100084	64	F	Breast Cancer
Jessica	100080	56	F	Ovarian Cancer
Karen	100084	38	F	Heart Disease

Table 2. 2-anonymity Table

	Zip	Age	Sex	Disease
1	100084	20-29	-	Flu
2	100080	20-29	-	Flu
3	100084	30-39	-	Mumps
4	100084	30-39	-	Heart Disease
5	100080	40-49	-	Flu
6	100080	40-49	-	Flu
7	100080	50-59	-	Lung Cancer
8	100080	50-59	-	Ovarian Cancer
9	100084	60-69	-	Lung Cancer
10	100084	60-69	-	Breast Cancer

Table 3 shows the 3-diversity version of table 1. By comparing QI values, Alice locates Dave's record in the second group. Supported by the information mentioned before, she excludes the possibility of ovarian cancer. If she believes that Dave's disease is serious, then the candidate of flu will be eliminated and Dave's privacy shall be breached.

Partitioning-based schemas are impossible to achieve the auxiliary independent anonymization, as they keep all the genuine records(records existing in the original data) in the published data. With QI values and other *auxiliary knowledge*, the adversary may eliminate most of candidate records in the published data and locate their target with high confidence.

According to the definition of *presence anonymity*, we replace 60% of the original records with fake records randomly, and get a published version which satisfies the *0.4-presence anonymity*. In table 4, Alice finds Dave's possible record easily, but she will not believe Dave has the lung cancer. As most of the genuine records have been replaced, the record which she locates is more likely to be a fake record. Therefore, no matter how much auxiliary knowledge she has, Alice could learn little about Dave from the published data. The *presence anonymity* cuts the relationships between the published data and auxiliary knowledge, and reach the goal of auxiliary independent anonymization.

**Table 3.** Bucketized Table

Zip	Age	Sex	Disease
100084	64	F	Lung Cancer
100084	23	M	Breast Cancer
100084	67	M	Flu
100084	31	M	Heart Disease
100084	38	F	Mumps
100080	55	M	Ovarian Cancer
100080	44	M	Flu
100080	21	F	Flu
100080	42	F	Flu
100080	56	F	Lung Cancer

**Table 4.** Randomly Replaced Table

	Zip	Age	Sex	Disease
1	100074	27	M	Flu
2	100080	40	M	Flu
3	100080	55	M	Lung Cancer
4	100074	69	M	Heart Disease
5	100074	51	M	Mumps
6	100080	21	F	Flu
7	100080	42	F	Flu
8	100084	74	M	Breast Cancer
9	100080	46	F	Ovarian Cancer
10	100074	38	F	Heart Disease

However, data in table 4 is unrepresentative to the original data; and it is impossible for researchers to obtain the correct conclusions from it. If the process of replacing genuine records with fake records is out of constrains, the statistical characteristics of the original data may be changed a lot. To guarantee the utility and privacy of data, the implementation of *presence anonymity* should meet the following requirements:

1. The number of fake records in anonymized data should be large enough, so that adversaries' confidence of learning information from the published data is reduced.
2. The process of replacing genuine records with fake records should be full of randomness to prevent adversaries from inferring the authenticity of published records.
3. The anonymized data must be representative to the original data, which means their statistical divergence isn't large.

We find the method of *synthesizing* meets the requirements above well. The idea of *synthesizing* is to learn a model from the original data and then to sample records with it. These new records will be published instead of the original data. We choose it, as the utility of *synthesizing* has been proved during the past work. [14] and [15] show significant work on performing statistical analysis of and drawing inferences from synthetic data.

## 4 Synthesizing

### 4.1 Notations

As in most works on microdata publication, we assume a microdata table consists of two parts: several quasi-identifier attributes and a sensitive attribute.

Let  $T = \{t_1, \dots, t_n\}$  be an original data table; and  $T^*$  is the published version of  $T$ .  $A = \{A_1, \dots, A_m\}$  is the set of attributes.  $t[A_i]$  denotes the value of attribute  $A_i$  for tuple  $t$ , and  $D[A_i](1 \leq i \leq d)$  denotes the domain of  $A_i$ .

$S$  is the sensitive attribute in  $T$ . The two tuples  $t_i$  and  $t_j$  are  $S$ -equivalent if  $t_i[S] = t_j[S]$ . This equivalence relation partitions  $T$  into several sensitive attribute equivalence classes  $EC(s)$ ,  $s \in D[S]$ ; where  $t \in EC(s)$  iff  $t[S]=s$ . Let  $Q = \{Q_1, \dots, Q_d\} \subseteq A-S$  be the set of quasi-identifier(QI) attributes. The domain of  $Q$  is  $D[Q] = \prod_{1 \leq i \leq d} D[Q_i]$ .

$U_s(q) = \{t \in EC(s) \mid t[Q] = q\}$  represents the set of tuples in  $EC(s)$  whose quasi-identifier values equal  $q$ .  $p(q|s)$  is the corresponding percentage of  $U_s(q)$  in  $EC(s)$ , which is computed by  $\frac{|U_s(q)|}{|EC(s)|}$ . The probability distribution of the equivalence class  $EC(s)$  can be represented by the set of  $p(q|s)(q \in D[Q])$ ; while the distribution of the whole table is described by the distributions of all  $EC(s)$ .

One requirement of data utility is to retain the correlations among arbitrary attributes, therefore the *synthesizing* should be performed in the domain of  $D[Q]$  in order to describe the distribution of records in the multi-dimensional space.

**Definition 2.** *The sampling space  $R = M_1 \times M_2 \times \dots \times M_d$ , where  $M_i(1 \leq i \leq d)$  corresponds to the attribute  $Q_i$  and  $D[M_i] = D[Q_i]$ . For any  $t \in EC(s)$ , if  $r \in R$  and  $r = t[Q]$ , then  $r$  is  $t$ 's mapping point of  $Q$  in the sampling space. Once the sampling granularity is fixed, the amount of sampling points in  $R$  is denoted by  $|R|$ .*

**Definition 3.**  *$t^* \in T^*$  is a record in the anonymized table; and  $r$  is the mapping points of  $t^*$ . If there is no  $t \in T$  satisfying  $r = t[Q]$ , then  $t^*$  is a fake record and  $r$  is a noise point; otherwise  $t^*$  is a genuine record. The set of fake records in  $EC(s)$  is denoted by  $ECF(s)$ , while the set of genuine records is  $ECR(s)$ .*

## 4.2 Learning

The *synthesizing* will be performed in each equivalence class separately; otherwise bias will be added into the distribution of sensitive values.

The objective of learning is to find the model for each  $EC(s)$  of the original data, which represents the distribution of mapping points. We choose smoothing as the learning mechanism. To smooth a data set means to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena. Its benefits include:

- Smoothing is to give a general idea of changes of values. The statistical characteristics of original data are retained.
- Smoothing helps to hide the peaks and isolated points(individuals whose QI values are quite different from others, thus can be identified easily) in the original data.
- Smoothing estimates the expectations of any points in the space. Fake records will be added through expectations of noise points.

In the smoothing family, the kernel regression estimation has been extensively studied in the statistics, machine learning and data mining. Given a sensitive

attribute value  $s \in S$ , using *Nadaraya-Watson* kernel weighted average [16], the probability expectation of  $q_m$  in  $EC(s)$  is,

$$\hat{p}(q_m | s) = \frac{\sum_{q_k \in D[Q]} p(q_k | s) \prod_{1 \leq i \leq d} K_i(Dis(q_m[A_i], q_k[A_i]))}{\sum_{q_k \in D[Q]} \prod_{1 \leq i \leq d} K_i(Dis(q_m[A_i], q_k[A_i]))} \tag{2}$$

If we don't find any  $q_k$  with  $s$  in the original data, then  $p(q_k | s) = 0$ .  $K_i$  is the kernel function for the  $i$ -th attribute  $A_i$ .

Generally the closer a point is to the estimated point, the greater its influence on the target will be. The fact can be described by the Epanechnikov kernel function [17]:

$$K_i(x_i) = \begin{cases} \frac{3}{4B_i} (1 - (\frac{x_i}{B_i})^2), & \text{if } |\frac{x_i}{B_i}| \leq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The bandwidth parameter  $B_i$  indicates the regression interval on the dimension of attribute  $A_i$ . The larger  $B_i$  is, the smoother the resulting distribution will be. The function  $Dis(q_m[A_i], q_k[A_i])$  in Equation 2 stands for the distance of attribute  $A_i$  from  $q_m$  to  $q_k$ . Both  $B_i$  and  $Dis(q_m[A_i], q_k[A_i])$  are advised to use the relative distance, as if  $A_i$  is a categorical attribute we can hardly calculate the absolute distance.

### 4.3 Sampling

The sampling is the procedure of generating the synthetic records by performing the multinomial sampling [18] with the smoothed distribution. To explain the details clearly, we'd like introduce some necessary definitions first.

**Definition 4.** *In a multinomial distribution, each trial results in exactly one of some fixed finite number  $k$  of possible outcomes, with probabilities  $p_1, \dots, p_k$  (so that for  $i=1, \dots, k$  and  $\sum_{i=1}^k p_i = 1$ ), and there are  $n$  independent trials. Then let the random variable  $X_i$  indicate the number of times outcome number  $i$  was observed over the  $n$  trials. The vector  $X=(X_1, \dots, X_k)$  follow a multinomial distribution with parameters  $n$  and  $p$ , where  $p=(p_1, \dots, p_k)$ . The probability mass function of the multinomial distribution is:*

$$\begin{aligned} f(x_1, x_k; n, p) &= p(X_1 = x_1 \wedge \dots \wedge X_k = x_k) \\ &= \begin{cases} \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x_k}, & \text{when } \sum_{i=1}^k x_i = n, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

for non-negative integers  $x_1 \dots x_k$ .

The procedure of generating a synthetic record for the equivalence class  $EC(s)$  can be treated as one trial; while points in the sampling space  $R$  represent the set of all possible outcomes with the probability of  $\hat{p}(r_k | s) (r_k \in R)$ . To generate  $n$  records for  $EC(s)$ , a sum of independent repetitions of this experiment should be executed, and the final result will be a sample from the multinomial distribution.

The typical procedure of multinomial sampling is as follow. First, reorder the smoothed distributions of EC(s) descendingly; and the result is  $\hat{p}(r_1 | s), \hat{p}(r_2 | s), \dots, \hat{p}(r_{|R|} | s)$ (this is only to speed up computation and not strictly necessary). For each trial, draw an auxiliary variable  $X$  from a uniform (0,1) distribution. The outcome will be:

$$j = \min_{j'=1}^k \left( \sum_{i=1}^{j'} p_i \geq X \right) \tag{5}$$

The  $Q$  value of  $r_j$  is then combined with the sensitive value  $s$  to generate a synthetic record. To finish the sampling of EC(s), the experiment above will be repeated independently for  $n$  times.

### 4.4 Privacy in Synthesizing

Basically, the *synthesizing* is the process of recoding the QI attributes of the original data. The smoothing introduces noises into the original distributions, which causes some fake records in the synthetic data. Adversaries could hardly determine which of them are genuine records as they know nothing about the distribution of original data. Their judgements about the authenticity of records mainly depend on the percentage of genuine records in synthetic data. If the percentage of genuine records in the resulting data is high, they may believe that their target is released with high confidence; and then execute the attack with auxiliary knowledge. Therefore the percentage of genuine records in the synthetic data affects the privacy of microdata seriously.

Suppose the smoothed distribution of EC(s) in learning is  $\hat{p}(r_1 | s), \hat{p}(r_2 | s), \dots, \hat{p}(r_{|R|} | s)$ , which describes the probabilities of points in the sampling space. In the multinomial sampling with  $n$ , their times of occurrences are expected to be  $\hat{p}(r_1 | s) \times n, \hat{p}(r_2 | s) \times n, \dots, \hat{p}(r_{|R|} | s) \times n$ . Therefore, the total number of genuine records in EC(s) is

$$\sum_{r \in R \wedge r=t[Q] \wedge t \in ECR(s)} \hat{p}(r|s) \times n.$$

To satisfy the definition of *presence anonymity*, the percentage of genuine records in the anonymized data is ought to be smaller than  $\alpha$ .

**Theorem 1.** *The equivalence class EC(s) satisfies  $\alpha$ -presence anonymity, iff:*

$$\sum_{r \in R \wedge r=t[Q] \wedge t \in ECR(s)} \hat{p}(r|s) \times n \leq \alpha \tag{6}$$

**Theorem 2.** *The synthetic table  $T^*$  satisfies  $\alpha$ -presence anonymity iff each EC(s) in  $T^*$  satisfies  $\alpha$ -presence anonymity.*

From Equation 6, we find how the privacy of synthetic data is depends on two properties: (1) the sampling granularity in the space; (2) the bandwidth parameter  $B_i$  of each attribute. The smoothing bandwidths which can be determined



by the accuracy requirements of research should be confirmed first, as they influence the utility of data obviously. The sampling granularity relies on the privacy threshold  $\alpha$ , and is determined recursively. We initialize it with the granularity of original data, and then calculate Equation 6 to see whether the *presence anonymity* is satisfied. If not, choose a finer-granularity sampling space. The procedure is repeated until Equation 6 is met.

#### 4.5 Algorithm for Synthesizing

The algorithm of *synthesizing* is consisted of three parts: (1) model learning by smoothing; (2) checking whether current sampling granularity meets the *presence anonymity*; (3) sampling with the model learned before to generate the published data. The detail is as follows:

---

##### Algorithm 1. Synthesizing Algorithm

---

```

1 Set the regression intervals of attributes based on the accuracy requirements;
2 forall  $s_i \in D[S]$  do
3   Count the size of  $EC(s_i)$  from the original data, and marked as  $|EC(s_i)|$ ;
4   forall  $q_j$  in the original data do
5     Compute  $p(q_j | s_i) = \frac{|U_{s_i}(q_j)|}{|EC(s_i)|}$ ;
6 forall  $s_i \in D[S]$  do
7   Initialize the sampling granularity with the granularity of the original data ;
8   forall  $r_j \in R$  in the sampling space do
9     Compute  $\hat{p}(r_j | s_i)$  use Equation 2;
10  while Equation 6 is not satisfied do
11    select a finer-granularity space;
12    forall  $r_j \in R$  in the sampling space do
13      Compute  $\hat{p}(r_j | s_i)$  use Equation 2;
14 forall  $s_i \in D[S]$  do
15   Sort the probability distributions in  $EC(s_i)$ 's smoothed model descendingly;
16   for  $i=1$  to  $|EC(s_i)|$  do
17     Draw a variable  $X$  from a uniform (0, 1) distribution;
18     Find the sampling point  $r_j$  of  $X$  by Equation 5;
19     Combine the  $Q$  value of  $r_j$  with  $s_i$  to generate a published record.
```

---

The time cost of learning is  $o(|D[S]| \times |R|^2 \times \frac{\prod_{1 \leq i < d} B_i}{|D[Q]|})$ , where  $|D[S]|$  is the number of different sensitive values. The values of  $|D[S]|$ ,  $B_i$  and  $|D[Q]|$  are fixed beforehand; so the time cost of learning depends on  $|R|$ . The time complexity of sampling is  $o(N \times |R|)$ , where  $N$  is the number of records we tend to sample. The time cost of the whole algorithm is mainly influenced by the number of points in the space, which actually is determined by the threshold  $\alpha$ . More privacy the published data is required, more time it will cost to generate the synthetic data.

## 5 Experiments

The main goals of the experiments includes: (1) to show the efficiency of the algorithm when we select different private thresholds; (2) to verify the small utility losses of *synthesizing* by the statistical divergences; (3) to check whether researchers can draw correct conclusions from synthetic data in the mining tasks.

The data set used in the experiment is a part of the 2006 American census survey collected from the US census(<http://www.census.gov/>). We select five attributes as shown in Table 5, and the sensitive attribute is Occupation. Tuples with missing values are eliminated and finally there are 44112 valid ones in total. The values of category attributes have been converted into numerics according to the code book. All algorithms are implemented in JAVA and the experiments are done on a 3.00 GHz Pertium (R) machine with 3.0GB of RAM.

In the experiment, we compare the *synthesizing* with the algorithm  $\mu$ -argus [19], which is an approximation algorithm of  $k$ -anonymity. To ensure the fairness, the parameters  $\alpha$  and  $k$  are fixed as table 6, so that the two algorithms provide the similar level of privacy.

Table 5. Attributes Summary

Attribute	Type	Domain	Step
AGE	Numeric	20-70	1
WORHOUR	Category	10-80	1
SEX	Enumerate	N/A	N/A
SALARY	Numeric	5000-60000	1000
OCCUPATION	Category	N/A	N/A

Table 6. Algorithm Parameters

	$\alpha$ of Synthesizing	$k$ of $\mu$ -argus
1	0.33	3
2	0.25	4
3	0.20	5
4	0.15	7
5	0.10	10

### 5.1 Computing Efficiency

The efficiency experiment concerns about the time costs of both algorithms when different parameters are selected. The results are shown in Fig. 11. When the privacy preserving demands little (for example,  $\alpha=1/k=0.33$ ), the efficiency of *synthesizing* is better than  $\mu$ -argus. However, its time cost increases rapidly as soon as a smaller  $\alpha$  is set. A small value of  $\alpha$  requires that the percentage of genuine records in the published data should be reduced. Thus, a finer-granularity space is needed to ensure that points in the regression area are enough to reduce the posterior probabilities of genuine records. As the finer-granularity space leads to: (1) more points to be smoothed in learning; (2) more points in the regression area; (3) more points in the sampling space; the time cost increases seriously.

Significantly,  $\mu$ -argus is an approximation algorithm of  $k$ -anonymity. The exact implementation of  $k$ -anonymity has been proved to be the NP-hard problem [20], so its time cost may be much higher than our experimental results. Even so, the efficiency of *synthesizing* is unendurable sometimes. There are three measurements to reduce its cost:

1. Generalize the QI attributes to acceptable levels before *synthesizing*, if the mining tasks demand little about the accuracy.

2. Select larger smoothing bandwidths, so that a coarser-granularity may also meet the value of  $\alpha$ .
3. Reduce the number of sampling points of each EC(s) in proportion.

In Fig. 1, as  $\alpha$  become smaller, the time cost of learning increases more rapidly than sampling. This is because the learning is executed for more than one times to find an appropriate granularity.

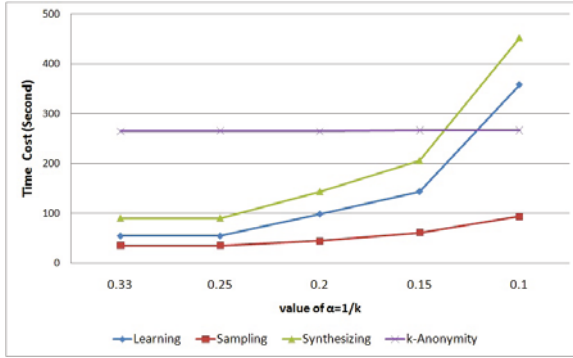


Fig. 1. Time Cost of Algorithms

### 5.2 Information Losses Measurement

The measurements of the statistical distance main contain the information theoretic approaches and probability measure based approaches. We choose the latter, as it cares the change of distributions more. The Bhattacharyya coefficient [21] can be used to determine the relative closeness of the two samples based on the partitions. When the domain space is split into a chosen number of partitions, the Bhattacharyya coefficient shall be:

$$Bhattacharyya(a, b) = \sum_{i=1}^n \sqrt{a_i * b_i} \tag{7}$$

where considering the samples  $a$  and  $b$ ,  $n$  is the number of partitions, and  $a_i$ ,  $b_i$  are the number of members  $a$  and  $b$  have in the  $i$ -th partition. The Bhattacharyya coefficient will be 0 if there is completely no overlap in every partition. The maximum is the number of sampling number, if  $a$  and  $b$  have the same number of members in each partition. Based on the Bhattacharyya coefficient, the measurement of information losses in our experiment is defined as:

$$Information\ Loss = 1 - \frac{Bhattacharyya\ coefficient}{Size\ of\ T} \tag{8}$$

Fig. 2 shows the experimental results. Obviously, information losses caused by the synthesizing are much smaller than  $\mu$ -argus, which is less than 3%.

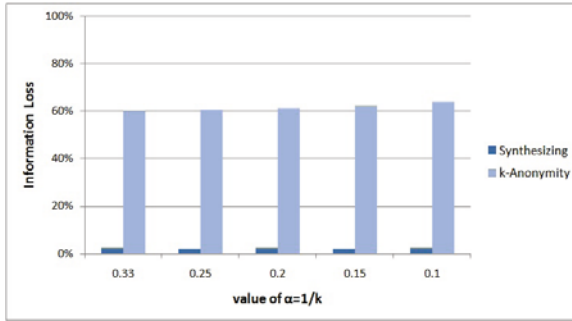


Fig. 2. Information Loss

### 5.3 Utility in Queries

The methods of *synthesizing* can not guarantee the correctness of all query results. For example, in the workloads requiring for the accurate values of a single record, the synthetic data may provide bad results. Here we'd like to verify the utility of synthetic data in aggregate queries, as their results depends on the statistical features more.

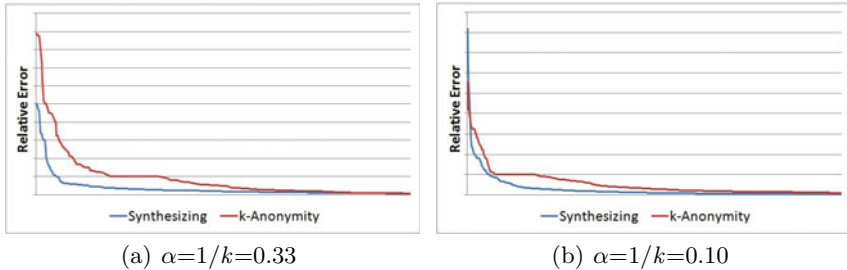


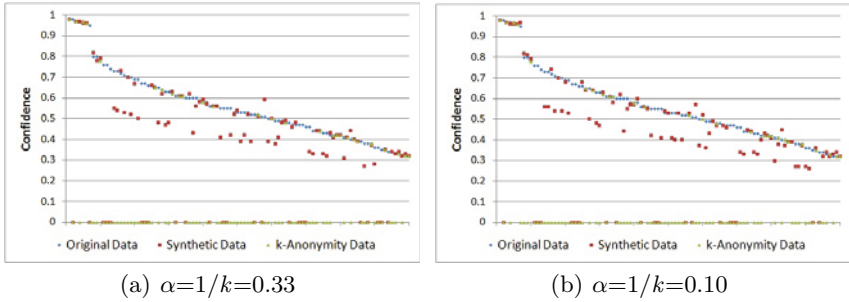
Fig. 3. Relative Error of Queries

In the experiment, we executed 3000 SQL queries on the original data and anonymized data. The data utility is measured by the relative error of query results. All the SQL sentences are generated randomly, which contain the aggregate functions (e.g., count, sum and avg). The query conditions may involve one, two or three attributes, which are also selected randomly. The results of  $\alpha=1/k=0.33$  and  $\alpha=1/k=0.10$  are shown in Fig. 3. The relative errors of the synthetic data are smaller than 10%, which is better than the  $k$ -anonymity data.

### 5.4 Utility in Data Mining Tasks

The utility of synthetic data in data mining is verified by the Apriori algorithm [22], which is a classic rule mining algorithm. In the experiment, the minimum

support of rules is set as 0.05. We order all the rules mined from the original data descendingly by their confidence, and select the top 100 ones as the strong association rules. Then we check whether they can be mined from the anonymized data, and compare the confidence with the original data. The results are shown in Fig. 4, from which we find more than 70% of these rules can be mined from the synthetic data, while only 30% from the  $k$ -anonymity data.



**Fig. 4.** Confidence of Rule Mining

## 6 Conclusion

Although existing microdata publication techniques promote the privacy obviously, the utility of published data still needs to be improved. We tried the way of *synthesizing* to protect personal information. The *synthesizing* overcomes the threat from the attack with auxiliary knowledge. Extensive experiments prove that the approach enables researchers to reach, from the resulting data, correct or approximately correct conclusions in most query and mining tasks.

During the research, we find some problems remain. First, the time complexity of the algorithm is too high. We are looking for some approximate methods to reduce the time cost. Second, smoothing has not been proved as the most suitable learning method. We may try some other learning algorithms. Third, the synthetic data must be used in more applications to verify the utility.

## Acknowledgment

This work was supported by NHTP (2009CB320706, 2008ZX01045-001, and 2009ZX01045-004-001).

## References

1. Sweeney, L.:  $k$ -anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10, 557–570 (2002)
2. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: *VLDB*, pp. 139–150 (2006)

3. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkatasubramanian, M.:  $\ell$ -diversity-diversity: Privacy beyond k-anonymity. In: ICDE, p. 24 (2006)
4. Koudas, N., Srivastava, D., Yu, T., Zhang, Q.: Distribution-based microdata anonymization. PVLDB 2, 958–969 (2009)
5. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE, pp. 106–115 (2007)
6. Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: KDD'08, pp. 265–273. ACM, New York (2008)
7. Xiao, X., Tao, Y.: M-invariance: towards privacy preserving re-publication of dynamic datasets. In: SIGMOD Conference, pp. 689–700 (2007)
8. Martin, D.J., Kifer, D., Machanavajjhala, A., Gehrke, J., Halpern, J.Y.: Worst-case background knowledge for privacy-preserving data publishing. In: ICDE, pp. 126–135 (2007)
9. Chen, B.C., Ramakrishnan, R., LeFevre, K.: Privacy skyline: Privacy with multi-dimensional adversarial knowledge. In: VLDB, pp. 770–781 (2007)
10. Brickell, J., Shmatikov, V.: The cost of privacy: destruction of data-mining utility in anonymized data publishing. In: KDD'08, pp. 70–78. ACM, New York (2008)
11. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
12. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13 (2001)
13. Kifer, D., Gehrke, J.: Injecting utility into anonymized datasets. In: SIGMOD Conference, pp. 217–228 (2006)
14. Raghunathan, T., Reiter, J., Rubin, D.: Multiple imputation for statistical disclosure limitation. Journal of Official Statistics (2003)
15. Woodcock, S.D., Benedetto, G.: Distribution-preserving statistical disclosure limitation. Comput. Stat. Data Anal. 53, 4228–4242 (2009)
16. Nadaraya, E.A.: On estimating regression. Theory of Probability and its Applications 9, 141–142 (1964)
17. Wolf, M.: Nonparametric econometrics: Theory and practice. qi li and jeffrey scott racine. Journal of the American Statistical Association 103, 885–886 (2008)
18. Trenkler, G.: Statistical distributions. Computational Statistics & Data Analysis 19, 483–484 (1995)
19. Hundepool, A., Willenborg, L.:  $\mu$ - and  $\tau$ -argus: Software for statistical disclosure control. In: Third Int'l Seminar Statistical Confidentiality (1997)
20. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: PODS, pp. 223–228 (2004)
21. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of the Calcutta Mathematical Society 35, 99–109 (1943)
22. Han, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco (2005)

# Anonymizing Transaction Data to Eliminate Sensitive Inferences

Grigorios Loukides<sup>1</sup>, Aris Gkoulalas-Divanis<sup>2,\*</sup>, and Jianhua Shao<sup>3</sup>

<sup>1</sup> Vanderbilt University, USA  
grigorios.loukides@vanderbilt.edu

<sup>2</sup> Vanderbilt University, USA  
aris.gkoulalas@vanderbilt.edu

<sup>3</sup> Cardiff University, UK  
j.shao@cs.cardiff.ac.uk

**Abstract.** Publishing transaction data containing individuals' activities may risk privacy breaches, so the need for anonymizing such data before their release is increasingly recognized by organizations. Several approaches have been proposed recently to deal with this issue, but they are still inadequate for preserving both data utility and privacy. Some incur unnecessary information loss in order to protect data, while others allow sensitive inferences to be made on anonymized data. In this paper, we propose a novel approach that enhances both data utility and privacy protection in transaction data anonymization. We model potential inferences of individuals' identities and their associated sensitive transaction information as a set of implications, and we design an effective algorithm that is capable of anonymizing data to prevent these sensitive inferences with minimal data utility loss. Experiments using real-world data show that our approach outperforms the state-of-the-art method in terms of preserving both privacy and data utility.

## 1 Introduction

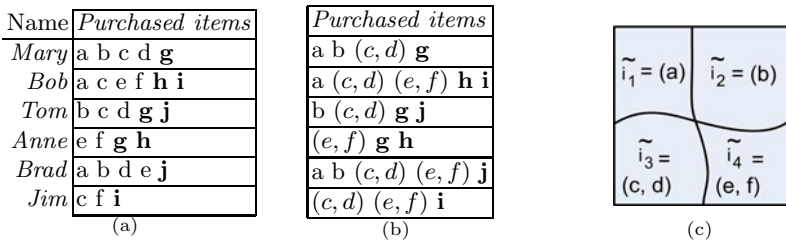
Transaction data are comprised of records, called *transactions*, that contain information relating to individuals' behaviors or activities, for example, search engine query terms used by an individual or goods purchased from a vendor by a consumer. Transaction data can play a central role in applications such as personalized web search and marketing analysis, but need to be anonymized before release because they contain private information about individuals. Unfortunately, simply de-identifying transactions (i.e., removing personal identifiers such as names), is insufficient to protect privacy. This became evident when a New York Times journalist managed to re-identify an individual from a de-identified search keywords dataset released by AOL Research [2].

---

\* The author is currently affiliated with IBM Research, Zürich.

### 1.1 Motivation

Publishing transaction data safely requires the elimination of two types of inference that pose privacy threats. The first type is *identity disclosure*, which occurs when an individual is identified to be associated with a particular transaction in the published data. Consider the release of de-identified data of Fig. 1(a), for example, where items purchased by several individuals are recorded. Some items are considered to be *sensitive* (denoted by bold letters), e.g. medicine, and individuals will not want their purchase of such items to be made public. Other items are considered to be *public*, and individuals would not mind revealing their purchase. Knowing that *Mary* has purchased *a*, *b* and *c*, an attacker can link her to the first transaction of Fig. 1(a), since this is the only transaction that contains *a*, *b* and *c*. The second type of inference is *sensitive itemset disclosure*, which occurs when an individual is identified to be associated with a set of sensitive items (or an itemset). For example, suppose that an attacker knows that *Tom* has purchased *c* and *d*. Although the attacker cannot associate *Tom* to a specific transaction, since there are two transactions in Fig. 1(a) that contain *c* and *d*, the attacker can still infer that *Tom* has purchased a sensitive item *g*, as both transactions contain *g*.



**Fig. 1.** An example of: (a) Original dataset, (b) Anonymized dataset, and (c) Set-based generalization

So, releasing transaction data involves a risk of identity and/or sensitive itemset disclosure. To manage such risks, techniques that help control the probability of disclosure may be employed. These techniques work typically by modifying public items [9,10,7], as it is often more important to keep sensitive items intact in applications [10]. We argue that a good method should produce anonymized data that satisfies two important requirements:

- privacy is preserved against both identity and sensitive itemset disclosure, since both types of disclosure can happen in practice, and
- data is protected no more than necessary, because over-protection can significantly reduce data utility.

To the best of our knowledge, none of the existing approaches for anonymizing transaction data [9,10,7,4,6] is capable of meeting both of these requirements. These methods deal with one type of disclosure [4,9,7,6] and make restrictive assumptions about privacy requirements [9,10,6], which tend to harm data utility unnecessarily [7].



## 1.2 Contributions

In this work, we propose a novel approach to anonymizing transaction data. The main idea behind our method is to allow data owners to specify detailed privacy requirements, so that protection for both identity and sensitive itemset disclosure is only exercised where necessary, and to incorporate these requirements into an effective anonymization algorithm. Our work makes the following contributions.

First, we propose a rule-based model that allows data owners to formulate detailed privacy protection requirements. Our model is founded on a notion of implication between public and sensitive items, called *PS-rule*. For example,  $ac \rightarrow \mathbf{hi}$  is a PS-rule which specifies that linking an individual to a specific transaction through public items  $ac$  (identity disclosure) and inferring  $\mathbf{hi}$  using  $ac$  (sensitive itemset disclosure) are to be prevented. Our rule model offers two important benefits: (i) it is the only one that does not place any restrictions on which public and sensitive items are to be protected, which allows a much wider class of privacy protection requirements to be specified than those considered in [9,10,6], and (ii) it captures requirements for protection against both identity and sensitive itemset disclosure.

Second, to anonymize transactions based on our privacy model, we design a new, effective method called Rule-Based Anonymization of Transactions (RBAT). RBAT uses item generalization [9,7], a technique that modifies transactions by replacing public items with more general but semantically consistent generalized items. This algorithm operates in a top-down fashion, starting with the most generalized transaction dataset and then gradually replacing generalized items with less general ones, as long as data remains protected. When applied to the data of Fig. 1(a), for example, RBAT constructs the dataset shown in Fig. 1(b). In this dataset items  $c$  and  $d$  are replaced by a generalized item  $(c,d)$ , which means that the actual transaction can contain  $c$  or  $d$  or  $c$  and  $d$ . Our algorithm has three notable strengths: (i) it is the first to employ generalization to prevent both identity and sensitive itemset disclosure, which is important in applications where alternative techniques (e.g. item elimination [10]) are not desirable, (ii) it exploits a pruning strategy that significantly speeds up the process of data anonymization, and (iii) it operates in a way that is independent of how data is generalized, which allows flexible generalization models (e.g., that of [7]) to be employed to enhance data utility. Our extensive experiments show that RBAT is able to achieve a result that is several orders of magnitude better than the state-of-the-art algorithm [9] in terms of data utility, while guaranteeing protection against sensitive itemset disclosure.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. In Section 3, we formally define our privacy specification model and the problem we consider. RBAT is presented in Section 4. Finally, Section 5 reports experimental results and Section 6 concludes the paper.

## 2 Related Work

There are several principles and algorithms for anonymizing relational data (see [3] for a survey), but, as shown in [10], they are inadequate for anonymizing transactions without incurring an excessive amount of information loss. This is because they have rather different semantics, e.g. transactions have a varying and typically large number of items.

Anonymizing transaction data to guard against identity disclosure has been considered in [9, 6, 7]. Terrovitis et al. [9] proposed  $k^m$ -anonymity, a principle that aims to prevent attackers from linking an individual to less than  $k$  transactions, assuming that they know at most  $m$  items of any transaction. Contrary to the approach we propose,  $k^m$ -anonymity can only guarantee protection against identity disclosure, and it does not permit flexible privacy requirements to be specified, as it assumes that all combinations of  $m$  public items require protection. The authors of [9] also designed Apriori Anonymization (AA), an algorithm that enforces  $k^m$ -anonymity using item generalization. AA operates in a bottom-up fashion; it begins with (single) items and iteratively replaces them with less specific generalized items, as long as  $k^m$ -anonymity is not satisfied. Another generalization-based algorithm was proposed by He and Naughton [6]. While this algorithm has been shown to achieve better data utility than AA does, it still considers a single privacy requirement only [8]. Moreover, it generates data that may be of limited utility in practice because the same item may be replaced by different generalized items in different transactions. This can cause problems for data mining algorithms to work effectively on anonymized transactions [3]. An approach to anonymizing transactions that allows data owners to protect specific itemsets from identity disclosure was proposed in our recent work [7], but it does not account for protection against sensitive itemset disclosure.

A few works focus on preventing sensitive itemset disclosure in publishing transaction data, but none of them allow detailed privacy requirements to be specified. Ghinita et al. [4] developed a novel approach that releases transactions in groups, each of which contains public items in their original form and a summary of the frequencies of the sensitive items. Since public items are released intact, [4] provides no protection against identity disclosure. Xu et al. [10] proposed  $(h, k, p)$ -coherence, a privacy principle that addresses both identity and sensitive itemset disclosure, and a suppression-based algorithm to enforce it. This principle treats public items similarly to  $k^m$ -anonymity (the function of parameter  $p$  is the same as  $m$  in  $k^m$ -anonymity) and additionally limits the probability of inferring any sensitive item using a parameter  $h$ . Since [4] effectively assumes that all items require protection against either identity or sensitive itemset disclosure, it limits the type of privacy requirement that can be specified and may overprotect data against specific privacy requirements.

## 3 Background and Problem Formulation

After presenting some preliminary definitions, we describe the generalization model and utility metric employed in this paper in Section 3.2. Subsequently,

we define our rule-based privacy protection model and formulate the problem we study in Sections 3.3 and 3.4, respectively.

### 3.1 Preliminaries

Let  $\mathcal{I} = \{i_1, \dots, i_M\}$  be a finite set of literals, called *items*, where  $M$  denotes the cardinality of the set. Any subset  $I \subseteq \mathcal{I}$  is called an *itemset* over  $\mathcal{I}$ , and we represent  $I$  as a concatenation of the items it contains. An itemset that has  $m$  items or equivalently a *size* of  $m$ , is called an  $m$ -itemset. Let  $\mathcal{P}$  and  $\mathcal{S}$  be sets of items that contain *public* and *sensitive* items respectively. We have  $\mathcal{P} \cup \mathcal{S} = \mathcal{I}$  and  $\mathcal{P} \cap \mathcal{S} = \emptyset$ .

A *transaction*  $T$  over  $\mathcal{I}$  is a pair  $T = (tid, I)$ , where  $I$  is an itemset and  $tid$  its unique identifier. A transaction dataset  $\mathcal{D} = \{T_1, \dots, T_N\}$  is a set of  $N$  transactions over  $\mathcal{I}$ , each having a different  $tid$ . A transaction  $T = (tid, J)$  supports an itemset  $I$  over  $\mathcal{I}$ , if  $I \subseteq J$ . Given an itemset  $I$  over  $\mathcal{I}$  in  $\mathcal{D}$ , we use  $sup(I, \mathcal{D})$  to represent the number of transactions in  $\mathcal{D}$  that support  $I$ . These transactions are called *supporting transactions* of  $I$  in  $\mathcal{D}$ . Moreover, we assume that  $\mathcal{D}$  contains all items of  $\mathcal{I}$ .

Given two itemsets  $I \in \mathcal{P}$  and  $J \in \mathcal{S}$ , we define a *PS-rule* as an implication of the form  $I \rightarrow J$ , where  $I$  is the *antecedent* and  $J$  the *consequent*. A PS-rule models an association between public and sensitive items and, as we will discuss later, it can be used to capture both identity disclosure based on  $I$  and sensitive itemset disclosure of  $J$ . In this work, we assume that PS-rules are specified by data owners based on their privacy requirements [5].

### 3.2 Generalization Methodology

We use item generalization [9,7], a technique which maps individual public items to generalized ones. We denote a generalized item by listing its item(s) in brackets, e.g.  $(c, d)$  or  $(e, f)$  in Fig. 1(b)<sup>1</sup>. Generalization can help prevent identity disclosure as it increases the number of transactions that may be linked to an individual through public items, i.e. the support of a generalized item in the anonymized data is greater than or equal to the support of any item mapped to it in the original data [7]. As an example, consider the mapping of items  $c$  and  $d$  in Fig. 1(a), to a generalized item  $(c, d)$  in Fig. 1(b).  $(c, d)$  is supported by 6 transactions in Fig. 1(b), whereas  $c$  and  $d$  are supported by 4 and 3 transactions in Fig. 1(a), respectively.

In this work, we use set-based generalization [7] which offers a larger space of possible generalizations than other models do [9,10] and has been shown to enhance data utility. This model uses a function  $\Phi$  to map each item  $i$  in  $\mathcal{P}$  to a non-empty subset of  $\mathcal{P}$  that contains  $i$ , and then uses this non-empty subset as a generalization of  $i$ . This is formally defined below.

**Definition 1 (Set-Based Generalization).** *A set-based generalization is a partition  $\tilde{\mathcal{P}}$  of  $\mathcal{P}$  in which each item  $i$  in  $\mathcal{P}$  is mapped to a generalized item  $\tilde{i}$  in  $\tilde{\mathcal{P}}$  using a generalization function  $\Phi : \mathcal{P} \rightarrow \tilde{\mathcal{P}}$ .*

<sup>1</sup> For clarity, we will drop  $()$  when a generalized item contains only one item.

*Example 1.* Consider the set-based generalization of public items  $\{a, b, c, d, e, f\}$  in Fig. 1(a) that is constructed using the partition illustrated in Fig. 1(c). Items  $a$  and  $b$  are mapped to themselves (i.e.,  $\Phi(a) = \tilde{i}_1 = a$  and  $\Phi(b) = \tilde{i}_2 = b$ ), while  $c$  and  $d$  are mapped to a generalized item  $\tilde{i}_3$  (i.e.,  $\Phi(c) = \Phi(d) = \tilde{i}_3 = (c, d)$ ) which is interpreted as representing  $c$  or  $d$  or  $c$  and  $d$ .

There may exist several generalizations of a dataset, and the one that harms data utility least, as quantified by information loss measures [9, 4, 10, 7], is typically preferred. In this work, we use the *Utility Loss (UL)* measure [7].

**Definition 2 (Utility Loss).** *The Utility Loss (UL) for a generalized item  $\tilde{i}_m$  is defined as  $UL(\tilde{i}_m) = \frac{2^{|\tilde{i}_m|-1}}{2^{|\mathcal{P}|-1}} \times w(\tilde{i}_m) \times \frac{\text{sup}(\tilde{i}_m, \tilde{\mathcal{D}})}{N}$ , where  $|\tilde{i}_m|$  denotes the number of items in  $\mathcal{P}$  that are mapped to  $\tilde{i}_m$  using  $\Phi$ , and  $w : \tilde{\mathcal{P}} \rightarrow [0, 1]$  is a function assigning a weight according to the perceived importance of  $\tilde{i}_m$ . Based on this definition, the Utility Loss (UL) for a generalized dataset  $\tilde{\mathcal{D}}$  is defined as  $UL(\tilde{\mathcal{D}}) = \sum_{\forall \tilde{i}_m \in \tilde{\mathcal{P}}} UL(\tilde{i}_m)$ .*

UL quantifies information loss based on the size, weight and support of generalized items, imposing a “large” penalty on generalized items that are comprised of a large number of “important” items that appear in many transactions. The denominators  $2^{|\mathcal{P}|-1}$  and  $N$  in Definition 2 are used for normalization purposes. For example, to compute the UL score for  $\tilde{i}_3 = (c, d)$  in Fig. 1(b) assuming  $w(\tilde{i}_3) = 1$ , we have  $UL(\tilde{i}_3) = \frac{2^2}{2^6-1} \times 1 \times \frac{5}{8} \approx 0.053$ .

### 3.3 Protecting PS-Rules

We now give sufficient and necessary conditions to guarantee protection against identity and sensitive itemset disclosure based on the antecedent and consequent of a PS-rule in Definition 3.

**Definition 3 (PS-rule protection).** *Given a generalized version  $\tilde{\mathcal{D}}$  of  $\mathcal{D}$  and parameters  $k$  and  $c$ , a PS-rule  $I \rightarrow J$  is protected in  $\tilde{\mathcal{D}}$  if and only if (1)  $\text{sup}(\tilde{I}, \tilde{\mathcal{D}}) \geq k$ , and (2)  $\text{conf}(\tilde{I} \rightarrow J, \tilde{\mathcal{D}}) \leq c$ , where  $\tilde{I} = \bigcup_{\forall i \in I} \Phi(i)$  is an itemset of  $\tilde{\mathcal{D}}$  constructed from  $I$ ,  $\text{conf}$  is a function, called confidence, defined by  $\frac{\text{sup}(\tilde{I} \cup J, \tilde{\mathcal{D}})}{\text{sup}(\tilde{I}, \tilde{\mathcal{D}})}$ ,  $k$  is an integer in  $[2, N]$ , and  $c$  is a real number in  $[0, 1]$ .*

The protection of a PS-rule is achieved when (1)  $\tilde{\mathcal{D}}$  contains at least  $k$  transactions that support the itemset  $\tilde{I}$ , which is comprised of the generalized items to which the items of the antecedent of the PS-rule are mapped, and (2) at most  $c \times 100\%$  of the transactions that support  $\tilde{I}$  in  $\tilde{\mathcal{D}}$ , also support  $\tilde{I} \cup J$ . Satisfying the first condition ensures that an attacker with the knowledge of the antecedent of a PS-rule cannot link an individual to less than  $k$  transactions in the released data. This effectively limits the probability of identity disclosure based on the antecedent of the PS-rule to no more than  $\frac{1}{k}$ . Satisfying the second condition thwarts sensitive itemset disclosure, because, knowing that an individual is associated with the antecedent of the PS-rule, an attacker cannot infer that this

individual is associated with the consequent of the rule, or with any superset of it (due to the monotonicity of support [11]), with a probability that exceeds  $c$ . Note that since our intention is to satisfy specified privacy requirements only, we do not attempt to limit the probability of linking an individual to subsets of sensitive itemsets appeared in the consequents of protected PS-rules.

*Example 2.* Consider the data of Fig. 1(b) and assume that a PS-rule  $ac \rightarrow \mathbf{hi}$  is to be protected. Suppose that we have  $k = 2$  and  $c = 0.5$ . It is easy to see that  $ac \rightarrow \mathbf{hi}$  is protected, as there are 3 transactions supporting the itemset  $\bigcup_{i \in \{a,c\}} \Phi(i) = a(c,d)$ , and only one of which supports  $a(c,d) \cup \mathbf{hi}$ . This implies that an attacker who knows that *Mary* has purchased  $a$  and  $c$ , needs to distinguish among three transactions in Fig. 1(b) to find out *Mary*'s real transaction. Furthermore, knowing that *Mary* has purchased  $a$  and  $c$ , an attacker cannot infer whether she purchased the sensitive itemset  $\mathbf{hi}$  with a probability higher than  $\frac{1}{3}$ .

It should be noted that detailed privacy requirements against both identity and sensitive itemset disclosure can be modeled using PS-rules, as there are no restrictions on the itemsets that can be used as antecedents or consequents of rules. To illustrate the expressive power of our model, we show how it can be used to specify the privacy requirements of  $k^m$ -anonymity [9] and of  $(h, k, p)$ -coherence [10]. To support  $k^m$ -anonymity, it suffices to construct one PS-rule  $I \rightarrow J$  for each  $m$ -itemset  $I$  in  $\mathcal{P}$  ( $J$  can be any item in  $\mathcal{S}$ , since  $k^m$ -anonymity does not account for sensitive itemset disclosure). Then, a transaction dataset in which these PS-rules are protected, for a given  $k$  and any  $c$ , will also satisfy  $k^m$ -anonymity for the same  $k$ . To support  $(h, k, p)$ -coherence, we need to construct one PS-rule  $I \rightarrow J$  for each  $p$ -itemset  $I$  in  $\mathcal{P}$  and for each 1-itemset (item)  $J$  in  $\mathcal{S}$ . By doing so, we ensure that a transaction dataset in which the specified PS-rules are protected, for a given  $k$  and  $c$ , will also satisfy  $(h, k, p)$ -coherence for the same  $k$  and  $h = c \times 100\%$ .

### 3.4 Problem Statement

**Problem.** Given a transaction dataset  $\mathcal{D}$ , a set of PS-rules  $\Theta$ , and parameters  $k, c$ , construct a generalized version  $\tilde{\mathcal{D}}$  of  $\mathcal{D}$  using the set-based generalization model such that: (1) each PS-rule in  $\Theta$  is protected in  $\tilde{\mathcal{D}}$ , and (2) the amount of utility loss  $UL(\tilde{\mathcal{D}})$  is minimal.

This is a challenging problem because (i) it is NP-hard (the proof is by reduction to the problem considered in [10]), and (ii) it may not have a solution, even when any amount of utility loss is allowed, as Theorem 1 shows (the proof of this theorem is omitted due to space constraints).

**Theorem 1.** Given  $\mathcal{D}$ , a set of PS-rules  $\Theta = \{I_1 \rightarrow J_1, \dots, I_j \rightarrow J_j\}$ , and parameters  $k$  and  $c$ , a generalized dataset  $\tilde{\mathcal{D}}$  in which each PS-rule in  $\Theta$  is protected, can be constructed using set-based generalization if and only if (1)  $N \geq k$ , and (2)  $\text{sup}(J_r, \mathcal{D}) \leq N \times c$ , for each  $I_r \rightarrow J_r, r \in [1, j]$  in  $\Theta$ .

## 4 Anonymization Algorithm

In this section, we introduce RBAT (Rule-Based Anonymization of Transactions), an effective heuristic to solve the aforementioned problem. Given that the most generalized version  $\tilde{\mathcal{D}}$  of  $\mathcal{D}$  (i.e. all public items of  $\mathcal{D}$  are mapped to a single generalized item  $\tilde{i}$ ) protects all PS-rules in  $\Theta$ <sup>2</sup>, RBAT operates in a top-down fashion, starting with  $\tilde{\mathcal{D}}$  and then progressively refining it (i.e., replacing generalized items in  $\tilde{\mathcal{D}}$  with more specific ones), for as long as all PS-rules in  $\Theta$  remain protected in  $\tilde{\mathcal{D}}$ . The pseudocode of RBAT is given in Algorithm 1.

---

### Algorithm 1. RBAT( $\tilde{i}, \tilde{\mathcal{D}}, \mathcal{D}, \Theta, k, c$ )

---

**input:** Generalized item  $\tilde{i}$ , generalized dataset  $\tilde{\mathcal{D}}$ , original dataset  $\mathcal{D}$ ,  $\Theta$ ,  $k$ ,  $c$

**output:** Dataset  $\tilde{\mathcal{D}}$  such that all PS-rules in  $\Theta$  are protected

1.  $\{\tilde{i}_l, \tilde{i}_r\} \leftarrow \text{Split}(\tilde{i})$
  2.  $\mathcal{D}' \leftarrow \text{Update}(\tilde{\mathcal{D}}, \mathcal{D})$
  3. **if**  $\text{Check}(\tilde{i}_l, \tilde{i}_r, \mathcal{D}', \Theta, k, c) = \text{true}$
  4.      $\tilde{\mathcal{D}} \leftarrow \mathcal{D}'$
  5.     **return**  $\text{RBAT}(\tilde{i}_l, \mathcal{D}', \Theta, k, c) \cup \text{RBAT}(\tilde{i}_r, \mathcal{D}', \Theta, k, c)$
  6. **else**
  7.     **return**  $\tilde{\mathcal{D}}$
- 

In step 1, RBAT splits  $\tilde{i}$  into two generalized items  $\tilde{i}_l$  and  $\tilde{i}_r$  with the help of a function *Split*. Then RBAT creates a temporary dataset  $\mathcal{D}'$  by scanning  $\tilde{\mathcal{D}}$  and mapping each item of  $\mathcal{D}$  contained in  $\tilde{i}$  to  $\tilde{i}_l$  or  $\tilde{i}_r$  (step 2). Next, in step 3, RBAT uses a function *Check* to test whether  $\mathcal{D}'$  can be safely released. If this check succeeds,  $\mathcal{D}'$  is assigned to  $\tilde{\mathcal{D}}$  and RBAT is recursively executed twice for  $\tilde{i}_l$  and  $\tilde{i}_r$ , respectively (steps 4 – 5). Otherwise,  $\tilde{\mathcal{D}}$  is returned (steps 6 – 7).

### 4.1 Split Function

*Split* divides a generalized item  $\tilde{i}$  into two disjoint subsets in a way that attempts to minimize information loss. This heuristic was selected because it outperformed a number of other heuristics we tested in terms of their ability to achieve a small amount of information loss and efficiency. Algorithm 2 describes the operation of *Split*. In step 1, we find two items  $i_j$  and  $i_m$  contained in  $\tilde{i}$  that will incur a maximum amount of information loss when generalized together, and  $g$  is a function that, given an itemset  $I$ , generates a generalized item comprised of all items of  $I$  and no other item (e.g.,  $g(ab) = (a, b)$ ). Then, in steps 2 – 3, we use  $i_j$  and  $i_m$  as “seeds” to start two new itemsets  $I_1$  and  $I_2$ . Following that, *Split* examines each item  $i_q$ ,  $q \notin \{j, m\}$  in  $\tilde{i}$  (step 4) and assigns  $i_q$  to either  $I_1$  or  $I_2$ , depending on which one results in less utility loss when it is generalized with  $i_q$  (steps 5 – 8). Last, two new (less) generalized items are created from the items in  $I_1$  and  $I_2$  and returned to RBAT (step 9).

<sup>2</sup> This is necessary for our problem to be solvable for the given parameters.

---

**Algorithm 2.** *Split*( $\tilde{i}$ )

---

**input:** Generalized item  $\tilde{i}$ **output:** Generalized items  $\tilde{i}_l$  and  $\tilde{i}_r$ 

1. find  $\{i_j, i_m\} \in \tilde{i}$  such that  $UL(g(i_j i_m))$  is maximum
  2.  $I_1 \leftarrow i_j$
  3.  $I_2 \leftarrow i_m$
  4. **for each**  $i_q \in \tilde{i}$  and  $q \notin \{j, m\}$
  5.     **if**  $UL(g(I_1 \cup i_q)) \leq UL(g(I_2 \cup i_q))$
  6.          $I_1 \leftarrow I_1 \cup i_q$
  7.     **else**
  8.          $I_2 \leftarrow I_2 \cup i_q$
  9. **return**  $\tilde{i}_l = g(I_1)$  and  $\tilde{i}_r = g(I_2)$
- 

## 4.2 Check Function

A naive implementation of *Check* would examine all the PS-rules in  $\Theta$  to test whether they are protected or not according to Definition 3. To avoid the significant computational overhead that this would require, we propose a pruning strategy that reduces the number of PS-rules that need to be examined. This strategy allows RBAT to find the same solution, but improves its efficiency by orders of magnitude (see Section 5.4). The following three theorems provide the basis for our pruning strategy (the proofs are omitted due to space limitations).

**Theorem 2.** *Given  $\tilde{\mathcal{D}}$  and PS-rules  $I \rightarrow J, I \rightarrow J'$  such that (1)  $I \rightarrow J$  is protected in  $\tilde{\mathcal{D}}$ , and (2)  $J \subseteq J'$ , it holds that  $I \rightarrow J'$  is protected in  $\tilde{\mathcal{D}}$ .*

Theorem 2 implies that we can prune all PS-rules whose antecedent is  $I$  and consequents are supersets of the consequent of a protected PS-rule  $I \rightarrow J$  in  $\tilde{\mathcal{D}}$ .

**Theorem 3.** *Given  $\tilde{\mathcal{D}}$  and PS-rules  $I \rightarrow J, I \rightarrow J'$  such that (1)  $I \rightarrow J$  is protected in  $\tilde{\mathcal{D}}$ , and (2)  $\sup(J, \mathcal{T}) \geq \sup(J', \mathcal{T})$ , where  $\mathcal{T}$  is the set of supporting transactions in  $\tilde{\mathcal{D}}$  for the itemset  $\tilde{I} = \bigcup_{v_i \in I} \Phi(i)$  of  $\tilde{\mathcal{D}}$  constructed from  $I$ , it holds that  $I \rightarrow J'$  is protected in  $\tilde{\mathcal{D}}$ .*

Consider a protected PS-rule  $I \rightarrow J$  in  $\tilde{\mathcal{D}}$ , an itemset  $\tilde{I} = \bigcup_{v_i \in I} \Phi(i)$  constructed from  $I$ , and the set  $\mathcal{T}$  of transactions supporting  $\tilde{I}$ . Theorem 3 implies that we can prune all PS-rules  $I \rightarrow J'$  whose consequents are equally or less supported than  $J$  in  $\mathcal{T}$ .

**Theorem 4.** *Given  $\tilde{\mathcal{D}}$ , a PS-rule  $I \rightarrow J$ , and a generalized item  $\tilde{i}$  in  $\tilde{\mathcal{D}}$  such that (1)  $\sup(\tilde{i}, \tilde{\mathcal{D}}) \geq k$ , (2)  $\frac{\sup(\tilde{i} \cup J, \tilde{\mathcal{D}})}{\sup(\tilde{i}, \tilde{\mathcal{D}})} \leq c$ , and (3)  $\bigcup_{v_i \in I} \Phi(i) \subseteq \tilde{i}$ , it holds that  $I \rightarrow J$  is protected in  $\tilde{\mathcal{D}}$ .*

Theorem 4 implies that we can prune all rules whose antecedent is comprised only of items that are mapped to subsets of a generalized item  $\tilde{i}$  in  $\tilde{\mathcal{D}}$ , when  $\tilde{i}$  is supported by at least  $k$  transactions in  $\tilde{\mathcal{D}}$ , at most  $c \times 100\%$  of which support  $\tilde{i} \cup J$ . To see the effectiveness of Theorems 3 and 4, assume that all public items

of a dataset  $\mathcal{D}$  are mapped to a single generalized item  $\tilde{i}$  when  $\mathcal{D}$  is generalized to  $\tilde{\mathcal{D}}$ . In this case, we can prune all PS-rules in  $\Theta$  if we know that a single PS-rule whose antecedent is comprised of all items in  $\tilde{i}$  and consequent is the most supported sensitive item in  $\tilde{\mathcal{D}}$  is protected.

The function *Check* (see Algorithm 3) is employed by RBAT to reduce the number of PS-rules to be checked based on Theorems 2, 3 and 4. In step 1, we check whether the support of either of the generalized items  $\tilde{i}_l$  and  $\tilde{i}_r$  in  $\mathcal{D}'$  is less than  $k$ . If one of them is, then there is at least one PS-rule in  $\Theta$  that will not be protected and thus *false* is returned (step 2). Otherwise, after constructing an initially empty set of protected PS-rules  $\mathcal{R}$  and initializing  $\mathcal{T}$  to the set of transactions of  $\mathcal{D}'$  that support  $\tilde{i}_l$  (steps 4 – 5), we use Theorems 2, 3, and 4 to identify PS-rules that do not need to be checked (steps 6 – 11). More specifically, we initialize  $I$  to the set of all items that are mapped to  $\tilde{i}_l$  and  $J$  to the most supported sensitive item in  $\mathcal{T}$  (steps 6 – 7) and check whether  $\text{conf}(\tilde{i}_l \rightarrow J, \mathcal{D}')$  is at most  $c$  (step 8). If this is the case,  $I \rightarrow J$  is protected in  $\mathcal{D}'$ , as we already know that  $\text{sup}(\tilde{i}_l, \mathcal{D}') \geq k$  from step 1. According to Theorem 2, all PS-rules having  $I$  as an antecedent and a superset of  $J$  as a consequent are also protected in  $\mathcal{D}'$ , and according to Theorem 4, all PS-rules whose antecedent is a subset of  $I$  and consequent is  $J$  are also protected. Furthermore, according to Theorem 3, PS-rules whose antecedent is  $I$  and consequent has a support that is less than or equal to that of  $J$  in  $\mathcal{T}$  are also protected. We add these rules to  $\mathcal{R}$  (steps 9 – 11). After that, we apply the same procedure described in steps 4 – 11 to  $\tilde{i}_r$  (steps 12 – 18). At this point,  $\mathcal{R}$  contains all PS-rules that can be pruned, so we only need to examine if each PS-rule in  $\Theta \setminus \mathcal{R}$  is protected (steps 19 – 21). If there is a PS-rule that is not protected in  $\mathcal{D}'$ , *Check* returns *false* (step 21), otherwise, it returns *true* (step 22).

### 4.3 An Example of RBAT

Consider applying RBAT to the data of Fig. 1(a). Given the most generalized item  $\tilde{i} = (a, b, c, d, e, f)$ ,  $\Theta = \{a \rightarrow \mathbf{j}, cd \rightarrow \mathbf{g}, e \rightarrow \mathbf{h}, ac \rightarrow \mathbf{hi}\}$ ,  $k = 2$ , and  $c = 0.5$ , RBAT first applies *Split* to  $\tilde{i}$ . This function finds the most “distant” items  $a, f$  and assigns them to  $I_1$  and  $I_2$  respectively. Then, *Split* assigns each of the remaining items to either  $I_1$  or  $I_2$  in a way that reduces information loss<sup>3</sup>. This results in  $\tilde{i}_l = (a, b)$  and  $\tilde{i}_r = (c, d, e, f)$ , as shown with (1) in Fig. 2, which are returned to RBAT. Next, RBAT creates the temporary dataset  $\mathcal{D}'$  of Fig. 3, in which public items are generalized to  $(a, b)$  and  $(c, d, e, f)$ , and calls *Check*.

*Check* computes the supports of  $(a, b)$  and  $(c, d, e, f)$  in  $\mathcal{D}'$  and, since both are at least 2, it constructs a PS-rule  $ab \rightarrow \mathbf{j}$  and finds  $\text{conf}((a, b) \rightarrow \mathbf{j}, \mathcal{D}')$  to be 0.5. *Check* then iterates over PS-rules in  $\Theta$  and adds  $a \rightarrow \mathbf{j}$  to  $\mathcal{R}$ . This is because  $a \rightarrow \mathbf{j}$  is the only PS-rule in  $\Theta$  whose antecedent is a subset of  $ab$ . Then, as  $\mathbf{g}$  is the most supported sensitive item in the supporting transactions of  $(c, d, e, f)$ , *Check* constructs  $cdef \rightarrow \mathbf{g}$ . Again, the confidence of  $(c, d, e, f) \rightarrow \mathbf{g}$  is 0.5, so *Check* iterates over  $\Theta$  and adds  $cd \rightarrow \mathbf{g}$  and  $e \rightarrow \mathbf{h}$  to  $\mathcal{R}$ . Next, *Check* considers  $ac \rightarrow \mathbf{hi}$ , which is the only PS-rule not contained in  $\mathcal{R}$ , and returns

<sup>3</sup> This is performed by computing *UL*, having weights set as in 7.



---

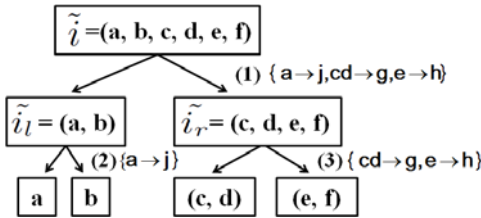
**Algorithm 3.** *Check*( $\tilde{i}_l, \tilde{i}_r, \mathcal{D}', \Theta, k, c$ )

---

**input:** Generalized items  $\tilde{i}_l, \tilde{i}_r$ , temporary dataset  $\mathcal{D}'$ , set of PS-rules  $\Theta, k, c$   
**output:** *true* if each PS-rule in  $\Theta$  is protected in  $\tilde{\mathcal{D}}$ , otherwise *false*

1. **if**  $\text{sup}(\tilde{i}_l, \mathcal{D}') < k$  or  $\text{sup}(\tilde{i}_r, \mathcal{D}') < k$
2.     **return false**
3. **else**
4.      $\mathcal{R} \leftarrow \{\}$                               $\triangleleft \mathcal{R}$  is used to store the protected PS-rules
5.      $\mathcal{T} \leftarrow$  the set of supporting transactions of  $\tilde{i}_l$  in  $\mathcal{D}'$
6.      $I \leftarrow \bigcup_{\forall i \in \tilde{i}_l} i$                       $\triangleleft I$  contains all items that are mapped in  $\tilde{i}_l$
7.      $J \leftarrow \underset{\forall i \in \mathcal{S}}{\text{argmax}} \text{sup}(i, \mathcal{T})$
8.     **if**  $\text{conf}(\tilde{i}_l \rightarrow J, \mathcal{D}') \leq c$
9.         **foreach**  $I' \rightarrow J'$  in  $\Theta$
10.             **if**  $I' \subseteq I$  and  $(J \subseteq J'$  or  $\text{sup}(J', \mathcal{T}) \leq \text{sup}(J, \mathcal{T}))$
11.                  $\mathcal{R} \leftarrow \mathcal{R} \cup \{I' \rightarrow J'\}$
12.      $\mathcal{T} \leftarrow$  the set of supporting transactions of  $\tilde{i}_r$  in  $\mathcal{D}'$
13.      $I \leftarrow \bigcup_{\forall i \in \tilde{i}_r} i$
14.      $J \leftarrow \underset{\forall i \in \mathcal{S}}{\text{argmax}} \text{sup}(i, \mathcal{T})$
15.     **if**  $\text{conf}(\tilde{i}_r \rightarrow J, \mathcal{D}') \leq c$
16.         **foreach**  $I' \rightarrow J'$  in  $\Theta$
17.             **if**  $I' \subseteq I$  and  $(J \subseteq J'$  or  $\text{sup}(J', \mathcal{T}) \leq \text{sup}(J, \mathcal{T}))$
18.                  $\mathcal{R} \leftarrow \mathcal{R} \cup \{I' \rightarrow J'\}$
19.     **foreach**  $I' \rightarrow J'$  in  $\Theta \setminus \mathcal{R}$
20.         **if**  $\text{sup}(\bigcup_{\forall i \in I'} \Phi(i), \mathcal{D}') < k$  or  $\text{conf}(\bigcup_{\forall i \in I'} \Phi(i) \rightarrow J', \mathcal{D}') > c$
21.             **return false**
22. **return true**

---



**Fig. 2.** Split of  $\tilde{i}$  by RBAT

Purchased items		
(a, b)	(c, d, e, f)	<b>g</b>
(a, b)	(c, d, e, f)	<b>h i</b>
(a, b)	(c, d, e, f)	<b>g j</b>
(c, d, e, f)	<b>g h</b>	
(a, b)	(c, d, e, f)	<b>j</b>
(c, d, e, f)	<b>i</b>	

**Fig. 3.** Temporary dataset  $\mathcal{D}'$

*true*, because the support of  $\Phi(a) \cup \Phi(c) = (a, b)(c, d, e, f)$  in Fig. 3 is at least 2 and  $\text{conf}(ac \rightarrow \mathbf{hi}, \mathcal{D}')$  does not exceed 0.5.

At this point, RBAT assigns  $\mathcal{D}'$  to  $\tilde{\mathcal{D}}$  and is executed again using each of the  $(a, b)$  and  $(c, d, e, f)$ . When RBAT runs with  $(a, b)$ , it performs split (2) in Fig. 2, and *Check* prunes  $a \rightarrow \mathbf{j}$ . Similarly, RBAT performs split (3) and prunes  $cd \rightarrow \mathbf{g}$  and  $e \rightarrow \mathbf{h}$ . Finally, RBAT returns the anonymized data of Fig. 1(b), because the PS-rules do not remain protected in the next recursion.

## 5 Experimental Evaluation

In this section, we present experimental results to confirm the effectiveness and efficiency of RBAT, including the performance of our pruning strategy.

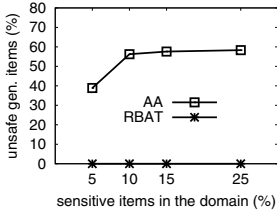
### 5.1 Experimental Setup

For our experiments, we used BMS-WebView-1 (referred to as BMS1), a real-world dataset containing click-stream data [11] that has been used in evaluating previous work [9, 4, 10]. BMS1 is comprised of 59602 transactions, whose maximum and average size are 267 and 2.5, and it has a domain size of 497.

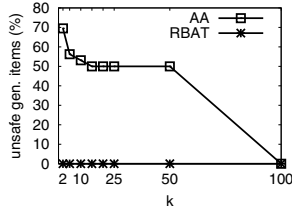
We evaluated the effectiveness and efficiency of RBAT by comparing it to the Apriori Anonymization (AA) algorithm [9]. Both algorithms were implemented in C++ and executed on an Intel 2.8GHz machine with 4GB of RAM. AA was applied to public items with  $m = 2$  using hierarchies described in [9]. The default value for parameter  $k$  is 5, the weights  $w(i_m)$  used to compute  $UL$  in RBAT are set as in [7], and, unless otherwise stated, the set of sensitive items  $\mathcal{S}$  is comprised of 10% of the most frequent items in  $\mathcal{I}$ . Furthermore,  $\Theta$  used in RBAT contains 50K PS-rules that are not protected in BMS1 and whose antecedent is a 2-itemset in  $\mathcal{P}$  and consequent an item in  $\mathcal{S}$ . Antecedents and consequents of PS-rules are selected uniformly at random. Note that this is a demanding scenario due to the large number of PS-rules because, intuitively, both the amount of information loss and the computational overhead incurred are expected to increase with the number of PS-rules.

Information loss is measured using both  $UL$  and Average Relative Error ( $ARE$ ). The latter is a widely-used measure [4, 7], which quantifies the error in answering a workload of queries on anonymized data and is independent of the way RBAT and AA work. We constructed workloads of 1000 COUNT() queries that retrieve a random 2-itemset in  $\mathcal{P}$  as in [7]. To capture protection against sensitive itemset disclosure, we measure the percentage of generalized items for which there is at least one sensitive item  $i_j$  such that  $\frac{sup(i \cup i_j, \tilde{\mathcal{D}})}{sup(i, \tilde{\mathcal{D}})} > c$  in  $\tilde{\mathcal{D}}$ . These items are called *unsafe generalized items*. We use these items to quantify protection, because their number does not depend on the number of specified PS-rules, i.e. there may be many PS-rules whose antecedent is the same unsafe generalized item, but they all suggest the same “unsafeness” in  $\tilde{\mathcal{D}}$ .

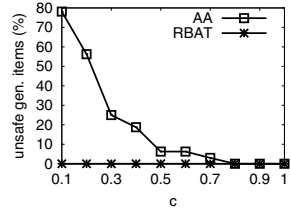
We designed three sets of experiments. The first set evaluates the algorithms in terms of their ability to provide protection against both identity and sensitive itemset disclosure, while the second one evaluates the effectiveness of the algorithms in terms of retaining data utility. In the third set, we examine the runtime efficiency of the algorithms and the effectiveness of our pruning strategy. Due to space limitations we only report a small subset of the conducted experiments.



**Fig. 4.** Unsafe gen. items vs. sens. items



**Fig. 5.** Unsafe gen. items vs.  $k$



**Fig. 6.** Unsafe gen. items vs.  $c$

## 5.2 Privacy Protection Evaluation

As shown in Section 4, RBAT is able to produce anonymized data that prevents both identity and sensitive itemset disclosure. This is in contrast to AA, which, as we experimentally verify in this section, is unable to protect data against sensitive itemset disclosure. To ensure that RBAT provides the same protection against identity disclosure as AA does and, additionally, strong protection against sensitive itemset disclosure, we created one PS-rule  $I \rightarrow J$  for each 2-itemset  $I$  in  $\mathcal{P}$  and for each 1-itemset (item)  $J$  in  $\mathcal{S}$ , and we set  $c = 0.2$ .

In our first experiment, we varied the number of sensitive items used. Fig. 4 reports the result with respect to protection, where the percentage of sensitive items varies from 5 to 25. By construction, RBAT produced anonymizations that contain no unsafe generalized items, while AA failed to offer sufficient protection in all tested cases. Specifically, the percentage of unsafe generalized items for AA was at least 38%, increasing to 58% when the percentage of sensitive items was 25%. This implies that the sensitive information of a large number of individuals represented in the data is susceptible to disclosure, particularly when a large fraction of items are sensitive.

Next, we investigated the amount of privacy protection offered by the algorithms under test using various  $k$  values in  $[2, 100]$ . As can be seen in Fig. 5, contrary to RBAT that created no unsafe generalized items, AA constructed generalized items at least 50% of which were unsafe, even when it ran with  $k = 50$ . In fact, a large  $k$  value of 100 had to be used in AA to prevent the construction of unsafe generalized items. This confirms that, guarding against sensitive itemset disclosure by applying AA with a large  $k$  value is a subpar strategy, because  $k$  needs to be enlarged to a point that it significantly harms data utility.

Last, we measured protection by varying  $c$  in  $[0.1, 1]$ . The result shown in Fig. 6 demonstrates again the superiority of RBAT - AA does not guarantee protection against sensitive itemset disclosure, as it created anonymizations in which up to 78% of generalized items are unsafe. As expected, the percentage of unsafe generalized items decreases as  $c$  increases, since a large  $c$  implies that less protection against sensitive itemset disclosure is required. However, the percentage of unsafe generalized items remains non-zero for  $c < 0.8$ , which indicates that AA is unable to generate data that prevents sensitive itemset disclosure.

### 5.3 Data Utility Evaluation

We examined the effectiveness of RBAT in terms of data utility by comparing two configurations, which use  $c = 0.7$  and  $c = 0.1$  respectively, to AA with respect to  $UL$  and  $ARE$ . RBAT with  $c = 1$  is similar to AA in that it guards against identity disclosure only. The configuration with  $c = 0.7$  is used to show the amount of data utility that RBAT needs to trade in order to thwart sensitive itemset disclosure. The unnormalized  $UL$  scores and  $ARE$  scores for various  $k$  values in  $[2, 100]$  are shown in Figs. 7 and 8 respectively. As can be seen, RBAT with  $c = 1$  retained more data utility than both AA and RBAT with  $c = 0.7$ , which verifies that *RBAT is more effective than AA in minimizing information loss*. Interestingly, RBAT with  $c = 0.7$  outperformed AA across all  $k$  values, which suggests that RBAT is able to eliminate sensitive itemset disclosure without incurring a large amount of information loss.

We then studied the effect of specifying detailed privacy requirements on data utility by considering increasingly larger sets of PS-rules. To be consistent, we required all PS-rules used in one set to be contained in all other, but larger sets used in the experiments. In this set of experiments, we used  $k = 5$  and set all other parameters as in the previous experiment. The results for  $UL$  and  $ARE$  shown in Figs. 9 and 10 respectively suggest that, due to its ability to take PS-rules into consideration when generating anonymizations, RBAT is able to retain significantly more data utility than AA. It is also interesting to observe that both  $UL$  and  $ARE$  scores for RBAT increase with the number of PS-rules. This is because a larger number of PS-rules model stronger privacy requirements, which then forces RBAT to trade off some utility for privacy in order to satisfy them. On the other hand, AA does not consider the specified PS-rules and thus the  $UL$  and  $ARE$  scores for AA remained constant.

### 5.4 Efficiency

We first compared the runtime of RBAT to that of AA by configuring RBAT using  $c = 0.7$  and varying  $k$  in  $[2, 100]$ . As shown in Fig. 11, RBAT is less efficient than AA. This is because RBAT additionally checks for the risk of sensitive itemset disclosure, which is a computationally demanding task involving support computations for a large number of itemsets 10.

Next, we studied how varying  $c$  in  $[0.1, 0.9]$  affects the runtime performance of RBAT. The result reported in Fig. 12 shows that RBAT requires less time to execute for smaller values of  $c$ , as it performs fewer recursions. We also note that RBAT runs up to 92% faster with  $c = 1$ , since *Check* only needs to compute the support of the antecedents of the rules<sup>4</sup>.

Last, we evaluated the effectiveness of our pruning strategy by examining the number of calls to the function *Check()* when RBAT runs with the pruning strategy and without it. As can be seen in Fig. 13, applying the proposed pruning strategy reduces the number of checks by several orders of magnitude, since most

<sup>4</sup> In this experiment, we excluded steps 3 – 21 from Algorithm 3.

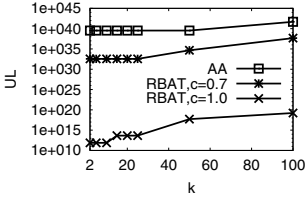


Fig. 7. *UL vs. k*

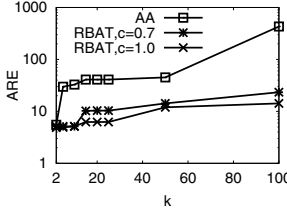


Fig. 8. *ARE vs. k*

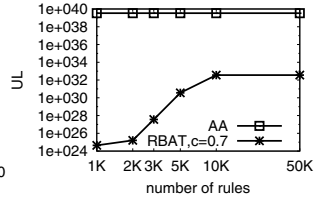


Fig. 9. *UL vs. # rules*

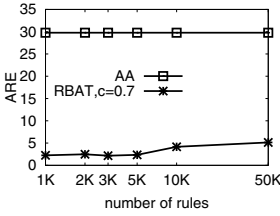


Fig. 10. *ARE vs. # rules*

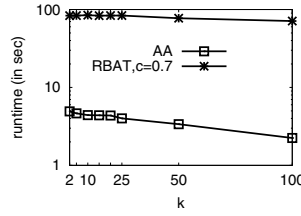


Fig. 11. *Efficiency vs. k*

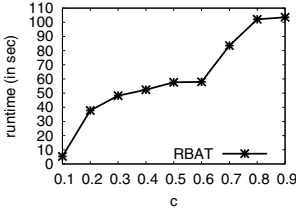


Fig. 12. *Efficiency vs c*

# PS-rules	# of calls to <i>Check()</i>	
	RBAT (no pruning)	RBAT
1K	283K	110
2K	495K	110
3K	528K	110
5K	777K	110
10K	1.26M	110
50K	1.42M	132

Fig. 13. *Pruning effectiveness*

rules are pruned in each recursion of RBAT. Furthermore, our pruning strategy improved the runtime of RBAT by at least 210%.

## 6 Conclusions

This paper proposes a novel approach to anonymizing transaction data according to detailed privacy requirements against identity and sensitive itemset disclosure. We design a privacy model that allows data owners to express a wide range of privacy requirements and incorporate it into an effective anonymization algorithm. As verified by extensive experiments, our algorithm incurs significantly less information loss than the state-of-the-art method [9] due to the flexible generalization model it employs, while guaranteeing protection against sensitive itemset disclosure. Furthermore, it addresses the computationally demanding problem of checking whether the specified requirements are satisfied in the anonymized data in an efficient way through the use of a pruning strategy.

## References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB'94, pp. 487–499 (1994)
2. Barbaro, M., Zeller, T.: A face is exposed for aol searcher no. 4417749. New York Times (August 2006)
3. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey on recent developments. *ACM Comp. Surv.* 42(4) (2010)
4. Ghinita, G., Tao, Y., Kalnis, P.: On the anonymization of sparse high-dimensional data. In: ICDE '08, pp. 715–724 (2008)
5. Gkoulalas-Divanis, A., Verykios, V.S.: Exact knowledge hiding through database extension. *IEEE TKDE* 21(5), 699–713 (2009)
6. He, Y., Naughton, J.F.: Anonymization of set-valued data via top-down, local generalization. *PVLDB* 2(1), 934–945 (2009)
7. Loukides, G., Gkoulalas-Divanis, A., Malin, B.: Coat: Constraint-based anonymization of transactions. *CoRR*, abs/0912.2548, Technical Report (2009)
8. Samarati, P.: Protecting respondents identities in microdata release. *IEEE TKDE* 13(9), 1010–1027 (2001)
9. Terrovitis, M., Mamoulis, N., Kalnis, P.: Privacy-preserving anonymization of set-valued data. *PVLDB* 1(1), 115–125 (2008)
10. Xu, Y., Wang, K., Fu, A.W.-C., Yu, P.S.: Anonymizing transaction databases for publication. In: KDD '08, pp. 767–775 (2008)
11. Zheng, Z., Kohavi, R., Mason, L.: Real world performance of association rule algorithms. In: KDD '01, pp. 401–406 (2001)

# Interval-Orientation Patterns in Spatio-temporal Databases

Dhaval Patel

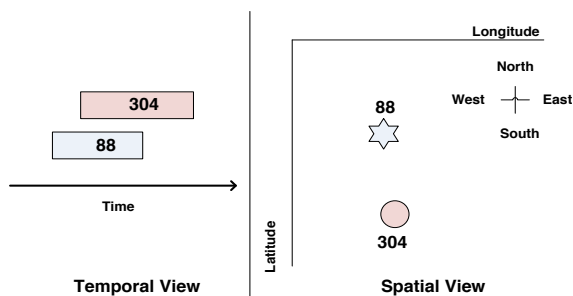
National University of Singapore  
dhaval@comp.nus.edu.sg

**Abstract.** In this paper, we present a framework to discover a spatio-temporal relationship patterns. In contrast to previous work in this area, features are modeled as durative rather than instantaneous. Our method takes into account feature’s duration to capture the temporal influence of a feature on other features in spatial neighborhood. We have developed an algorithm to discover a temporal-spatial feature interaction patterns, called the *Interval-Orientation Patterns*. Interval-Orientation pattern is a frequent sequence of features with annotation of temporal and directional relationships between every pairs of features. The proposed algorithm employs Hash-based joining technique to improve the efficiency. We also extend our approach to accommodate an incremental mining as updates in real world spatio-temporal databases are common. The incremental algorithm employs an optimization that is based on previously generated patterns to prune the non-promising candidates early. We evaluate our algorithms on synthetic dataset to demonstrate its efficiency and scalability. We also present the patterns identified from real world drought, vegetation and video action databases. We also show that the patterns discovered from video dataset can improve the classification accuracy of activity recognition.

## 1 Introduction

Pattern mining in spatio-temporal databases aims to discover useful spatio-temporal relationships that are hidden among features. Existing algorithms [14,16,17,18,6,10] have focused on discovering patterns from instantaneous features, that is, features with no duration. This assumption allows the discovered pattern to be simplified to a set of features such as {NormalTemp, HighPrecepitation, NoDrought} or an ordered sequence of features such as {NormalTemp  $\rightarrow$  HighPrecepitation  $\rightarrow$  NoDrought}. However, features in many real world spatio-temporal databases are durative. For example, spatio-temporal features extracted from video, climate or image dataset such as STIP descriptors [8], SaTScan descriptors [7,9], Cuboid descriptors [4] and etc. have location and duration information. Using location and duration information of features, we can mine useful spatio-temporal relationships.

For example, consider a spatio-temporal pattern,  $\langle \{88 \rightarrow 304\}, \{\text{Overlap}\}, \{\text{North}\} \rangle$ , shown in Fig 1. This pattern, obtained from KTH video dataset [4], indicates that feature 88 overlaps feature 304 in time and feature 88 is in the nearby “North” region of feature 304. A detailed investigation revealed that this pattern occurs in 6 videos having walking activity, 1 video having running activity and 1 video having



**Fig. 1.** Spatio-temporal relationship between STIP features in video dataset

jogging activity. In short, this pattern help to discriminate a walking activity with running or jogging activities. Moreover, our experiments reveal that such patterns leads to increase in the classification accuracy of activity recognition. Similarly, we observe that earthquake occurs “during” high atmospheric pressure occurring in the nearby “North” region<sup>[1]</sup>. This insight is useful in the development of effective earthquake prediction.

In this paper, we define a new class of pattern called Interval-Orientation Pattern, in short IO pattern, to capture the important spatio-temporal relationship among features. IO pattern is an ordering of features having annotation of temporal and directional relationships among all pairs of features in the sequence. As our dataset is non-transactional, we use prevalence index<sup>[13]</sup> to calculate the support of IO pattern. Moreover, updates in real world databases are common. For example, remote sensing images are captured daily in multitemporal satellite image dataset(<http://www.sat.dundee.ac.uk>), monthly mean temperature and precipitation are captured at each weather stations in National Climatic Data Center([www.ncdc.noaa.gov](http://www.ncdc.noaa.gov)). Such update may introduce new patterns or invalid some existing patterns. Recomputing frequent IO patterns from scratch is time consuming process. Thus, we suggest an efficient incremental method to maintain the discovered patterns. The key contributions of this work are summarized as follow:

1. We define a new class of pattern to capture the invariant ordering among features with annotation of temporal and spatial relationship between each pair of features. An efficient algorithm called IOMiner is designed to mine complete set of frequent IO patterns. IOMiner is two-stage algorithm. The first stage adapts disjoint cubes based hashing<sup>[15]</sup> and efficiently discovers length 2 IO patterns. In the second stage, we use Hash-based join to find the longer length IO patterns.
2. Our incremental approach, called as IncIOMiner(Incremental IOMiner), is constructed under the framework defined by IOMiner. IncIOMiner prunes the search space by estimating the upper bound of prevalence index using previous computations. A Pattern Tree is used to maintain all previous computations.
3. We evaluate our framework on synthetic and real-world dataset. Experiments on synthetic dataset show scalability and efficiency of the proposed approach. We

<sup>1</sup> [http://www.usatoday.com/weather/research/2009-11-12-hquakeweather12\\_ST\\_N.htm](http://www.usatoday.com/weather/research/2009-11-12-hquakeweather12_ST_N.htm)



discover IO patterns from drought, vegetation and video dataset. Further, we show that the patterns discovered from video dataset can improve the classification accuracy of activity recognition.

## 2 Preliminaries

Let  $F$  is a set of  $k$  features;  $F = \{f_1, f_2, \dots, f_k\}$ . Let  $D$  be a set of feature instances, where each feature instance is given by a tuple  $\langle \text{identifier, feature, \{latitude, longitude\}, \{start\_time, end\_time\}} \rangle$ . We use identifier,  $d_i$ , to refer a feature instance and  $d_i.start\_time$  to refer the start time of  $d_i$ . Figure 2(a) lists the features instances used as working example in this paper. We use  $D_{0,t}$  to denote a set of feature instances having  $start\_time \geq 0$  and  $end\_time < t$ . This dataset is updated with a set of feature instances having  $start\_time \geq t$  and  $end\_time < (t + \Delta t)$  denoted as  $D_{t,t+\Delta t}$ .

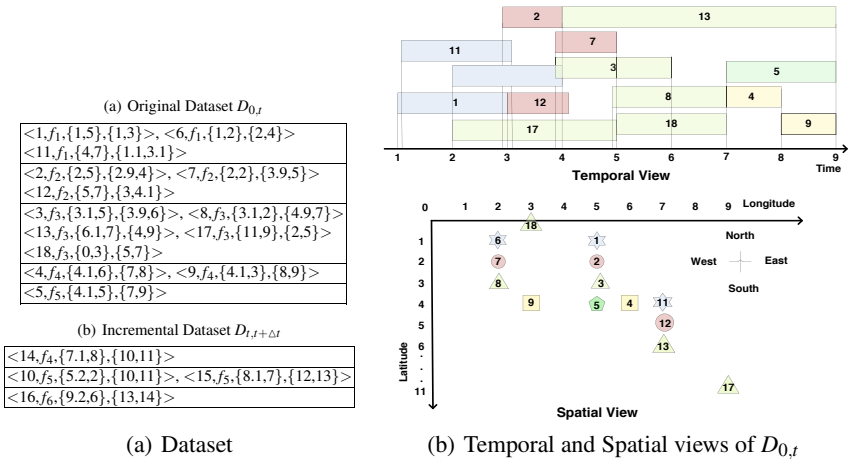


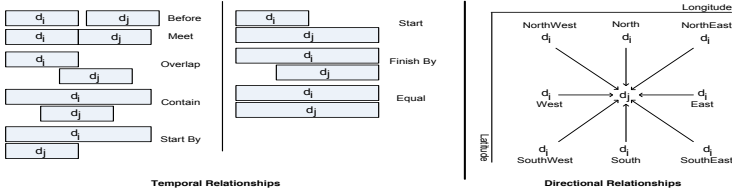
Fig. 2. Working Dataset

Two feature instances  $d_i$  and  $d_j$  are **neighbor in time** domain if  $(\min\{d_i.end\_time, d_j.end\_time\} - \max\{d_i.start\_time, d_j.start\_time\}) \geq 0$  or  $\min\{|d_i.end\_time - d_j.start\_time|, |d_i.start\_time - d_j.end\_time|\} \leq T_\delta$ , where  $T_\delta$  is a time threshold. For example, when  $T_\delta = 1$ , instances 1 and 2 are neighbor in time but instances 1 and 18 are not.

Two feature instances  $d_i$  and  $d_j$  are **neighbor in space** domain if  $Dist(d_i, d_j) \leq R_\delta$ , where  $R_\delta$  is a distance threshold. In this paper,  $Dist(d_i, d_j)^2 = (d_i.latitude - d_j.latitude)^2 + (d_i.longitude - d_j.longitude)^2$ . For example, when  $R_\delta = 2$ , instances 1 and 2 are **neighbor in space** but instances 1 and 13 are not.

The temporal relationship of feature instance  $d_i$  with respect to feature instance  $d_j$ , denoted as  $TR(d_i, d_j)$ , is one of the eight values : **Equal, Meet, Overlap, Contain, Before, Start, Start\_By, Finish\_By**. For example,  $TR(1, 2) = O$  and  $TR(1, 13) = B$ .

The directional relationship of feature instance  $d_i$  with respect to feature instance  $d_j$ , denoted as  $DR(d_i, d_j)$ , is one of the nine values : **North, South, East, West**,



**Fig. 3.** Temporal and Directional Relationships of instance  $d_i$  w.r.t. instance  $d_j$

NorthEast, NorthWest, SouthEast, SouthWest and ND. ND represent a situation when both instances are from same location. For example,  $DR(1, 2) = N$  and  $DR(1, 18) = SE$ . Fig. 3 shows the temporal and directional information of  $d_i$  with respect to  $d_j$ .

The temporal relationship between two features  $f_i$  and  $f_j$  (denoted as  $TR(f_i, f_j)$ ) also takes one of the eight values as described earlier. However,  $TR(f_i, f_j)$  is just an abstract specification. Similarly,  $DR(f_i, f_j)$  is an abstract directional relationship specification between features  $f_i$  and  $f_j$  and also takes one of the nine directional values.

We define an **interval-orientation** pattern as a sequence of features with abstract temporal and directional relationship specifications between every pairs of features. It is given as  $P = \langle \{f_1 \rightarrow f_2 \rightarrow \dots \rightarrow f_n\}, \{TR(f_1, f_2), \dots, TR(f_1, f_n), TR(f_2, f_3), \dots, TR(f_2, f_n), \dots, TR(f_{n-1}, f_n)\}, \{DR(f_1, f_2), \dots, DR(f_1, f_n), DR(f_2, f_3), \dots, DR(f_2, f_n), \dots, DR(f_{n-1}, f_n)\} \rangle$ . The length of IO pattern  $P$ , in short  $|P|$ , is the number of features in the sequence. For example,  $P = \langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$  is a length 2 pattern.

A **pattern instance** of length  $n$  IO pattern  $P$  is a sequence of  $n$  feature instances from  $D$ , denoted as  $I = \{i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow \dots \rightarrow i_n\}$ , such that  $I$  satisfies following conditions

- feature of  $j^{th}$  instance in  $I$  is equal to  $j^{th}$  feature in  $P$
- start time of  $j^{th}$  instance in  $I \geq$  start time of all  $i^{th}$  instances in  $I$ , where  $i < j$
- adjacent instances in  $I$  must be neighbor in **time** and **space**
- the temporal and directional relationship between every pairs of instances in  $I$  conform to those specified in  $P$
- instances in  $I$  are unique

For example,  $I = \{1 \rightarrow 2\}$  is a pattern instance of  $P = \langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$  as feature of instance 1 (instance 2) is  $f_1$  ( $f_2$ ), the start time of instance 1 is earlier than start time of instance 2, instance 1 and 2 are neighbor in time and space,  $TR(1, 2) = O$  and  $DR(1, 2) = N$ . Similarly,  $\{6 \rightarrow 7\}$  and  $\{11 \rightarrow 12\}$  are two other pattern instances of  $P$ .

Given an IO pattern  $P$ , let  $ISet$  is a set of all **pattern instances** of  $P$ . We use  $ISet_i$  to refer  $i^{th}$  pattern instance in  $ISet$  and  $ISet_{i,j}$  to refer  $j^{th}$  feature instance from pattern instance  $ISet_i$ . Also,  $ISet_{*,j}$  is a set of  $j^{th}$  feature instances from all pattern instances in  $ISet$ . The **participation ratio** of  $j^{th}$  feature  $f_j$  in IO pattern  $P$ , denoted as  $p\_ratio(f_j, P)$ , is defined as:

$$p\_ratio(f_j, P) = \frac{\# \text{ unique instances of } f_j \text{ in } ISet_{*,j}}{\# \text{ instances of } f_j \text{ in } D} \quad (1)$$

The **prevalence index** of  $P$ , denoted as  $pi(P)$ , is the minimum participation ratio of the features present in pattern  $P$ . Formally,  $pi(P) = \text{minimum}\{p\_ratio(f_i, P) \text{ where } f_i \in P\}$ . A pattern  $P$  is **frequent** if  $pi(P) \geq \text{min\_pi}$ .

For example, consider a pattern  $P = \langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$ .  $ISet = \{\{1 \rightarrow 2\}, \{6 \rightarrow 7\}, \{11 \rightarrow 12\}\}$  is a set of all pattern instances of  $P$ .  $ISet_1$  refers to  $\{1 \rightarrow 2\}$ .  $ISet_{1,2} = 2$ ,  $ISet_{*,1} = \{1,6,11\}$  and  $ISet_{*,2} = \{2,7,12\}$ .  $p\_ratio(f_1, P) = \frac{3}{3}$  and  $p\_ratio(f_2, P) = \frac{3}{3}$ . Hence,  $pi(P) = \text{minimum}\{1, 1\} = 1$ .

**Problem Statement:** Given a set of spatio-temporal features  $D_{0,t}$ , a distance threshold  $R_\delta$ , a time threshold  $T_\delta$  and minimum prevalence index threshold  $min\_pi$ , we aim to find all frequent IO patterns efficiently. Further, given an incremental database  $D_{t,t+\Delta t}$  to the existing database  $D_{0,t}$ , find all frequent IO patterns in  $D_{0,t+\Delta t} = \{D_{0,t} \cup D_{t,t+\Delta t}\}$ , with minimum possible recomputations.

### 3 Algorithm IOMiner

Algorithm IOMiner is two-stage algorithm; the first stage discovers length 2 IO patterns and second stage uses length 2 IO patterns to extend the length  $l(\geq 2)$  pattern into length  $l+1$  pattern. Algorithm 1 outline IOMiner. In Line 1, we generate all length 2 IO patterns from given database  $D$ . The generated frequent patterns are stored in  $fre\_2\_Set$  and  $ext\_PatSet$ . Next, each frequent pattern from  $ext\_PatSet$  is extended into longer length patterns(Lines 3-8). Now, we explain how length 2 patterns are discovered and extended further. The input parameter is  $D = D_{0,t}$ ,  $T_\delta = 1$ ,  $R_\delta = 2$  and  $min\_pi = 0.50$ .

---

#### Algorithm 1. IOMiner( $D, min\_pi, T_\delta, R_\delta$ )

---

**Output:**  $patSet$  = frequent IO patterns

- 1  $fre\_2\_Set = \{\text{Discover length 2 IO patterns from } D\}$
- 2  $patSet = fre\_2\_Set, ext\_PatSet = fre\_2\_Set$
- 3 **while**  $ext\_PatSet \neq \emptyset$  **do**
- 4     Select pattern  $P$  from  $ext\_PatSet$  and remove  $P$  from  $ext\_PatSet$
- 5      $ext\_Set = \{\text{Extend } P\}$
- 6      $ext\_PatSet = ext\_Set \cup ext\_PatSet$
- 7     Add  $P$  to  $patSet$
- 8 **end**
- 9 **return**  $patSet$

---

#### 3.1 Discover Length 2 Patterns

The naive method to generate length 2 patterns is candidate set generation and test approach. However, this approach generates total  $N^2 * 8 * 8$  length 2 candidate patterns when number of features are  $N$ . Clearly, this is not a feasible solution.

Our method first hashes all feature instances from  $D$  into disjoint cubes. We use latitude, longitude and start time information of instance for hashing. In particular, space-time dimensions(i.e.,  $\{\text{latitude} \times \text{longitude} \times \text{start time}\}$ ) are divided into set of disjoint cubes  $\{\langle x_1, y_1, t_1 \rangle, \langle x_2, y_2, t_2 \rangle, \dots, \langle x_p, y_p, t_p \rangle, \dots, \langle x_q, y_q, t_q \rangle\}$ . A disjoint window of width  $\frac{R_\delta}{2}$  is used to divide each space dimension and disjoint window of width  $T_\delta$  is used to divide the time dimension. Figure 4(a) represents hashing of instances given in Fig. 2(a)(a).

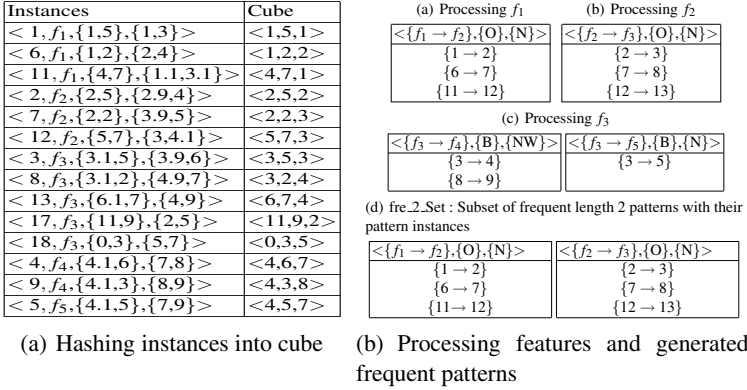


Fig. 4. Discovering length 2 IO patterns

Given a feature instance  $d_i \in D$ , we use hashing to discover all other instances  $d_j$  from  $D$  such that  $d_i$  and  $d_j$  are neighbor in time and space as well as  $d_i.start\_time \leq d_j.start\_time$ . Let  $d_i$  is in cube  $\langle x_m, y_m, t_n \rangle$ . All instances satisfying above conditions with  $d_i$  must be in one of the cubes  $\langle x_i, y_i, t_s \rangle$  in the set  $N_{\langle x_m, y_m, t_n \rangle} = \{ \langle x_i, y_i, t_s \rangle \mid (|x_i - x_m| \leq 2) \wedge (|y_i - y_m| \leq 2) \wedge (t_n \leq t_s) \wedge (d_i.end\_time + T_\delta \geq (t_s * T_\delta)) \}$ . We refer  $N_{\langle x_m, y_m, t_n \rangle}$  as neighbor-set of the cube  $\langle x_m, y_m, t_n \rangle$ . For example, consider an feature instance 4. It belongs to cube  $\langle 4, 6, 7 \rangle$ . Hence, other feature satisfying above conditions must be in cubes  $\langle 4, 6, 7 \rangle$  and  $\langle 4, 5, 7 \rangle$  w.r.t. our working example. Note that, cube  $\langle 4, 7, 1 \rangle$  is not selected as start time of instances in  $\langle 4, 7, 1 \rangle$  are earlier than the start times of instances in cube  $\langle 4, 6, 7 \rangle$ .

Now, we discuss the basic steps to find length 2 IO patterns. We process each feature separately. In our example, we have total five features  $F = \{f_1, f_2, f_3, f_4, f_5\}$ . Consider a feature  $f_1 \in F$ . It's instances are 1, 6 and 11 in  $D$ . We process each of these instances one by one.

- While processing instance 1, we obtain that it belongs to cube  $\langle 1, 5, 1 \rangle$  and it's  $N_{\langle 1, 5, 1 \rangle} = \{ \langle 1, 5, 1 \rangle, \langle 2, 5, 2 \rangle, \langle 3, 5, 3 \rangle \}$ . We form pattern instances between instance 1 and instances from cube in  $N_{\langle 1, 5, 1 \rangle}$ . Here, cubes  $\langle 1, 5, 1 \rangle$ ,  $\langle 2, 5, 2 \rangle$  and  $\langle 3, 5, 3 \rangle$  contain instances 1, 2 and 3 respectively. Thus, the resultant pattern instances are  $\{1 \rightarrow 1\}$ ,  $\{1 \rightarrow 2\}$  and  $\{1 \rightarrow 3\}$ . we use these pattern instances to generate candidate length 2 patterns. Using  $\{1 \rightarrow 2\}$ , we generate  $P = \langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$ . We also maintain  $\{1 \rightarrow 2\}$  as a pattern instance of  $P$ . Next,  $\{1 \rightarrow 3\}$  is not valid pattern instance since  $Dist(1, 3) > R_\delta$ . Also,  $\{1 \rightarrow 1\}$  is not valid pattern instance as it contains instance 1 more than one time. As a result, we do not generate any candidate pattern for  $\{1 \rightarrow 1\}$  and  $\{1 \rightarrow 3\}$ .
- Next, we process instance 6. Instance 6 belongs to cube  $\langle 1, 2, 2 \rangle$  and it's  $N_{\langle 1, 2, 2 \rangle} = \{ \langle 1, 2, 2 \rangle, \langle 2, 2, 3 \rangle, \langle 3, 2, 4 \rangle, \langle 0, 3, 5 \rangle \}$ . This cubes contain feature instances 6, 7, 8 and 18. Hence, we generate candidate patterns using pattern instances  $\{6 \rightarrow 6\}$ ,  $\{6 \rightarrow 7\}$ ,  $\{6 \rightarrow 8\}$  and  $\{6 \rightarrow 18\}$ . Pattern instance  $\{6 \rightarrow 7\}$  generates  $\langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$ . As, this pattern is already generated, we append  $\{6 \rightarrow 7\}$  as

it's pattern instance. We do not process  $\{6 \rightarrow 6\}$  as explained previously. Remaining pattern instances are not neighbor in space.

- Instance 11 belongs to cube  $\langle 4, 7, 1 \rangle$  and it's  $N_{\langle 4, 7, 1 \rangle} = \{\langle 4, 7, 1 \rangle, \langle 5, 7, 3 \rangle, \langle 6, 7, 4 \rangle, \langle 2, 5, 2 \rangle\}$ . We follow similar process as explained previously.

Once all instances of feature  $f_1$  are processed, we calculate  $pi$  of each generated patterns. Those patterns satisfying  $min\_pi$  threshold are stored in  $fre\_2\_Set$  with their pattern instances. In similar manner, we process other features from  $F$ . Figure 4(b) lists generated candidate patterns using features  $f_1, f_2$  and  $f_3$ .

### 3.2 Extend IO Pattern

We extend IO pattern  $P(\in ext\_PatSet)$  by length 2 pattern  $Q(\in fre\_2\_Set)$  if last feature of  $P$  is same as the first feature of  $Q$ . The extension process generates candidate patterns and it's pattern instances. This method goes as follow: a pattern instance  $p_i$  of  $P$  is joined with pattern instance  $q_i$  of  $Q$ , if last feature instance in  $p_i$  matches with the first feature instance in  $q_i$ . Joining  $p_i$  with  $q_i$  generates one candidate pattern and it's pattern instance.

For example, consider extension of IO pattern  $P = \langle \{f_1 \rightarrow \underline{f_2}\}, \{O\}, \{N\} \rangle$  (See Figure 4(b)(d)). This pattern will be extended by  $Q = \langle \{\underline{f_2} \rightarrow f_3\}, \{O\}, \{N\} \rangle$ . During extension, a pattern instance  $\{1 \rightarrow \underline{2}\}$  from  $P$  is joined with pattern instance  $\{\underline{2} \rightarrow 3\}$  from  $Q$  and generate pattern instance  $\{1 \rightarrow 2 \rightarrow 3\}$ . Using  $\{1 \rightarrow 2 \rightarrow 3\}$ , we generate a pattern  $\langle \{f_1 \rightarrow f_2 \rightarrow f_3\}, \{O, B, O\}, \{N, N, N\} \rangle$  with pattern instance  $\{1 \rightarrow 2 \rightarrow 3\}$ . Similarly, other possible pattern instances are  $\{6 \rightarrow 7 \rightarrow 8\}$  and  $\{11 \rightarrow 12 \rightarrow 13\}$ . Both pattern instances suggest to generate candidate pattern  $\langle \{f_1 \rightarrow f_2 \rightarrow f_3\}, \{O, B, O\}, \{N, N, N\} \rangle$ . Once all possible joining are done between pattern instances of  $P$  and  $Q$ ,  $pi$  is calculated for each generated patterns. For the current case, only one pattern in generated and it is also frequent pattern. We follow similar process if  $P$  can be extended by any other pattern from  $fre\_2\_Set$ . All generated frequent patterns are stored in  $ext\_PatSet$  for further extension.

Note that, when we extend a pattern  $P$  by length 2 pattern  $Q$ , we join pattern instances of  $P$  with pattern instances of  $Q$ . In this paper, we use Hash-based joining method to improve the efficiency. In particular, we build a hash table for each pattern  $Q$ ,  $Q \in fre\_2\_Set$  after finishing the first stage. Hash table of pattern  $Q$  hashes the pattern instance of  $Q$  using it's first feature instance as a key.

## 4 Algorithm IncIOMiner

We have applied IOMiner to database  $D_{0,t}$ . Now,  $D_{0,t}$  is updated by  $D_{t,t+\Delta t}$ . The main motivation of IncIOMiner is to improve the time complexity by avoiding unnecessary computations. To achieve this goal, all generated patterns from  $D_{0,t}$  are maintained in pattern tree  $PT$  (See Figure 5). The pattern with solid rectangle are frequent patterns. The negative border pattern, represented as dotted rectangle, is infrequent pattern but its prefix is frequent. Observe that, each pattern  $P$  in  $PT$  stores information derived from it's pattern instances. For example, consider a pattern  $P = \langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$

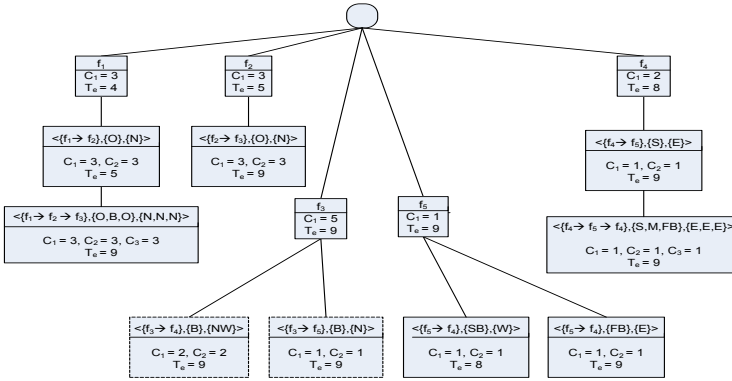


Fig. 5. PT: Pattern Tree

in  $PT$  and it's set of pattern instances  $ISet$  in Figure 4(b).  $C_i$  denotes the number of unique feature instances in  $ISet_{*,j}$ . Hence,  $C_1 = 3$  and  $C_2 = 3$ .  $T_e$  is the end time of feature instance  $d_i \in ISet_{*,|P|}$  such that  $d_i.end\_time$  is greater than end time of any other instances from  $ISet_{*,|P|}$ . Recall,  $|P|$  is the length of  $P$ .

---

**Algorithm 2.**  $InciOMiner(D_{0,t}, D_{t,t+\Delta t}, min\_pi, T_\delta, R_\delta, PT)$ 


---

**Output:**  $patSet =$  frequent temporal patterns

- 1  $[fre\_2\_Set, ext\_Set] = \text{GetExtendExtensionPattern}(D_{0,t}, D_{t,t+\Delta t}, PT, min\_pi, T_\delta, R_\delta)$
  - 2  $patSet = fre\_2\_Set$
  - 3 **while**  $ext\_Set \neq \emptyset$  **do**
  - 4     Select pattern  $P$  from  $ext\_Set$  and remove  $P$  from  $ext\_Set$
  - 5      $new\_Set = \{\text{Extend } P\}$
  - 6      $ext\_Set = ext\_Set \cup new\_Set$
  - 7     Add  $P$  to  $patSet$
  - 8 **end**
  - 9  $canSet = \{\text{patterns } P \mid P \in PT \wedge \text{no new pattern instance of } P \text{ is generated}\}$
  - 10  $freSet = \{\text{calculate } pi \text{ of each pattern in } canSet \text{ and find frequent patterns}\}$
  - 11 **return**  $\{patSet \cup freSet\}$
- 

Algorithm 2 outline the details. In Line 1, we obtain  $fre\_2\_Set$  and  $ext\_Set$ . Here,  $fre\_2\_Set$  is set of those frequent length 2 patterns for which new pattern instance is generated after the database update. The pattern from  $fre\_2\_Set$  is used for **extension**. Also,  $ext\_Set$  is a set of those frequent patterns for which new instance are created or there is a chance that new instance will be created on extension. A pattern  $P$  from  $ext\_Set$  will be extended using patterns from  $fre\_2\_Set$  (Similar to Section 3.2). Also generated patterns are stored in  $ext\_Set$  for further extension (Lines 3-7). Finally, Lines 9-10 process those patterns from  $PT$  for which no new pattern instance is generated after the update. For such pattern,  $PT$  contains all information to calculate it's  $pi$  value.

**4.1 Obtain  $fre\_2\_Set$**

Proposition 41 explains how frequent patterns used for extension are obtained.

**Proposition 41.** *Let  $D'$  represents a set of all feature instances from  $D_{0,t}$  having  $end\_time \geq t - T_\delta$ . Let  $E'$  and  $E$  denote the set of all features having instances in  $D'$  and  $D_{t,t+\Delta t}$  respectively. Let  $P = \langle \{f_1 \rightarrow f_2\}, \{TR(f_1, f_2)\}, \{DR(f_1, f_2)\} \rangle$  is length 2 pattern. When database  $D_{0,t}$  is updated to  $D_{0,t+\Delta t}$ , new pattern instance of  $P$  may be generated if one of the following condition is satisfied:*

- (i) both features  $f_1$  and  $f_2$  are present in  $E$ .
- (ii) feature  $f_1$  is present in  $E'$  and feature  $f_2$  is present in  $E$ .

*Proof.* Any pair of feature instances from incremental data  $D_{t,t+\Delta t}$  generates new pattern instance. As  $E$  is a set of features that has instance in  $D_{t,t+\Delta t}$ , a new pattern instance of  $P = \langle \{f_1 \rightarrow f_2\}, \{TR(f_1, f_2)\}, \{DR(f_1, f_2)\} \rangle$  may be generated if  $E$  contains both  $f_1$  and  $f_2$ . Next, a pair of instances where first instance is from  $D_{0,t}$  and second instance if from  $D_{t,t+\Delta t}$  might generate new pattern instance for  $P$ . As instances in  $D_{t,t+\Delta t}$  have start time  $\geq t$ , only those instances from  $D_{0,t}$  having end time  $\geq (t - T_\delta)$  are close in time with any instance from  $D_{t,t+\Delta t}$ . In other word, a pattern  $P$  having first feature from  $E'$  and second feature from  $E$  might generate new pattern instance. □

Following proposition 41 we use feature instances from  $\{D' \cup D_{t,t+\Delta t}\}$  and obtain a set of those length 2 patterns for which new pattern instance is generated after the database update. In our case,  $D' = \{4,5,9,13\}$ . The generated patterns are stored in  $can\_2\_Set$  (See second column in Table 1). Our next step is to obtain frequent patterns from  $can\_2\_Set$ . In short, we have to obtain pattern instances of each pattern in  $can\_2\_Set$  using complete data  $D_{0,t+\Delta t}$ . As we have already generated pattern instances using  $\{D' \cup D_{t,t+\Delta t}\}$ , we generate pattern instances of each pattern in  $can\_2\_Set$  using  $D_{0,t}$ . (i.e., original dataset). However, not all patterns in  $can\_2\_Set$  require to regenerate it's pattern instances from  $D_{0,t}$ . Now, we present a technique for estimating upper bound of  $pi(P)$ ,  $P \in can\_2\_Set$ .

**Table 1.** Analysis of length 2 patterns obtained using proposition 41

Sr. No.	$P \in can\_2\_Set$	Instances using proposition 41	Info. of $P$ in $PT$ $\{C_1, C_2\}$	estimated $p\_ratio(f_i, P)$	$UB\_pi(P)$
1	$\langle \{f_4 \rightarrow f_5\}, \{B\}, \{NE\} \rangle$	$\{9 \rightarrow 10\}, \{14 \rightarrow 15\}$	-	$\{\frac{2}{3}, \frac{2}{5}\}$	0.50
2	$\langle \{f_5 \rightarrow f_6\}, \{M\}, \{NE\} \rangle$	$\{15 \rightarrow 16\}$	-	$\{\frac{1}{3}, \frac{1}{7}\}$	0.33
3	$\langle \{f_3 \rightarrow f_4\}, \{B\}, \{NW\} \rangle$	$\{13 \rightarrow 14\}$	$\{2, 2\}$	$\{\frac{3}{5}, \frac{3}{7}\}$	0.60
4	$\langle \{f_3 \rightarrow f_5\}, \{B\}, \{N\} \rangle$	$\{13 \rightarrow 15\}$	$\{1, 1\}$	$\{\frac{2}{5}, \frac{2}{7}\}$	0.40

Given a length 2 pattern  $P$  from  $can\_2\_Set$  and it's pattern instances set  $I$  generated using  $\{D' \cup D_{t,t+\Delta t}\}$ , one of the following two cases is possible:

**Case :**  $P \notin PT$ . There is no pattern instance of  $P$  in  $D_{0,t}$ . In short, we already have true  $p\_ratio$  of each feature in  $P$ . For this case, we put  $P$  in  $fre\_2\_Set$  if  $pi(P) \geq min\_pi$ . We also record information about  $P$  in  $PT$  and remove it from  $can\_2\_Set$ .

**Case :**  $P \in PT$ . We estimate upper bound of  $p\_ratio$  for each feature in  $P$  (See Equation 2) and then finally obtain  $pi(P)$  (See Equation 3). If  $UB\_pi(P) < min\_pi$ , we update  $C_i$  and  $T_e$  of  $P$  in  $PT$  and remove  $P$  from  $can\_2\_Set$ . However, if  $UB\_pi(P) \geq min\_pi$ , we have to obtain its pattern instances from  $D_{0,t}$ .

$$p\_ratio(f_i, P) = \frac{C_i + \{\# \text{ of unique instance in } I_{*,i}\}}{\# \text{ of instance of } e_i \text{ in } D_{0,t+\Delta t}} \quad (2)$$

$$UB\_pi(P) = \text{minimum}\{p\_ratio(f_i, P)\} \quad (3)$$

Table 1 present an analysis of proposed technique on our working example. Note that, first two patterns satisfy first case (they are not generated in previous round) and last two patterns satisfy second case. Clearly, pattern 1 qualify  $min\_pi$  threshold and is kept in  $fre\_2\_Set$ . pattern 2 and 4 do not qualify  $min\_pi$  threshold and thus removed from  $new\_2\_Set$ . We generate valid instances for the remaining patterns in  $new\_2\_Set$  by scanning database  $D_{0,t}$  and those patterns satisfy  $min\_pi$  threshold requirement are later transferred to  $fre\_2\_Set$ .

## 4.2 Obtain $ext\_Set$

Proposition 42 explain how we obtain IO patterns which will be **extended** further.

**Proposition 42.** *Let frequent pattern  $R$  is generated by extending pattern  $P$  with length 2 pattern  $Q$ . A new pattern instance of pattern  $R$  is generated after the database update if at least one of the following condition is satisfied:*

- (i) *The value of  $P.T_e$  in  $PT \geq (t - T_\delta)$*
- (ii) *at least one pattern instance is generated for pattern  $P$  after the database update*

*Proof.* Let appending  $D_{t,t+\Delta t}$  to  $D_{0,t}$  generate at least one pattern instance  $v$  for pattern  $R$ . Careful observation suggest that, at least one feature instance in  $v$  must be from incremental part  $D_{t,t+\Delta t}$ . Otherwise,  $v$  must be generated in previous iteration.

If only last feature instance in  $v$  is from  $D_{t,t+\Delta t}$ , then immediate prefix of  $R$  (i.e.,  $P$ ) must be from previous database  $D_{0,t}$ . However, The value of  $P.T_e$  must be  $\geq (t - T_\delta)$  to qualify  $T_\delta$ . If more than one feature instances of  $v$  are from updated dataset  $D_{t,t+\Delta t}$  then immediate prefix of  $R$  (i.e.,  $P$ ) has at least one pattern instance generated after the database update.  $\square$

To obtain the patterns satisfying condition 1, we traverse  $PT$  and obtain those patterns having  $(t - T_e) \leq t_\delta$  and put them in  $ext\_Set$ . Further,  $fre\_2\_Set$  contains prefix of those patterns which satisfy the second condition. Finally,  $ext\_Set = ext\_Set \cup fre\_2\_Set$ . Table 2 presents  $fre\_2\_Set$  and  $ext\_Set$  obtained for our working example.

Algorithm 3 summarize the details. We generate length 2 patterns using Proposition 41 (Line 2). Next, we refine  $can\_2\_Set$  to remove non frequent candidate without generating their pattern instances. In Line 4, we traverse  $PT$  and obtain a set of patterns satisfying condition 1 discussed in Proposition 42. Note that, we only consider those patterns satisfying  $min\_pi$  threshold while traversal. In Line 5, we generate pattern instances of pattern present in  $can\_2\_Set$  and  $ext\_Set$ . To derive pattern instances of given



**Table 2.**  $fre\_2\_Set$  and  $ext\_Set$ 

$fre\_2\_Set$	$ext\_Set$
$\langle \{f_4 \rightarrow f_5\}, \{B\}, \{NE\} \rangle$	$\langle \{f_4 \rightarrow f_5\}, \{B\}, \{NE\} \rangle$
$\langle \{f_3 \rightarrow f_4\}, \{B\}, \{NW\} \rangle$	$\langle \{f_3 \rightarrow f_4\}, \{B\}, \{NW\} \rangle$
	$\langle \{f_1 \rightarrow f_2 \rightarrow f_3\}, \{O, B, O\}, \{N, N, N\} \rangle$
	$\langle \{f_2 \rightarrow f_3\}, \{O\}, \{N\} \rangle$

pattern  $P$ , use similar process explained in section 3. For example, consider a pattern  $P = \langle \{f_1 \rightarrow f_2 \rightarrow f_3\}, \{O, B, O\}, \{N, N, N\} \rangle$ . We first derive pattern instances of  $\langle \{f_1 \rightarrow f_2\}, \{O\}, \{N\} \rangle$  and  $\langle \{f_2 \rightarrow f_3\}, \{O\}, \{N\} \rangle$  and then join them to derive it's pattern instances. Finally,  $fre\_2\_Set$  and  $ext\_Set$  is returned. Note that,  $ext\_Set$  does not contain  $P = \langle \{f_4 \rightarrow f_5\}, \{S\}, \{E\} \rangle$  and it's extended patterns as  $P$  is not frequent. We omit proof of completeness of IncIOMiner for brevity.

---

**Algorithm 3.** GetExtendExtensionSet( $D_{0,t}$ ,  $D_{t,t+\Delta t}$ ,  $PT$ ,  $min\_pi$ ,  $T_\delta$ ,  $R_\delta$ )

---

**Output:**  $fre\_2\_Set$ ,  $ext\_Set$

- 1 Obtain  $D'$  from  $D_{0,t}$
  - 2  $can\_2\_Set = \{\text{length 2 patterns using } \{D' \cup D_{t,t+\Delta t}\}\}$
  - 3  $can\_2\_Set = \text{Refine } can\_2\_Set \text{ using upperbound estimation (Eq. 2 and 3)}$
  - 4  $ext\_Set = \{P \mid (P \in PT) \wedge (|P| > 1) \wedge (P.Te \geq (t-T_\delta)) \wedge (UB.pi(P) \geq min\_pi)\}$
  - 5 Generate pattern instances of patterns from ( $ext\_Set$  and  $can\_2\_Set$ ) using  $D_{0,t+\Delta t}$
  - 6  $fre\_2\_Set = fre\_2\_Set \cup \{\text{frequent patterns from } can\_2\_Set\}$
  - 7  $fre\_ext\_Set = fre\_2\_Set \cup \{\text{frequent patterns from } ext\_Set\}$
  - 8 return  $\{fre\_2\_Set, fre\_ext\_Set\}$
- 

## 5 Experimental Results

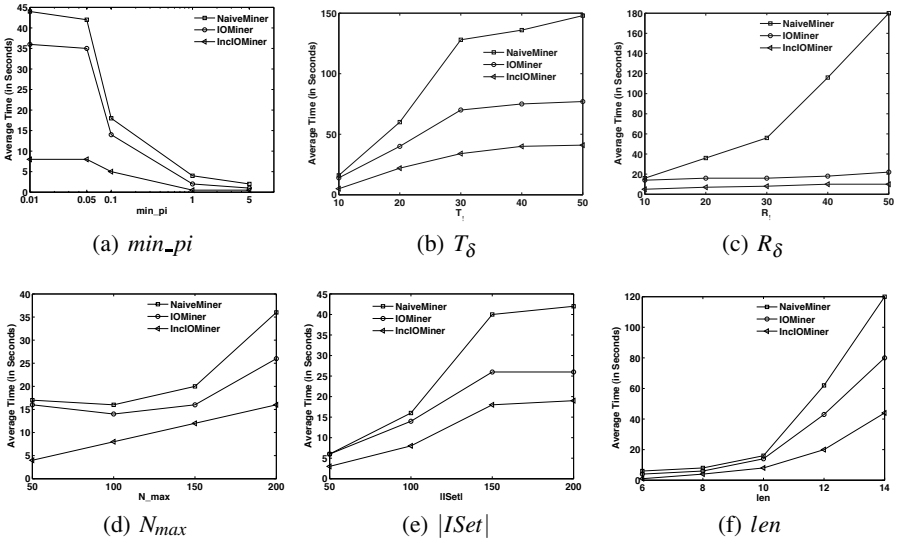
We conduct detailed performance study. All algorithms are implemented in C and compiled with -O2 option. We use a Pentium 4 machine with 3GB RAM running windows. No other user processes were running at that time. We also implement an algorithm called *NaiveMiner* that use only cube based hashing to obtain the longer length pattern. Actually, NaiveMiner is motivated from [19,12].

### 5.1 Efficiency Experiments

We first implement a synthetic data generator to obtain spatio-temporal features. Table 3 summarizes the experimental parameters used to generate the dataset. Essentially, we generate features so that the generated data has total  $N_{max}$  maximal patterns of average length  $len$  and each pattern has  $|ISet|$  number of average pattern instances. While generating features, we also use parameters  $R_\delta$ ,  $T_\delta$  and  $min\_pi$ . With using generator and user defined input parameters, we create an initial dataset  $D_0$  and it's four increments. The size of increment (i.e.,  $inc\_size$ ) is 10% of original data size.

**Table 3.** Experimental Parameter

Parameter	Definition	Default value
$N_{max}$	Number of maximal IO patterns	100
$len$	Average length of maximal IO patterns	10
$ ISet $	Average number of pattern instances for each IO patterns	100
$dur$	Average duration of features	10
$N$	Number of features type	100
$R_\delta$	Distance threshold for spatial proximity	10
$T_\delta$	Distance threshold for temporal proximity	10
$min\_pi$	Prevalence index threshold	0.1
$inc\_size$	Size of incremental database	10%
$map$	Latitude-Longitude extent of the map	1000 x 1000

**Fig. 6.** Effects of varying  $min\_pi$ ,  $T_\delta$  and  $R_\delta$ ,  $N_{Max}$ ,  $|ISet|$  and  $len$ 

In our first set of experiments, we use default value of parameters given in Table 3 to generate spatio-temporal features and vary user defined parameters especially,  $min\_pi$ ,  $R_\delta$  or  $T_\delta$ . Note that, the data generated using default parameters has around 70K number of instances in the original data and subsequent increment has around 5K, 7K, 8K and 7K instances. In our experiment, we capture runtime of NaiveMiner, IOMiner and IncIOMiner for mining each datasets (Original dataset and its 4 increment) and report the average running time [11]. We first vary  $min\_pi$  from 0.01% to 5%. Figure 6(a) reports the obtained result. As we reduce  $min\_pi$ , running time increased. However, IncIOMiner runs very fast as  $min\_pi$  reduce. Similarly, we vary  $T_\delta$  from 10 to 50 (See Figure 6(b)) and  $R_\delta$  from 10 to 50 (See Figure 6(c)). For both cases, run time of NaiveMiner and IOMiner increase rapidly. The reason is, increasing  $R_\delta$  or  $T_\delta$  generates many patterns.

In next set of experiments, we vary  $N_{max}$ ,  $len$  or  $|ISet|$ . Note that, increasing any of these parameters indirectly increases the size of database and also required to increase the number of features (i.e.,  $N$ ) while generating dataset. More specifically, these set of

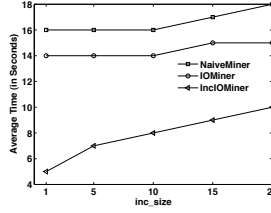


Fig. 7. Effects of varying size of increment

experiments also capture effects of increasing size of database and number of features on average running time. We vary  $N_{max}$  from 10 to 200 (size of original database linearly increases from 7K to 142K). Figure 6(d) present the results. Clearly, as size of data increase, the performance gap between IOMiner and IncIOMiner also increase. Similar apply when we vary  $|ISet|$  from 10 to 200(See Figure 6(e)). Similarly, we vary average length of maximal patterns from 6 to 14 and observed that algorithm IncIOMiner is efficient(See Figure 6(f)). Note that, we use default value of increment size(10%) for these set of experiments.

We vary  $inc\_size$  from 1% to 20% (See Figure 7). Note that, as we increase the size of increment, total database size also increase. It is obvious that keeping  $min\_pi$  constant and increasing total database size will generate less number of patters. We can observe that when size of increments is less, IncIOMiner is very efficient algorithm. Also, NaiveMiner is outperformed by IOMiner in all the experiments.

## 5.2 Effectiveness Experiments

We conduct qualitative experimental study on three real-world public datasets named drought, vegetation and human activity videos.

**Drought Dataset.** This dataset, obtained from National Climatic Data Center, includes monthly average of three variables named temperature(AvgTemp), precipitation (AvgPcp) and palmer drought severity index(PDSI). Climate divisions at various spatial locations record value of above variables. Current dataset has total 334 climatic divisions and we use 10 year data captured from 1999 to 2008. In short, we have real valued time series of each variable at each climate division. We use discretization process [2] to extract spatio-temporal features such as episode of high or low temperature(precipitation), episode of cold or wet drought at particular location. Finally, our working dataset has around 85K features. From such dataset, we discover key spatio-temporal relationships to link the drought condition to high/low events. We run IOMiner with  $R_\delta = 30$ ,  $T_\delta = 5$  and  $min\_pi = 5\%$ . It has been noted that high temperature and low precipitation creates many problems and one of the problem is period of dry spell(i.e., Low PDSI)(<http://www.sciencedaily.com/releases/2007/08/070820144517.htm>)(See Pattern P3 in Table 4).

**Vegetation Dataset.** This dataset, obtained from ISLSCP II Data Archive, includes seven variables named precipitation(Pcp), temperature(Temp), cloud cover(CloudCover), diurnal temperature range(DiurnalTemp), wet dry frequency(WDFre), vapor pressure

(VPressure) and normalized difference vegetation index(NDVI). This dataset has global mean monthly value for each variable from 1991-1995. We use discretization process [2] to extract spatio-temporal features where a feature represent an episode of high or low value for each attribute at certain location. Finally, our working dataset has around 45K features. From this dataset, we discover key spatio-temporal relationships that links high/low vegetation condition to high/low episode of any variables. We run IOMiner with  $R_\delta = 20$ ,  $T_\delta = 3$  months and  $min\_pi = 5\%$ .

**Human Activity Video Dataset.** We use two different datasets named KTH dataset (<http://www.nada.kth.se/cvap/actions/>) and ICPR 2010 Challenge Dataset. KTH dataset has 25 persons engaged in the following activities: running, walking, jogging, boxing, clapping and waving. We use subset of dataset that includes 10 persons where each person repeats each activity 4 times for about 4 seconds each, wearing different clothing. Thus we have total 240 video clips. First, we extract spatio temporal word, a kind of STIP descriptor, described in [4] from each video clip. Finally, our working dataset has around 60K features. From this dataset, we discover key spatio-temporal relationships that help to discriminate human activities. Patterns P8-P10 in Table 4 are the patterns discovered from KTH dataset.

To further validate the usefulness of IO patterns, we extend a bag of spatio-temporal words classification technique [4] for human action recognition. Our approach extends the bag of words dataset with discovered IO patterns as suggested in [1]. Next, we perform 10 fold cross validation using SVM classifier on initial bag of words dataset, denoted as Bag\_of\_Words, and extended bag of word dataset, denoted as Bag\_of\_Word+IO. Instead of IO patterns, we also consider other patterns such as temporal pattern(TP) and interval Pattern(IP). Temporal pattern does not consider the space dimension [12] and interval pattern is an IO pattern without the direction relationship. We observe that when we use IO pattern with bag of words representation, the classification accuracy is improved by **2.5%**. We also use ICPR 2010 challenge dataset. This dataset contains videos

**Table 4.** Subset of IO patterns obtained from real world dataset

<b>Drought Dataset</b>	
P1.	$\langle \{High\_PDSI \rightarrow High\_AvgPcp\}, \{C\}\{NO\} \rangle$
P2.	$\langle \{Normal\_AvgTemp \rightarrow High\_AvgPcp \rightarrow Normal\_PDSI\}, \{M,B,B\}, \{NW,NW,NW\} \rangle$
P3.	$\langle \{Low\_PDSI \rightarrow High\_AvgTemp \rightarrow Low\_AvgPcp\}\{C,C,B\}, \{N,N,N\} \rangle$
<b>Vegetation Dataset</b>	
P4.	$\langle \{Low\_CloudCover \rightarrow Low\_NDVI\}, \{M\}, \{NO\} \rangle$
P5.	$\langle \{High\_Temp \rightarrow Low\_NDVI\}, \{S\}, \{N\} \rangle$
P6.	$\langle \{High\_Pcp \rightarrow High\_CloudCover \rightarrow Low\_DiurnalTemp\}, \{S,S,E\}, \{N,N,N\} \rangle$
P7.	$\langle \{High\_Pcp \rightarrow High\_VPressure \rightarrow High\_NDVI\}, \{S,S,E\}\{N,N,NO\} \rangle$
<b>Human Activity Video Dataset</b>	
P8.	$\langle \{88 \rightarrow 304\}, \{O\}, \{N\} \rangle$ [6:Walking, 1:Running and 1:Jogging]
P9.	$\langle \{109 \rightarrow 259\}, \{C\}, \{N\} \rangle$ [1: Walking, 3:Running]
P10.	$\langle \{302 \rightarrow 322\}, \{C\}, \{S\} \rangle$ [1:Running, 3:Jogging]

**Table 5.** Comparing classification results of human activity detection

Datasets	Bag_of_Words	Bag_of_Words+TP	Bag_of_Words+SP	Bag_of_Words+IO
KHT Dataset	79.43	76.45	79.85	<b>81.93</b>
ICPR 2010 Challenge Dataset	39.00	40.00	46.53	<b>55.00</b>

of executions of 6 classes of human-human interactions: shake-hands, point, hug, push, kick and punch. We use video from set 1 (total 60 videos) and obtain spatio-temporal words for each execution (i.e., video). We repeat 10 fold cross validation using SVM classifier on initial bag of words dataset and extended dataset. Table 5 lists the results.

## 6 Related Work

Existing studies have proposed various spatio-temporal patterns from instantaneous features. A co-location pattern is a set of features whose instances are frequently located together in space [13]. A topological pattern [15] is extension of co-location pattern with additional consideration of temporal constraints. Table 6 summarize various ordered patterns proposed for spatio-temporal databases. Each pattern in Table 6 is a sequence of features with constraints in space and time dimensions. All these proposals consider non-transactional database.

**Table 6.** Spatio-Temporal Patterns

Patterns	Space Dimension		Time Dimension	
	Directional Relation	Spatial Proximity	Temporal Distance	Allen's Temporal Relationship
Generalized flow pattern [16,10]	√	All features in the sequence are <b>neighbor</b> of each other Adjacent features in the sequence are <b>neighbor</b> of each other	All features in the sequence are <b>neighbor</b> of each other	Before
Flow pattern [17]	√		Adjacent features in the sequence are <b>neighbor</b> of each other	
Sequential pattern [6]	×			Adjacent features in the sequence are <b>neighbor</b> of each other
Interval-Orientation pattern	√			
Interval patterns	×			

Dealing with non-transactional database is an issue. [16,17] discretize the space and obtain the transactions. Later, they extend the prefix based pattern growth algorithm for mining desired patterns. Adapting transactional approach to discover IO patterns requires every feature instance must be present in each transaction. Majority of papers that does not discretize the space uses the prevalence index [13,15,10] as an interesting measure. The work in [6] use non-antimonote density ratio as an interesting measure for sequential pattern. Further, the proposed algorithm is candidate set generation-and-test approach (i.e., NaiveMiner). Our work use prevalence index as an interesting measure and hash-based joining [11] to generate the IO patterns. The works in [14] deals with moving object databases.

Recently, many algorithms consider incremental mining of traditional transactional database, such as [11,3]. However, these algorithms are not applicable directly for incremental mining. Reason is our underlying database is non-transactional and we use prevalence index measure as an interesting measure for IO patterns. To the best of our knowledge, there is no existing method which can be adapted to discover IO patterns efficiently and incrementally from spatio-temporal databases.

## 7 Conclusion

In this paper, we discuss an algorithm for mining IO patterns from spatio-temporal databases. We extended the proposed algorithm to work in incremental fashion. The

incremental algorithm reuse already known knowledge to reduce the search space. Experiments on synthetic data indicates efficiency and scalability of the proposed algorithm. Experiments on real world data suggest that we mine useful knowledge. In future, we like to extend our framework such that it allow an interactive analysis.

## References

1. Cheng, H., Yan, X., Han, J., Yu, P.S.: Direct discriminative pattern mining for effective classification. In: ICDE, pp. 169–178 (2008)
2. Ding, W., Tomasz, S., Josue, S.: Discovery of geospatial discriminating patterns from remote sensing dataset. In: SDM (2008)
3. Ding, Y., Lee, K., Cheng, H., Krishna, G., Li, Z., Ma, X., Zhou, Y., Han, J.: Cispan: Comprehensive incremental mining algorithms of closed sequential patterns for multi-versioned software mining. In: SDM, pp. 84–95. SIAM, Philadelphia (2008)
4. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavioural recognition via sparse spatio-temporal features. In: VS-PETS, pp. 65–72 (2005)
5. Huang, Y., Zhang, L., Zhang, P.: Can we apply projection based frequent pattern mining paradigm to spatial co-location mining. In: AKDDM. LNCS, pp. 433–448 (2005)
6. Huang, Y., Zhang, L., Zhang, P.: A framework for mining sequential patterns from spatio-temporal event data set. In: TKDE, pp. 433–448 (2008)
7. Kuldorff, M., Athas, W., Feuer, E., Miller, B., Key, C.: Evaluating cluster alarms: A space-time scan statistic and brain cancer in los alamos. In: AJPH, pp. 1377–1380 (1998)
8. Laptev, I.: On space-time interest points. In: IJCV, pp. 107–123 (2005)
9. Mohammadi, S., Janeja, V., Gangopadhyay, A.: Discretized spatio-temporal scan window. In: SIAM (2009)
10. Mohan, P., Shekhar, S., Shine, J., Rogers, J.: Cascading spatio-temporal pattern discovery: A summary of results. In: SDM (2010)
11. Parthasarathy, S., Zaki, M.J., Ogihara, M., Dwarkadas, S.: Incremental and interactive sequence mining. In: KDD (1999)
12. Patel, D., Hsu, W., Lee, M.L.: Mining relationships among interval-based events for classification. In: SIGMOD (2008)
13. Shekhar, S., Huang, Y.: Discovering spatial co-location patterns: A summary of results. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 236–256. Springer, Heidelberg (2001)
14. Verhein, F.: Mining complex spatio-temporal sequence patterns. In: SDM (2009)
15. Wang, J., Hsu, W., Lee, M.L.: A framework for mining topological patterns in spatio-temporal databases. In: CIKM, pp. 429–436. ACM, New York (2005)
16. Wang, J., Hsu, W., Lee, M.L.: Mining generalized spatio-temporal patterns. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 649–661. Springer, Heidelberg (2005)
17. Wang, J., Hsu, W., Lee, M.L., Wang, J.: Flowminer: Finding flow patterns in spatio-temporal databases. In: ICTAI, pp. 14–21. IEEE, Los Alamitos (2004)
18. Wie, L., Shan, M.: Mining temporal co-orientation pattern from spatio-temporal databases. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 895–903. Springer, Heidelberg (2007)
19. Yang, H., Parthasarathy, S., Mehta, S.: A generalized framework for mining spatio-temporal patterns in scientific data. In: KDD, pp. 716–721. ACM, New York (2005)

# Efficient K-Nearest Neighbor Search in Time-Dependent Spatial Networks\*

Ugur Demiryurek, Farnoush Banaei-Kashani, and Cyrus Shahabi

University of Southern California  
Department of Computer Science  
Los Angeles, CA 90089-0781  
{demiryur, banaeika, shahabi}@usc.edu

**Abstract.** The class of  $k$  Nearest Neighbor ( $k$ NN) queries in spatial networks has been widely studied in the literature. All existing approaches for  $k$ NN search in spatial networks assume that the weight (e.g., travel-time) of each edge in the spatial network is constant. However, in real-world, edge-weights are time-dependent and vary significantly in short durations, hence invalidating the existing solutions. In this paper, we study the problem of  $k$ NN search in time-dependent spatial networks where the weight of each edge is a function of time. We propose two novel indexing schemes, namely Tight Network Index (*TNI*) and Loose Network Index (*LNI*) to minimize the number of candidate nearest neighbor objects and, hence, reduce the invocation of the expensive fastest-path computation in time-dependent spatial networks. We demonstrate the efficiency of our proposed solution via experimental evaluations with real-world data-sets, including a variety of large spatial networks with real traffic-data.

## 1 Introduction

Recent advances in online map services and their wide deployment in hand-held devices and car-navigation systems have led to extensive use of location-based services. The most popular class of such services is  $k$ -nearest neighbor ( $k$ NN) queries where users search for geographical points of interests (e.g., restaurants, hospitals) and the corresponding directions and travel-times to these locations. Accordingly, numerous algorithms have been developed (e.g., [20,15,19,21,13,16,22]) to efficiently compute the distance and route between objects in large road networks.

The majority of these studies and existing commercial services makes the simplifying assumption that the cost of traveling each edge of the road network is constant (e.g., corresponding to the length of the edge) and rely on pre-computation of distances in the network. However, the actual travel-time on road networks heavily depends on the traffic congestion on the edges and hence is a function of the time of the day, i.e.,

---

\* This research has been funded in part by NSF grant CNS-0831505 (CyberTrust) and in part from METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We thank Professor David Kempe for helpful discussions.

travel-time is *time-dependent*. For example, Figure 1 shows the real-world travel-time pattern on a segment of I-10 freeway in Los Angeles between 6AM and 8PM on a weekday. Two main observations can be made from this figure. First, the arrival-time to the segment entry determines the travel-time on that segment. Second, the change in travel-time is significant and continuous (not abrupt), for example from 8:30AM to 9:00AM, the travel-time of this segment changes from 30 minutes to 18 minutes (40% decrease). These observations have major computation implications: the fastest path from a source to a destination may vary significantly depending on the departure-time from the source, and hence, the result of spatial queries (including  $k$ NN) on such dynamic network heavily depends on the time at which the query is issued.

Figure 2 shows an example of time-dependent  $k$ NN search where an ambulance is looking for the nearest hospital (with least travel-time) at 8:30AM and 2PM on the same day on a particular road network. The time-dependent travel-time (in minutes) and the arrival time for each edge are shown on the edges. Note that the travel-times on an edge changes depending on the arrival time to the edge in Figures 2(a) and 2(b). Hence, the query issued by the ambulance at 8:30AM and 2PM would return different results.

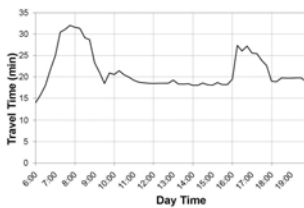
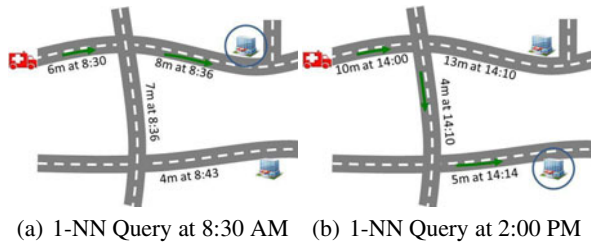


Fig. 1. Real-world travel-time



(a) 1-NN Query at 8:30 AM (b) 1-NN Query at 2:00 PM

Fig. 2. Time-dependent 1-NN search

Meanwhile, an increasing number of navigation companies have started releasing their time-dependent travel-time information for road networks. For example, Navteq [17] and TeleAtlas [21], the leading providers of navigation services, offer traffic flow services that provide time-dependent travel-time (at the temporal granularity of as low as five minutes) of road network edges up to one year. The time-dependent travel-times are usually extracted from the historical traffic data and local information like weather, school schedules, and events. Based on Navteq's analysis, the time-dependent weight information improves the travel-time accuracy by an average of 41% when compared with typical speeds (time-independent) on freeways and surface streets. Considering the availability of time-dependent travel-time information for road networks on the one hand and the importance of time-dependency for accurate and realistic route planning on the other hand, it is essential to extend existing literature on spatial query processing and planning (such as  $k$ NN queries) in road networks to a new family of time-dependent query processing solutions.

Unfortunately, once we consider time-dependent edge weights in road networks, all the proposed  $k$ NN solutions assuming constant edge-weights and/or relying on distance precomputation would fail. However, one can think of several new baseline solutions.



Firstly, Dreyfus [7] has studied the relevant problem of time-dependent shortest path planning and showed that this problem can be solved by a trivially-modified variant of any label-setting (e.g., Dijkstra) static shortest path algorithm. Consequently, we can develop a primitive solution for the time-dependent  $k$ NN problem based on the incremental network expansion (INE [19]) approach where Dreyfus's modified Dijkstra algorithm is used for time-dependent distance calculation. With this approach, starting from a query object  $q$  all network nodes reachable from  $q$  are visited in order of their time-dependent travel-time proximity to  $q$  until all  $k$  nearest objects are located (i.e., blind network expansion). However, considering the prohibitively high overhead of executing blind network expansion particularly in large networks with a sparse (but perhaps large) set of data objects, this approach is far too slow to scale for real-time  $k$ NN query processing. Secondly, we can use time-expanded graphs [9] to model the time-dependent networks. With time-expanded graphs the time domain is discretized and at each discrete time instant a snapshot of the network is used to represent the network. With this model, the time-dependent  $k$ NN problem is reduced to the problem of computing the minimum-weight paths through a series of static networks. Although this approach allows for exploiting the existing algorithms for  $k$ NN computation on static networks, it often fails to provide the correct results because the model misses the state of the network between any two discrete time instants. Finally, with a third approach we can precompute time-dependent shortest paths between all possible sources and destinations in the network. However, shortest path precomputation on time-dependent road networks is challenging. Because, the shortest path on time-dependent networks (i.e., a network where edge weights are function of time) depends on the departure time from the source, and therefore, one needs to precompute all possible shortest paths for *all possible departure-times*. Obviously, this is not a viable solution because the storage requirements for the precomputed paths would quickly exceed reasonable space limitations. With our prior work [4], for the first time we introduced the problem of *Time-Dependent  $k$  Nearest Neighbor* (TD- $k$ NN) search to find the  $k$ NN of a query object that is moving on a *time-dependent network*. With this work, we also investigated the first two baseline approaches discussed above (the third approach is obviously inapplicable) by extensive experiments to rigorously characterize the inefficiency and inaccuracy of the two baseline solutions, respectively.

In this paper, we address the disadvantages of both baseline approaches by developing a novel technique that efficiently and accurately finds  $k$ NN of a query object in time-dependent road networks. A comprehensive solution for TD- $k$ NN query should a) efficiently answer the queries in (near) real-time in order to support moving object  $k$ NN search on road networks, b) be independent of density and distribution of the data objects, and c) effectively handle the database updates where nodes, links, and data objects are added or removed. We address these challenges by developing two types of complementary index structures. The main idea behind these index structures is to localize the search space and minimize the costly time-dependent shortest path computation between the objects hence incurring low computation costs. With our first index termed *Tight Network Index (TNI)*, we can find the nearest objects without performing any shortest path computation. Our experiments show that in 70% of the cases the nearest neighbor can be found with this index. For those cases that the nearest objects cannot be

identified by TNI, our second index termed *Loose Network Index (LNI)* allows us to filter in only a small number of objects that are potential candidates (and filter out the rest of the objects). Subsequently, we only need to perform the shortest path computation only for these candidates. Our TD- $k$ NN algorithm consists of two phases. During the first phase (off-line), we partition the spatial network into subnetworks (cells) around the data objects by creating two cells for each data object called *Tight Cell (TC)* and *Loose Cell (LC)* and generate TNI and LNI on these cells, respectively. In the second phase (online), we use TNI and LNI structures to immediately find the first nearest neighbor and then expand the search area to find the remaining  $k-1$  neighbors.

The remainder of this paper is organized as follows. In Section 2 we review the related work on both  $k$ NN and time-dependent shortest path studies. In Section 3 we formally define the TD- $k$ NN query in spatial networks. In Section 4 we establish the theoretical foundation of our algorithms and explain our query processing technique. In Section 5 we present experimental results on variety of networks with actual time-dependent travel-times generated from real-world traffic data (collected for past 1.5 years). In Section 6 we conclude and discuss our future work.

## 2 Related Work

In this section we review previous studies on  $k$ NN query processing in road networks as well as time-dependent shortest path computation.

### 2.1 $k$ NN Queries in Spatial Networks

In [19], Papadias et al. introduced Incremental Network Expansion (INE) and Incremental Euclidean Restriction (IER) methods to support  $k$ NN queries in spatial networks. While INE is an adaption of the Dijkstra algorithm, IER exploits the Euclidean restriction principle in which the results are first computed in Euclidean space and then refined by using the network distance. In [15], Kolahdouzan and Shahabi proposed first degree *network Voronoi diagrams* to partition the spatial network to network Voronoi polygons (NVP), one for each data object. They indexed the NVPs with a spatial access method to reduce the problem to a point location problem in Euclidean space. Cho et al. [2] presented a system UNICONS where the main idea is to integrate the precomputed  $k$ NNs into the Dijkstra algorithm. Hu et al. [12] proposed a distance signature approach that precomputes the network distance between each data object and network vertex. The distance signatures are used to find a set of candidate results and Dijkstra is employed to compute their exact network distance. Huang et al. addressed the  $k$ NN problem using *Island* approach [13] where each vertex is associated to all the data points that are in radius  $r$  (so called islands) covering the vertex. With their approach, they utilized a restricted network expansion from the query point while using the precomputed islands. Recently Samet et al. [20] proposed a method where they associate a label to each edge that represents all nodes to which a shortest path starts with this particular edge. The labels are used to traverse *shortest path quadrees* that enables geometric pruning to find the network distance. With all these studies, the edge weight functions are assumed to be constant and hence the shortest path computations

and precomputations are no longer valid with time-varying edge weights. Unlike the previous approaches, we make a fundamentally different assumption that the weight of the network edges are time-dependent rather than fixed.

## 2.2 Time-Dependent Shortest Path Studies

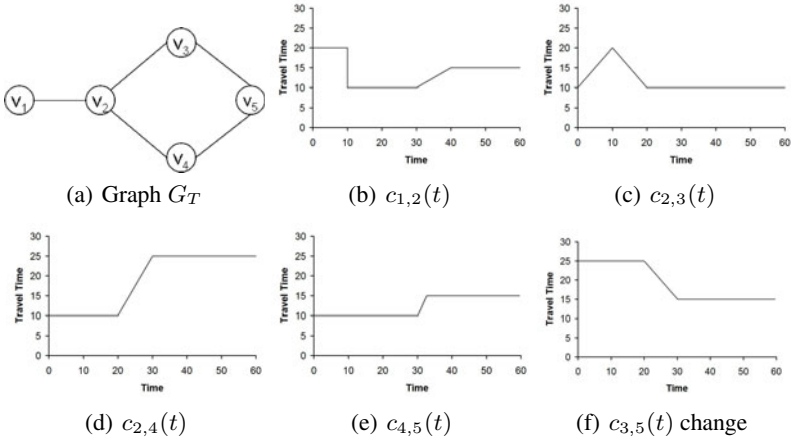
Cooke and Halsey [3] introduced the first time-dependent shortest path (TDSP) solution where dynamic programming is used over a discretized network. In [1], Chabini proposed a discrete time TDSP algorithm that allows waiting at network nodes. In [9], George and Shekhar proposed a time-aggregated graph where they aggregate the travel-times of each edge over the time instants into a time series. All these studies assume the edge weight functions are defined over a finite discrete sequence of time steps  $t \in t_0, t_1, \dots, t_n$ . However, discrete-time algorithms have numerous shortcomings. First, since the entire network is replicated for every specified time step, the discrete-time methods require an extensive amount of storage space for real-world scenarios where the spatial network is large. Second, these approaches can only provide approximate results since the computations are done on discrete-times rather than in continuous time. In [7], Dreyfus proposed a generalization of Dijkstra algorithm, but his algorithm is showed (by Halpren [11]) to be true only in FIFO networks. If the FIFO property does not hold in a time-dependent network, then the problem is NP-Hard as shown in [18]. Orda and Rom [18] proposed a Bellman-Ford based solution where edge weights are piece-wise linear functions. In [6], Ding et al. used a variation of label-setting algorithm which decouples the path-selection and time-refinement by scanning a sequence of time steps of which the size depends on the values of the arrival time functions. In [14], Kanoulas et al. introduced allFP algorithm in which they, instead of sorting the priority queue by scalar values, maintain a priority queue of all the paths to be expanded. Therefore, they enumerate all the paths from a source to a destination which yields exponential run-time in the worst case.

## 3 Problem Definition

In this section, we formally define the problem of time-dependent  $k$ NN search in spatial networks. We assume a road network containing a set of data objects (i.e., points of interest such as restaurants, hospitals) as well as query objects searching for their  $k$ NN. We model the road network as a *time-dependent weighted graph* where the non-negative weights are time-dependent travel-times (i.e., positive piece-wise linear functions of time) between the nodes. We assume both data and query objects lie on the network edges and all relevant information about the objects is maintained by a central server.

**Definition 1.** A *Time-dependent Graph* ( $G_T$ ) is defined as  $G_T(V, E)$  where  $V$  and  $E$  represent set of nodes and edges, respectively. For every edge  $e(v_i, v_j)$ , there is a cost function  $c_{(v_i, v_j)}(t)$  which specifies the cost of traveling from  $v_i$  to  $v_j$  at time  $t$ .  $\square$

Figure 3 depicts a road network modeled as a time-dependent graph  $G_T(V, E)$ . While Figure 3(a) shows the graph structure, Figures 3(b), 3(c), 3(d), 3(e), and 3(f) illustrate the time-dependent edge costs as piece-wise linear functions for the corresponding



**Fig. 3.** A Time-dependent Graph  $G_T(V, E)$

edges. For each edge, we define upper-bound ( $max(c_{v_i, v_j})$ ) and lower-bound ( $min(c_{v_i, v_j})$ ) time-independent costs. For example, in Figure 3(b),  $min(c_{v_1, v_2})$  and  $max(c_{v_1, v_2})$  of edge  $e(v_1, v_2)$  are 10 and 20, respectively.

**Definition 2.** Let  $\{s = v_1, v_2, \dots, v_k = d\}$  represent a path which contains a sequence of nodes where  $e(v_i, v_{i+1}) \in E$  and  $i = 1, \dots, k - 1$ . Given a  $G_T$ , a path  $(s \rightsquigarrow d)$  from source  $s$  to destination  $d$ , and a departure-time at the source  $t_s$ , the time-dependent travel time  $TT(s \rightsquigarrow d, t_s)$  is the time it takes to travel along the path. Since the travel-time of an edge varies depending on the arrival-time to that edge (i.e., arrival dependency), the travel time is computed as follows:

$$TT(s \rightsquigarrow d, t_s) = \sum_{i=1}^{k-1} c_{(v_i, v_{i+1})}(t_i) \text{ where } t_1 = t_s, t_{i+1} = t_i + c_{(v_i, v_{i+1})}(t_i), i = 1, \dots, k.$$

The upper-bound travel-time  $UTT(s \rightsquigarrow d)$  and the lower-bound travel time  $LTT(s \rightsquigarrow d)$  are defined as the maximum and minimum possible times to travel along the path, respectively. The upper and lower bound travel time are computed as follows,

$$UTT(s \rightsquigarrow d) = \sum_{i=1}^{k-1} max(c_{v_i, v_{i+1}}), LTT(s \rightsquigarrow d) = \sum_{i=1}^{k-1} min(c_{v_i, v_{i+1}}), i = 1, \dots, k.$$

To illustrate the above definitions in Figure 3, consider  $t_s = 5$  and path  $(v_1, v_2, v_3, v_5)$  where  $TT(v_1 \rightsquigarrow v_5, 5) = 45$ ,  $UTT(v_1 \rightsquigarrow v_5) = 65$ , and  $LTT(v_1 \rightsquigarrow v_5) = 35$ .

Note that we do not need to consider arrival-dependency when computing  $UTT$  and  $LTT$  hence;  $t$  is not included in their definitions. Given the definitions of  $TT$ ,  $UTT$  and  $LTT$ , the following property holds for any path in  $G_T$ :  $LTT(s \rightsquigarrow d) \leq TT(s \rightsquigarrow d, t_s) \leq UTT(s \rightsquigarrow d)$ . We will use this property in subsequent sections to establish some properties of our algorithm.

**Definition 3.** Given a  $G_T$ ,  $s$ ,  $d$ , and  $t_s$ , the time-dependent shortest path  $TDSP(s, d, t_s)$  is a path with the minimum travel-time among all paths from  $s$  to  $d$ . Since we consider the travel-time between nodes as the distance measure, we refer to  $TDSP(s, d, t_s)$  as

time-dependent fastest path  $TDFP(s, d, t_s)$  and use them interchangeably in the rest of the paper.  $\square$

In a  $G_T$ , the fastest path from  $s$  to  $d$  is based on the departure-time from  $s$ . For instance, in Figure 3, suppose a query looking for the fastest path from  $v_1$  to  $v_5$  at  $t_s = 5$ . Then,  $TDFP(v_1, v_5, 5) = \{v_1, v_2, v_3, v_5\}$ . However, the same query at  $t_s = 10$  returns  $TDFP(v_1, v_5, 10) = \{v_1, v_2, v_4, v_5\}$ . Obviously, with constant edge weights (i.e., time-independent), the query would always return the same path as a result.

**Definition 4.** A time-dependent  $k$  nearest neighbor query (TD- $k$ NN) is defined as a query that finds the  $k$  nearest neighbors of a query object which is moving on a time-dependent network  $G_T$ . Considering a set of  $n$  data objects  $P = \{p_1, p_2, \dots, p_n\}$ , the TD- $k$ NN query with respect to a query point  $q$  finds a subset  $P' \subseteq P$  of  $k$  objects with minimum time-dependent travel-time to  $q$ , i.e., for any object  $p' \in P'$  and  $p \in P - P'$ ,  $TDFP(q, p', t) \leq TDFP(q, p, t)$ .  $\square$

In the rest of this paper, we assume that  $G_T$  satisfies the First-In-First-Out (FIFO) property. This property suggests that moving objects exit from an edge in the same order they entered the edge. In practice many networks, particularly transportation networks, exhibit FIFO property. We also assume that objects do not wait at a node, because, in most real-world applications, waiting at a node is not realistic as it requires the moving object to exit from the route and find a place to park and wait.

## 4 TD-KNN

In this section, we explain our proposed TD- $k$ NN algorithm. TD- $k$ NN involves two phases: an off-line spatial network indexing phase and an on-line query processing phase. During the off-line phase, the spatial network is partitioned into *Tight Cells (TC)* and *Loose Cells (LC)* for each data object  $p$  and two complementary indexing schemes *Tight Network Index (TNI)* and *Loose Network Index (LNI)* are constructed. The main idea behind partitioning the network to *TCs* and *LCs* is to localize the  $k$ NN search and minimize the costly time-dependent shortest path computation. These index structures enable us to efficiently find the data object (i.e., generator of a tight or loose cell) that is in shortest time-dependent distance to the query object  $q$ . During the on-line phase, TD- $k$ NN finds the first nearest neighbor of  $q$  by utilizing the *TNI* and *LNI* constructed in the off-line phase. Once the first nearest neighbor is found, TD- $k$ NN expands the search area by including the neighbors of the nearest neighbor to find the remaining  $k-1$  data objects. In the following sections, we first introduce our proposed index structures and then describe online query processing algorithm that utilizes these index structures.

### 4.1 Indexing Time-Dependent Network (Off-Line)

In this section, we explain the main idea behind tight and loose cells as well as the construction of tight and loose network index structures.

**Tight Network Index (TNI).** The tight cell  $TC(p_i)$  is a sub-network around  $p_i$  in which any query object is guaranteed to have  $p_i$  as its nearest neighbor in a time-dependent network. We compute tight cell of a data object by using parallel Dijkstra algorithm that grows shortest path trees from each data object. Specifically, we expand from  $p_i$  (i.e., the generator of the tight cell) assuming maximum travel-time between the nodes of the network (i.e., UTT), while in parallel we expand from each and every other data object assuming minimum travel-time between the nodes (i.e., LTT). We stop the expansions when the shortest path trees meet. The main rationale is that if the upper bound travel-time between a query object  $q$  and a particular data object  $p_i$  is less than the lower bound travel-times from  $q$  to any other data object, then obviously  $p_i$  is the nearest neighbor of  $q$  in a time-dependent network. We repeat the same process for each data object to compute its tight cell. Figure 4 depicts the network expansion from the data objects during the tight cell construction for  $p_1$ . For the sake of clarity, we represent the tight cell of each data object with a polygon as shown in Figure 5. We generate the edges of the polygons by connecting the adjacent border nodes (i.e., nodes where the shortest path trees meet) of a generator to each other. Lemma 1 proves the property of  $TC$ :

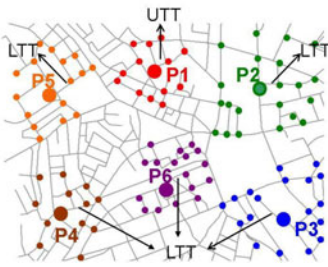


Fig. 4. Tight cell construction for  $P_1$

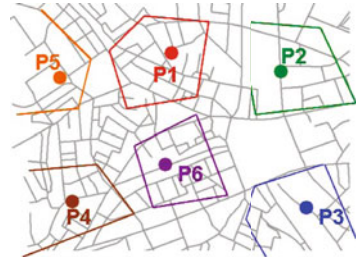


Fig. 5. Tight Cells

**Lemma 1.** Let  $P$  be a set of data objects  $P = \{p_1, p_2, \dots, p_n\}$  in  $G_T$  and  $TC(p_i)$  be the tight cell of a data object  $p_i$ . For any query point  $q \in TC(p_i)$ , the nearest neighbor of  $q$  is  $p_i$ , i.e.,  $\{\forall q \in TC(p_i), \forall p_j \in P, p_j \neq p_i, TDFP(q, p_i, t) < TDFP(q, p_j, t)\}$ .

*Proof.* We prove the lemma by contradiction. Assume that  $p_i$  is not the nearest neighbor of the query object  $q$ . Then there exists a data object  $p_j$  ( $p_i \neq p_j$ ) which is closer to  $q$ ; i.e.,  $TDFP(q, p_j, t) < TDFP(q, p_i, t)$ . Let us now consider a point  $b$  (where the shortest path trees of  $p_i$  and  $p_j$  meet) on the boundary of the tight cell  $TC(p_i)$ . We denote *shortest upper-bound path* from  $p_i$  to  $b$  (i.e., the shortest path among all  $UTT(p_i \rightsquigarrow b)$  paths) as  $D_{UTT}(p_i, b)$ , and similarly, we denote *shortest lower-bound path* from  $p_j$  to  $b$  (i.e., the shortest path among all  $LTT(p_j \rightsquigarrow b)$  paths) as  $D_{LTT}(p_j, b)$ . Then, we have  $TDFP(q, p_i, t) < D_{UTT}(p_i, b) = D_{LTT}(p_j, b) < TDFP(q, p_j, t)$ . This is a contradiction; hence,  $TDFP(q, p_i, t) < TDFP(q, p_j, t)$ .  $\square$

As we describe in Section 4.2, if a query point  $q$  is inside a specific  $TC$ , one can immediately identify the generator of that  $TC$  as the nearest neighbor for  $q$ . This stage can be expedited by using a spatial index structure generated on the  $TC$ s. Although  $TC$ s

are constructed based on the network distance metric, each  $TC$  is actually a polygon in Euclidean space. Therefore,  $TC$ s can be indexed using spatial index structures (e.g., R-tree [10]). This way a function (i.e.,  $contain(q)$ ) invoked on the spatial index structure would efficiently return the  $TC$  whose generator has the minimum time-dependent network distance to  $q$ . We formally define Tight Network Index as follows.

**Definition 5.** Let  $P$  be the set of data objects  $P = \{p_1, p_2, \dots, p_n\}$ , the Tight Network Index is a spatial index structure generated on  $\{TC(p_1), TC(p_2), \dots, TC(p_n)\}$ .  $\square$

As illustrated in Figure 5 the set of tight cells often does not cover the entire network. For the cases where  $q$  is located in an area which is not covered by any tight cell, we utilize the Loose Network Index ( $LNI$ ) to identify the candidate nearest data objects. Next, we describe  $LNI$ .

**Loose Network Index (LNI).** The loose cell  $LC(p_i)$  is a sub-network around  $p_i$  outside which any point is guaranteed *not* to have  $p_i$  as its nearest neighbor. In other words, data object  $p_i$  is guaranteed *not* to be the nearest neighbor of  $q$  if  $q$  is outside of the loose cell of  $p_i$ . Similar to the construction process for  $TC(p_i)$ , we use the parallel shortest path tree expansion to construct  $LC(p_i)$ . However, this time, we use minimum travel-time between the nodes of the network (i.e.,  $LTT$ ) to expand from  $p_i$  (i.e., the generator of the loose cell) and maximum travel-time (i.e.,  $UTT$ ) to expand from every other data object. Lemma2 proves the property of  $LC$ :

**Lemma 2.** Let  $P$  be a set of data objects  $P = \{p_1, p_2, \dots, p_n\}$  in  $G_T$  and  $LC(p_i)$  be the loose cell of a data object  $p_i$ . If  $q$  is outside of  $LC(p_i)$ ,  $p_i$  is guaranteed not to be the nearest neighbor of  $q$ , i.e.,  $\{\forall q \notin LC(p_i), \exists p_j \in P, p_j \neq p_i, TDFP(q, p_i, t) > TDFP(q, p_j, t)\}$ .

*Proof.* We prove by contradiction. Assume that  $p_i$  is the nearest neighbor of a  $q$ , even though the  $q$  is outside of  $LC(p_i)$ ; i.e.,  $TDFP(q, p_i, t) < TDFP(q, p_j, t)$ . Suppose there exists a data object  $p_j$  whose loose cell  $LC(p_j)$  covers  $q$  (such a data object must exist, because as we will next prove by Lemma 3, the set of loose cells cover the entire network). Let  $b$  be a point on the boundary of  $LC(p_i)$ . Then, we have,  $TDFP(q, p_j, t) < D_{UTT}(p_j, b) = D_{LTT}(p_i, b) < TDFP(q, p_i, t)$ . This is a contradiction; hence,  $p_i$  cannot be the nearest neighbor of  $q$ .  $\square$

As illustrated in Figure 6, loose cells, unlike  $TC$ s, collectively cover the entire network and have some overlapping regions among each other.

**Lemma 3.** Loose cells may overlap, and they collectively cover the network.

*Proof.* As we mentioned, during loose cell construction,  $LTT$  is used for expansion from the generator of the loose cell. Since the parallel Dijkstra algorithm traverses every node until the priority queue is empty as described in [8], every node in the network is visited; hence, the network is covered. Since the process of expansion with  $LTT$  is repeated for each data object, in the overall process some nodes are visited more than once; hence, the overlapping areas. Therefore, loose cells cover the entire network and may have overlapping areas. Note that if the edge weights are constant, the LCs would not overlap, and TCs cells and LCs would be the same.

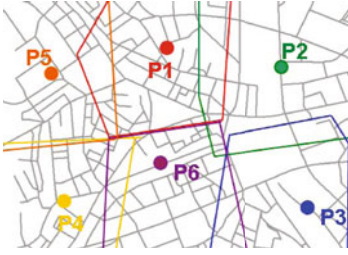


Fig. 6. Loose Cells

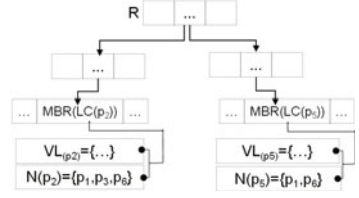


Fig. 7. LN R-tree

Based on the properties of tight and loose cells, we know that loose cells and tight cells have common edges (i.e., all the tight cell edges are also the edges of loose cells). We refer to data objects that share common edges as *direct neighbors* and remark that loose cells of the direct neighbors always overlap. For example, consider Figure 6 where the direct neighbors of  $p_2$  are  $p_1$ ,  $p_3$ , and  $p_6$ . This property is especially useful for processing  $k-1$  neighbors (see Section 4.2) after finding the first nearest neighbor. We determine the direct neighbors during the generation of the loose cells and store the neighborhood information in a data component. Therefore, finding the neighboring cells does not require any complex operation.

Similar to *TNI*, we can use spatial index structures to access loose cells efficiently. We formally define the Loose Network Index (*LNI*) as follows.

**Definition 6.** Let  $P$  be the set of data objects  $P = \{p_1, p_2, \dots, p_n\}$ , the Loose Network Index is a spatial index structure generated on  $\{LC(p_1), LC(p_2), \dots, LC(p_n)\}$ . □

Note that *LNI* and *TNI* are complementary index structures. Specifically, if a  $q$  cannot be located with *TNI* (i.e.,  $q$  falls outside of any *TC*), then we use *LNI* to identify the *LCs* that contain  $q$ ; based on Lemma 2, the generators of such *LCs* are the only NN candidates for  $q$ .

**Data Structures and Updates.** With our approach, we use R-Tree [10] like data structure to implement *TNI* and *LNI*, termed *TN R-tree* and *LN R-tree*, respectively. Figure 7 depicts *LN R-tree* (*TN R-tree* is a similar data structure without extra pointers at the leaf nodes, hence not discussed). As shown, *LN R-tree* has the basic structure of an R-tree generated on minimum bounding rectangles of loose cells. The difference is that we modify R-tree by linking its leaf nodes to the the pointers of additional components that facilitate TD-kNN query processing. These components are the direct neighbors ( $N(p_i)$ ) of  $p_i$  and the list of nodes ( $VL_{p_i}$ ) that are inside  $LC(p_i)$ . While  $N(p_i)$  is used to filter the set of candidate nearest neighbors where  $k > 1$ , we use  $VL_{p_i}$  to prune the search space during TDSP computation (see Section 4.2).

Our proposed index structures need to be updated when the set of data objects and/or the travel-time profiles change. Fortunately, due to local precomputation nature of TD-kNN, the affect of the updates with both cases are *local*, hence requiring minimal change in tight and loose cell index structures. Below, we explain each update type.

*Data Object Updates:* We consider two types of object update; insertion and deletion (object relocation is performed by a deletion following by insertion at the new location).



With a location update of a data object  $p_i$ , only the tight and loose cells of  $p_i$ 's neighbors are updated locally. In particular, when a new  $p_i$  is inserted, first we find the loose cell(s)  $LC(p_j)$  containing  $p_i$ . Clearly, we need to shrink  $LC(p_j)$  and since the loose cells and tight cells share common edges, the region that contains  $LC(p_j)$  and  $LC(p_j)$ 's direct neighbors needs to be adjusted. Towards that end, we find the neighbors of  $LC(p_j)$ ; the tight and loose cells of these direct neighbors are the only ones affected by the insertion. Finally, we compute the new TCs and LCs for  $p_i, p_j$  and  $p_j$ 's direct neighbors by updating our index structures. Deletion of a  $p_i$  is similar and hence not discussed.

*Edge Travel-time Updates:* With travel-time updates, we do not need to update our index structures. This is because the tight and loose cells are generated based on the minimum (LTT) and maximum (UTT) travel-times of the edges in the network that are time-independent. The only case we need to update our index structures is when minimum and/or maximum travel-time of an edge changes, which is not that frequent. Moreover, similar to the data object updates, the affect of the travel-time profile update is local. When the maximum and/or minimum travel-time of an edge  $e_i$  changes in the network, we first find the loose cell(s)  $LC(p_j)$  that overlaps with  $e_i$  and thereafter recompute the tight and loose cells of  $LC(p_j)$  and its direct neighbors.

## 4.2 TD- $k$ NN Query Processing (Online)

So far, we have defined the properties of  $TNI$  and  $LNI$ . We now explain how we use these index structures to process  $k$ NN queries in  $G_T$ . Below, we first describe our algorithm to find the nearest neighbor (i.e.,  $k=1$ ), and then we extend it to address the  $k$ NN case (i.e.,  $k \geq 1$ ).

**Nearest Neighbor Query.** We use  $TNI$  or  $LNI$  to identify the nearest neighbor of a query object  $q$ . Given the location of  $q$ , first we carry out a depth-first search from the  $TNI$  root to the node that contains  $q$  (Line 5 of Algorithm 1). If a tight cell that contains  $q$  is located, we return the generator of that tight cell as the first NN. Our experiments show that, in most cases (7 out of 10), we can find  $q$  with  $TNI$  search (see Section 5.2). If we cannot locate  $q$  in  $TNI$  (i.e., when  $q$  falls outside all tight cells), we proceed to search  $LNI$  (Line 7). At this step, we may find one or more loose cells that contain  $q$ . Based on Lemma 2, the generators of these loose cells are the only possible candidates to be the NN for  $q$ . Therefore, we compute TDFP to find the distance between  $q$  and each candidate in order to determine the first NN (Line 8-12). We store the candidates in a minimum heap based on their travel-time to  $q$  (Line 10) and retrieve the nearest neighbor from the heap in Line 12.

**$k$ NN Query.** Our proposed algorithm for finding the remaining  $k-1$  NNs is based on the direct neighbor property discussed in Section 4.1. We argue that the second NN must be among the direct neighbors of the first NN. Once we identify the second NN, we continue by including the neighbors of the second NN to find the third NN and so on. This search algorithm is based on the following Lemma which is derived from the properties of  $TNI$  and  $LNI$ .

**Lemma 4.** *The  $i$ -th nearest neighbor of  $q$  is always among the neighbors of the  $i-1$  nearest neighbors of  $q$ .*

**Algorithm 1** NN-Algorithm( $q, \text{TNI}, \text{LNI}$ )

---

```

1: //  $q$ : location of the query object,
2: //  $S$ : an array containing the candidate set
3: //  $H$ : a minimum heap,  $p$ : the first NN
4: Initialize  $S$  and  $H$ ;
5:  $p \leftarrow \text{contain}_{\text{TNI}}(q)$ ;
6: if  $p$  is null then
7:    $S \leftarrow \text{contain}_{\text{LNI}}(q)$ ;
8:   for each data object  $s_i$  in  $S$  do
9:      $\text{computeTDFP}(q, s_i, t)$ ;
10:    insert  $s_i$  to  $H$ ;
11:   end for
12:    $p \leftarrow \text{deHeap } H$ ;
13: end if
14: return  $p$ ;
```

---

(a) Algorithm 1

**Algorithm 2**  $k$ NN-Algorithm( $\text{TNI}, \text{LNI}, q, k$ )

---

```

1: //  $N$ : an array of NN set
2: //  $H$ : a minimum heap,  $p$ : any NN,  $k$ : number of NN
3: Initialize  $H, N$ 
4:  $p \leftarrow \text{NN-Algorithm}(q, \text{TNI}, \text{LNI})$ ;
5: add  $p$  to  $N$ 
6: while  $N.\text{size} \leq k$  do
7:   for each neighbor  $s_i$  of  $N$  do
8:      $\text{computeTDFP}(q, s_i, t)$ 
9:     add  $s_i$  to  $H$ ;
10:   end for
11:    $p \leftarrow \text{deHeap } H$ ; // find next NN
12:   add  $p$  to  $N$ 
13: end while
14: return  $N$  // return  $k$ NN
```

---

(b) Algorithm 2

**Fig. 8.** kNN query algorithm in time-dependent road networks

*Proof.* We prove this lemma by induction. We prove the base case (i.e., the second NN is a direct neighbor of the first NN of  $q$ ) by contradiction. Consider Figure 9 where  $p_2$  is the first NN of  $q$ . Assume that  $p_5$  (which is not a direct neighbor of  $p_2$ ) is the second NN of  $q$ . Since  $p_2$  and  $p_5$  are not direct neighbors, a point  $w$  on the time-dependent shortest path between  $q$  and  $p_5$  can be found that is outside both  $LC(p_2)$  and  $LC(p_5)$ . However,  $p_5$  cannot be a candidate NN for  $w$ , because  $w$  is not in  $LC(p_5)$ . Thus, there exists another object such as  $p_1$  which is closer to  $w$  as compared to  $p_5$ . Therefore,  $TDFP(w, p_5, t) > TDFP(w, p_1, t)$ . However, as shown in Figure 9, we have  $TDFP(q, p_5, t) = TDFP(q, w, t) + TDFP(w, p_5, t) > TDFP(q, w, t) + TDFP(w, p_1, t) = TDFP(q, p_1, t)$ . Thus,  $p_5$  is farther from  $q$  than both  $p_2$  and  $p_1$ , which contradicts the assumption that  $p_5$  is the second NN of  $q$ . The proof of inductive step is straight forward and similar to the above proof by contradiction; hence, due to lack of space, we omit the details.  $\square$

The complete TD- $k$ NN query answering process is given in Algorithm 2. Algorithm 2 calls Algorithm 1 to find the first NN and add it to  $N$ , which maintains the current set of nearest neighbors (Lines 4-5). To find the remaining  $k - 1$  NNs, we expand the search area by including the neighboring loose cells of the first NN. We compute the TDSP for each candidate and add each candidate to a minimum heap (Lines 9) based on its time-dependent travel-time to  $q$ . Thereafter, we select the one with minimum distance as the second NN (Line 11). Once we identify the second NN, we continue by investigating the neighbor loose cells of the second NN to find the third NN and so on. Our experiments show that the average number of neighbors for a data object is a relatively small number less than 9 (see Section 5.2).

**Time-dependent Fastest Path Computation.** As we explained, once the nearest neighbor of  $q$  is found and the candidate set is determined, the time-dependent fastest path from  $q$  to all candidates must be computed in order to find the next NN. Before we explain our TDFP computation, we note a very useful property of loose cells. That is, given  $p_i$  is the nearest neighbor of  $q$ , the time-dependent shortest path from  $q$  to  $p_i$  is guaranteed to be in  $LC(p_i)$  (see Lemma 5). This property indicates that we only need

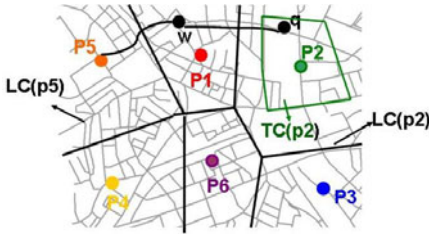


Fig. 9. Second NN Example

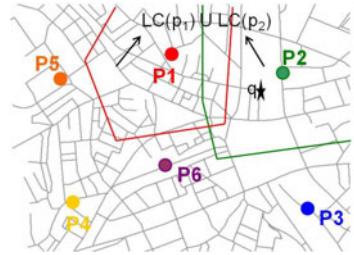


Fig. 10. TDSP localization

to consider the edges contained in the loose cell of  $p_i$  when computing TDFP from  $q$  to  $p_i$ . Obviously, this property allows us to localize the time-dependent shortest path search by extensively pruning the search space. Since the localized area of a loose cell is substantially smaller as compared to the complete graph, the computation cost of TDFP is significantly reduced. Note that the subnetwork bounded by a loose cell is on average  $1/n$  of the original network where  $n$  is the total number of sites.

**Lemma 5.** *If  $p_i$  is the nearest neighbor of  $q$ , then the time-dependent shortest path from  $q$  to  $p_i$  is guaranteed to be inside the loose cell of  $p_i$*

*Proof.* We prove by contradiction. Assume that  $p_i$  is the NN of  $q$  but a portion of TDFP from  $q$  to  $p_i$  passes outside of  $LC(p_i)$ . Suppose a point  $l$  on that outside portion of the path. Since  $l$  is outside  $LC(p_i)$ , then  $\exists p_j \in P, p_j \neq p_i$  that satisfies  $D_{LTT}(p_i, l) > D_{UTT}(p_j, l)$  and hence  $TDFP(p_i, l, t) > TDFP(p_j, l, t)$ . Then,  $TDFP(p_i, q, t) = TDFP(p_i, l, t) + TDFP(l, q, t) > TDFP(p_j, l, t) + TDFP(l, q, t) = TDFP(p_j, q, t)$ , which contradicts the fact that  $p_i$  is the NN of  $q$ .  $\square$

We note that for TD-kNN with  $k > 1$ , the TDFP from  $q$  to the  $k$ th nearest neighbor will lie in the combined area of neighboring cells. Figure 10 shows an example query with  $k > 1$  where  $p_2$  is assumed to be the nearest neighbor (and the candidate neighbors of  $p_2$  are,  $p_1, p_6$  and  $p_3$ ). To compute the TDFP from  $q$  to data object  $p_1$ , we only need to consider the edges contained in  $LC(p_1) \cup LC(p_2)$ . Below, we explain how we compute the TDFP from  $q$  to each candidate.

As initially showed by Dreyfus [7], the TDFP problem (in FIFO networks) can be solved by modifying any label-setting or label-correcting static shortest path algorithm. The asymptotic running times of these modified algorithms are same as those of their static counterparts. With our approach, we implement a time-dependent A\* search (a label-setting algorithm) to compute TDFP between  $q$  and the candidate set. The main idea with A\* algorithm is to employ a heuristic function  $h(v)$  (i.e., lower-bound estimator between the intermediate node  $v_i$  and the target  $t$ ) that directs the search towards the target and significantly reduces the number of nodes that have to be traversed. With static road networks where the length of an edge is considered as the cost, the Euclidean distance between  $v_i$  and  $t$  is the lower-bound estimator. However, with time-dependent road networks, we need to come up with an estimator that never overestimates the travel-time between  $v_i$  and  $t$  for all possible departure-times (from  $v_i$ ). One simple

---

**Algorithm 3** TDFP( $G_T(V, E), q, d, t_s$ )

---

```

1: //  $q$ :source,  $d$ :target,  $t_s$ :departure-time from node  $v$ ,
2: //  $cost(v)$ :cost from  $s$  to  $v$ ,  $pre(v)$ :previous node in optimal path
3:  $Q \leftarrow$  set of nodes in  $LC(q)$  and  $LC(d)$ 
4:  $\forall v \in Q$   $cost(v) = \infty$ ,  $cost(q) = 0$ 
5: while  $Q$  is not empty do
6:    $v_i \leftarrow$  node in  $Q$  with smallest cost
7:   remove  $v_i$  from  $Q$ 
8:   IF  $v_i = d$  THEN return path
9:   for each neighbor  $v_j$  of  $v_i$ 
10:     $l(v_j) = cost(v_i) + h_{LC}(v_i) + TT(v_i, v_j, t_{v_i})$ 
11:    IF  $l(v_j) < cost(v_j)$  THEN
12:       $cost(v_j) = l(v_j)$ 
13:       $pre(v_j) = v_i$ 
14:       $t_{v_j} = d_t(v_i) + TT(v_i, v_j, t_{v_i})$ 
15: end while

```

---

**Fig. 11.** TDSP Algorithm

lower-bound is  $d_{euc}(v_i, t) / \max(\text{speed})$ , i.e., the Euclidean distance between  $v_i$  and  $t$  divided by the maximum speed among the edges in the entire network. Although this estimator is guaranteed to be a lower-bound between  $v_i$  and  $t$ , it is a very loose bound, hence yields insignificant pruning. Fortunately, our approach can use Lemma 5 to obtain a much tighter lower-bound. Since the shortest path from  $q$  to  $p_i$  is guaranteed to be inside  $LC(p_i)$ , we can use the maximum speed in  $LC(p_i)$  to compute the lower-bound. We outline our time-dependent A\* algorithm in Algorithm 3 where essential modifications (as compared to [7]) are in Lines 3, 10 and 14. As mentioned, to compute TDFP from  $q$  to candidate  $p_i$ , we only consider the nodes in the loose cell that contains  $q$  and  $LC(p_i)$  (Line 3). To compute the labels for each node, we use arrival time and the estimator (i.e.,  $cost(v_i) + h_{LC}(v_i)$  where  $h_{LC}(v_i)$  is the lower-bound estimator calculated based on the maximum speed in the loose cell) to each node that form the basis of the greedy algorithm (Line 10). In Lines 10 and 14,  $TT(v_i, v_j, t_{v_i})$  finds the time-dependent travel-time from  $v_i$  to  $v_j$  (see Section 3).

## 5 Experimental Evaluation

### 5.1 Experimental Setup

We conducted several experiments with different spatial networks and various parameters (see Figure 12) to evaluate the performance of TD- $k$ NN. We run our experiments on a workstation with 2.7 GHz Pentium Duo Processor and 12GB RAM memory. We continuously monitored each query for 100 timestamps. For each set of experiments, we only vary one parameter and fix the remaining to the default values in Figure 12. With our experiments, we measured the tight cell hit ratio and the impact of  $k$ , data and query object cardinality as well as the distribution. As our dataset, we used Los Angeles ( $LA$ ) and San Joaquin ( $SJ$ ) road networks with 304,162 and 24,123 segments, respectively.

We evaluate our proposed techniques using a database of actual time-dependent travel-times gathered from real-world traffic sensor data. For the past 1.5 year, we have

been collecting and archiving speed, occupancy, volume sensor data from a collection of approximately 7000 sensors located on the road network of LA. The sampling rate of the data is 1 reading/sensor/min. Currently, our database consists of about 900 million sensor reading representing traffic patterns on the road network segments of LA. In order to create the time-dependent edge weights of  $SJ$ , we developed a system [5] that synthetically generates time-dependent edge weights for  $SJ$ .

## 5.2 Results

**Impact of Tight Cell Hit Ratio and Direct Neighbors.** As we explained, if a  $q$  is located in a certain tight cell  $TC(p_i)$ , our algorithm immediately reports  $p_i$  as the first NN. Therefore, it is essential to assess the coverage area of the tight cells over the entire network. Figure 13(a) illustrates the coverage ratio of the tight cells with varying data object cardinality (ranging from 1K to 20K) on two data sets. As shown, the average tight cell coverage is about %68 of the entire network for both LA and SJ. This implies that the first NN of a query can be answered immediately with a ratio of 7/10 with no further computation. Another important parameter affecting the TD- $k$ NN algorithm is the average number of direct neighbors for each data object. Figure 13(b) depicts the average number of neighbor cells with varying data object cardinality. As shown, the average number of neighbors is less than 9 for both LA and SJ.

Parameters	Default	Range
Number of objects	10 (K)	1,5,10,15,20(K)
Number of queries	3 (K)	1,2,3,4,5 (K)
Number of k	20	1,10,20,30,40,50
Object Distribution	Uniform	Uniform, Gaussian
Query Distribution	Uniform	Uniform, Gaussian

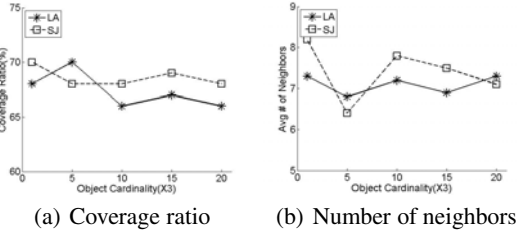


Fig. 12. Experimental Parameters

Fig. 13. Time-dependent fastest path localization

As mentioned, in [7] we developed an incremental network expansion algorithm (based on [7]) to evaluate  $k$ NN queries in time-dependent networks. Below we compare our results with this naive approach. For the rest of the experiments, since the experimental results with both LA and SJ networks differ insignificantly and due to space limitations, we only present the results from LA dataset.

**Impact of  $k$ .** In this experiment, we compare the performance of both algorithms by varying the value of  $k$ . Figure 14(a) plots the average response time versus  $k$  ranging from 1 to 50 while using default settings in Figure 12 for other parameters. The results show that TD- $k$ NN outperforms naive approach for all values of  $k$  and scales better with the large values of  $k$ . As illustrated, when  $k=1$ , TD- $k$ NN generates the result set almost instantly. This is because a simple `contain()` function is enough to find the first NN. As the value of  $k$  increases, the response time of TD- $k$ NN increases at linear rate.

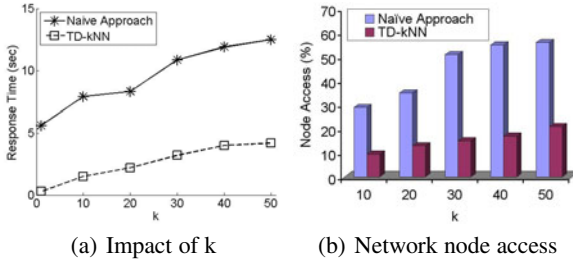


Fig. 14. Response time and node access versus  $k$

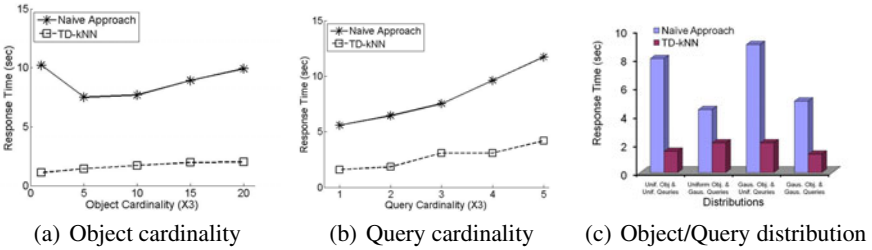


Fig. 15. Impact of  $k$

Because, TD- $k$ NN, rather than expanding the search blindly, benefits from localized computation. In addition, we compared the average number of network node access with both algorithms. As shown in Figure 14(b), the number of nodes accessed by TD- $k$ NN is less than the naive approach for all values of  $k$ .

**Impact of Object and Query Cardinality.** Next, we compare the algorithms with respect to cardinality of the data objects ( $P$ ). Figure 15(a) shows the impact of  $P$  on response time. The response time linearly increases with the number of data objects in both methods where TD- $k$ NN outperforms the naive approach for all cases. From  $P=1K$  to  $5K$ , the performance gap is more significant. This is because, for lower densities where data objects are possibly distributed sparsely, naive approach requires larger portion of the network to be retrieved. Figure 15(b) shows the impact of the query cardinality ( $Q$ ) ranging from  $1K$  to  $5K$  on response time. As shown, TD- $k$ NN scales better with larger  $Q$  and the performance gap between the approaches increases as  $Q$  grows.

**Impact of Object/Query Distribution.** Finally, we study the impact of object, query distribution. Figure 15(c) shows the response time of both algorithms where the objects and queries follow either uniform or Gaussian distributions. TD- $k$ NN outperforms the naive approach significantly in all cases. TD- $k$ NN yields better performance for queries with Gaussian distribution. This is because as queries with Gaussian distribution are clustered in the network, their nearest neighbors would overlap hence allowing TD- $k$ NN to reuse the path computations.

## 6 Conclusion and Future Work

In this paper, we proposed a time-dependent  $k$  nearest neighbor search algorithm (TD- $k$ NN) for spatial networks. With TD- $k$ NN, unlike the existing studies, we assume the edge weights of the network are time varying rather than fixed. In real-world, time-varying edge utilization is inherit in almost all networks (e.g., transportation, internet, social networks). Hence, we believe that our approach yields a much more realistic scenario and is applicable to  $k$ NN applications in other domains. We intend to pursue this study in two directions. First, we plan to investigate new data models for effective representation of time-dependent spatial networks. This is critical in supporting development of efficient and accurate time-dependent algorithms, while minimizing the storage and cost of the computation. Second, we intend to study a variety of other spatial queries (including continuous  $k$ NN, range and skyline queries) in time-dependent networks.

## References

1. Chabini, I.: The discrete-time dynamic shortest path problem: Complexity, algorithms, and implementations. *Journal of Transportation Research Record*, 16–45 (1999)
2. Cho, H.-J., Chung, C.-W.: An efficient and scalable approach to cnn queries in a road network. In: *Proceedings of VLDB (2005)*
3. Cooke, L., Halsey, E.: The shortest route through a network with timedependent internodal transit times. *Journal of Mathematical Analysis and Applications* (1966)
4. Demiryurek, U., Kashani, F.B., Shahabi, C.: Towards  $k$ -nearest neighbor search in time-dependent spatial network databases. In: Kikuchi, S., Sachdeva, S., Bhalla, S. (eds.) *Databases in Networked Information Systems*. LNCS, vol. 5999, pp. 296–310. Springer, Heidelberg (2010)
5. Demiryurek, U., Pan, B., Kashani, F.B., Shahabi, C.: Towards modeling the traffic data on road networks. In: *Proceedings of SIGSPATIAL-IWCTS (2009)*
6. Ding, B., Yu, J.X., Qin, L.: Finding time-dependent shortest paths over graphs. In: *Proceedings of EDBT (2008)*
7. Dreyfus, P.: An appraisal of some shortest path algorithms. *Journal of Operation Research* 17 (1969)
8. Erwig, M., Hagen, F.: The graph voronoi diagram with applications. *Journal of Networks* 36 (2000)
9. George, B., Kim, S., Shekhar, S.: Spatio-temporal network databases and routing algorithms: A summary of results. In: Papadias, D., Zhang, D., Kollios, G. (eds.) *SSTD 2007*. LNCS, vol. 4605, pp. 460–477. Springer, Heidelberg (2007)
10. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: *Proceedings of SIGMOD (1984)*
11. Halpern, J.: Shortest route with time dependent length of edges and limited delay possibilities in nodes. *Journal of Mathematical Methods of Operations Research* 21 (1969)
12. Hu, H., Lee, D.L., Xu, J.: Fast nearest neighbor search on road networks. 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. 186–203. Springer, Heidelberg (2006)
13. Huang, X., Jensen, C.S., Saltenis, S.: The island approach to nearest neighbor querying in spatial networks. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) *SSTD 2005*. LNCS, vol. 3633, pp. 73–90. Springer, Heidelberg (2005)

14. Kanoulas, E., Du, Y., Xia, T., Zhang, D.: Finding fastest paths on a road network with speed patterns. In: Proceedings of ICDE (2006)
15. Kolahdouzan, M., Shahabi, C.: Voronoi-based k nn search in spatial networks. In: Proceedings of VLDB (2004)
16. Lauther, U.: An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In: Geoinformation and Mobilitat (2004)
17. Navteq, <http://www.navteq.com> (last visited January 2, 2010)
18. Orda, A., Rom, R.: Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM* 37 (1990)
19. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial networks. In: Proceedings of VLDB (2003)
20. Samet, H., Sankaranarayanan, J., Alborzi, H.: Scalable network distance browsing in spatial databases. In: Proceedings of SIGMOD (2008)
21. TeleAtlas, <http://www.teleatlas.com> (last visited January 2, 2010)
22. Wagner, D., Willhalm, T.: Geometric speed-up techniques for finding shortest paths in large sparse graphs. In: Proceedings of Algorithms-ESA (2003)



# Hybrid Indexing and Seamless Ranking of Spatial and Textual Features of Web Documents

Ali Khodaei<sup>1</sup>, Cyrus Shahabi<sup>1</sup>, and Chen Li<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Southern California, Los Angeles, CA, USA

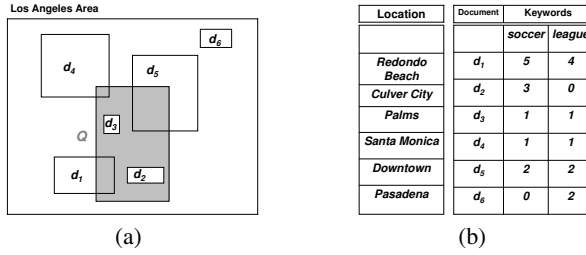
<sup>2</sup> Department of Computer Science, University of California at Irvine, CA, USA  
{khodaei, shahabi}@usc.edu, chenli@ics.uci.edu

**Abstract.** There is a significant commercial and research interest in location-based web search engines. Given a number of search keywords and one or more locations that a user is interested in, a location-based web search retrieves and ranks the most textually and spatially relevant web pages. In this type of search, both the spatial and textual information should be indexed. Currently, no efficient index structure exists that can handle both the spatial and textual aspects of data simultaneously and accurately. Existing approaches either index space and text separately or use inefficient hybrid index structures with poor performance. Moreover, most of these approaches cannot accurately *rank* web-pages based on a combination of space and text and are not easy to integrate into existing search engines. In this paper, we propose a new index structure called Spatial-Keyword Inverted File to handle location-based web searches in an integrated/efficient manner. To seamlessly find and rank relevant documents, we develop a new distance measure called spatial tf-idf. We propose four variants of spatial-keyword relevance scores and two algorithms to perform top-k searches. As verified by experiments, our proposed techniques outperform existing index structures in terms of search performance and accuracy.

## 1 Introduction

There is a large amount of location-based information generated and used by many applications. The Internet is the most popular source of data with location-specific information, such as documents describing schools at certain regions, Wikipedia pages containing spatial information and images with annotations and information about the places they were taken. Users of such a web-based application often need to query the system by providing requirements on a location as well as keywords in order to find relevant documents, as illustrated by the following example.

Suppose we have a collection of web pages, and each page describes objects for a specific location, such as a district, a city, or a county. Objects can be schools, real-state agencies, golf courses, and sports teams. We want to build a system to allow users to search on these documents. Consider a user, Mike, who moves to the central part of Los Angeles. He likes to play soccer, and wants to find soccer leagues in this area so that he can choose one to join. He submits a query to the system with two keywords “soccer league” and specifies “Central Los Angeles” as the location restriction. Figure 1 shows the location of his query represented as a shaded region. Our goal is to find the best documents that are of interest to Mike.



**Fig. 1.** A spatial-keyword query on documents with location information

Suppose there are six documents in the repository with locations close to the Central Los Angeles. Figure 1(a) shows these locations represented as rectangles. In addition, each document has text keywords in its content. Figure 1(b) shows the frequencies of the two query keywords in these documents. We want to find the most relevant results (documents) to the query. The result cannot be found with a simple keyword-only query since none of the documents may have the actual keywords “Central Los Angeles” or even “Los Angeles” in them.

One way to answer the query is to find the documents with a location contained in the query region and with the two query keywords, as suggested in several studies in the literature [2,6,10]. Using this approach, we can only find document  $d_3$  as an answer, since it satisfies both conditions. Document  $d_5$  is not an answer since even though it has both query keywords, its region is not totally contained in the query region. One major limitation of this approach is that many documents with partial spatial and/or textual matching will not be considered, even though they include information that could be interesting to the user.

In this paper we show how to support spatial-keyword queries on documents with spatial information. We demonstrate how to rank documents by seamlessly combining spatial and textual features, in order to find highly relevant answers to user queries. In our running example, an interesting question is how to measure the relevance of a document to the query. Intuitively, a document could be of interest to the user if it has at least one of the query keywords, and its location is close to the region mentioned in the query. Document  $d_6$  is not very relevant to the query, since its region is far from the query region. The other five documents are all overlapping with the query region, and thus could be potentially of interest to the user. Hence, we need to rank them since the user may be only interested in the most relevant documents. However, it is not clear how to measure the relevance of the documents to user query. For example, it is clear that document  $d_3$  should have a high relevance since its region is contained in the query region, and both query keywords appear in the document. However, it is not clear how relevant  $d_2$  is to the query, since even though its region is contained in the query region, it does not have the keyword “league.” The other documents,  $d_1$ ,  $d_4$ , and  $d_5$ , all have the two query keywords, but with different frequencies, and they have different amounts of overlapping areas with the query region.

In this paper, we present a ranking method that considers both the spatial overlap of a document with a query and the frequencies of the query keywords in the document in order to compute a relevance score of the document to the query. We present a new

scoring mechanism to calculate the spatial relevance of a document with respect to a query, and propose a method to combine the spatial relevance and the textual relevance.

Given a good ranking method for documents, a natural question to ask is how to efficiently index and search the location-specific documents. There are several challenges. First, space and text are two totally different data types requiring different index structures. For instance, conventional text engines are set-oriented while location indexes are usually based on two-dimensional and Euclidean spaces. Second, the ranking and search processes should not be separated. Otherwise, the ranking process will rank all the candidate documents (instead of only the relevant documents), making the query processing inefficient. Third, the meaning of spatial relevance and textual relevance and a way to combine them using the proposed index structure have to be defined accurately. Finally, it should be easy to integrate the index structure into existing search engines.

To solve the above problems, we propose a new hybrid index structure called *Spatial-Keyword Inverted File* (“SKIF” for short), which can handle the spatial and textual features of data simultaneously and in a similar manner. SKIF is an inverted file capable of indexing and searching on both textual and spatial data in a similar, integrated manner. Towards this end, the space is partitioned into a number of grid cells and each cell is treated similar to a textual keyword. We describe the structure of SKIF, and present two efficient algorithms for answering a ranking query using SKIF.

To summarize, we have the following contributions in this paper:

- We define the problem of ranking queries on documents with spatial and textual information.
- We develop a new scoring method called *spatial tf-idf* to compute the spatial relevance of a document to a query, and combine both spatial relevance and textual relevance for a query.
- We develop an efficient hybrid index structure for indexing both the spatial and textual aspects of the documents. We present two algorithms for answering a ranking query using the structure.
- We have conducted an experimental evaluation on real and synthetic datasets to show that our techniques can answer ranking queries efficiently and accurately.

## 2 Related Work

Existing index structures for handling the spatial-keyword queries can be categorized into two broad groups: 1) individual index structures, and 2) hybrid index structures. Individual index structures use one index for each set of features of the data (space or text). The index structures of choice for the spatial data are usually grid,  $R^*$ -tree or quadtree. For text, inverted files are often used. Using separate index structures, documents satisfying the textual part of the query and documents satisfying the spatial part of the query are retrieved separately using the textual and spatial indexes, respectively.

Hybrid index structures combine the textual and spatial indexes into one index structure. Two basic designs introduced in [1] are *inverted file- $R^*$ -tree* and  *$R^*$ -tree-inverted file*. *Inverted file- $R^*$ -tree* is essentially an inverted file on top of  $R^*$ -trees. For a given spatial-keyword query, the query keywords are filtered using the inverted file and then  $R^*$ -trees corresponding to those keywords are traversed to search/filter based on the

spatial features of the data. This structure performs well only when the number of keywords is very small. *R\*-tree-inverted file* is an R\*-tree on top of inverted files. For a given spatial-keyword query, R\*-tree's leaf nodes intersecting the query location are retrieved first and then all the inverted lists corresponding to those keywords are traversed. The main disadvantage of this method is in its spatial filtering step, which usually generates many candidate objects. Two other structures are those presented in [6] and [10]. Both index structures use grid as a spatial index and inverted file as a textual index. With both approaches, spatial and textual search processes are separated and query processing require two stages. Neither of the approaches can support spatial-keyword relevance ranking.

Several improved hybrid index structures are introduced more recently. In [2] a hybrid index structure called KR\*-tree is proposed. KR\*-tree extends R\*-tree-inverted file structure by augmenting each node of R-tree with the list of all the keywords appearing in the objects of that subtree. At the query time, KR\*-tree is traversed and for each node, not only the spatial intersection with the query region is checked but the node is also checked for the presence of the query keywords. The result to the query are the objects contained in the query region that have all the query keywords (AND semantics). KR\*-tree is only good for spatial objects with a small number of keywords. Another hybrid index structure called IR<sup>2</sup>-tree is presented in [3], which combines R-tree with signature files. With this method, each node in R-tree is augmented with a *signature* representing the union of the keywords (text) of the objects in the subtree rooted at that node. Similar to KR\*-tree, IR<sup>2</sup>-tree can identify the subtrees that do not contain the query keywords and eliminate them from the search process early on. Using IR<sup>2</sup>-tree, the final result set is a ranked list of objects containing all the query keywords in order of their distances to the query point. IR<sup>2</sup>-tree performs better than index structures in [1] and [2] but still has its own shortcomings. At times, the signature files are not able to eliminate the objects not satisfying the query keywords (false hits). This results in loading and reading more objects, which is costly. Furthermore, the performance gets worse when the number of query keywords increases or when the final result is very far from the query point.

Very recently another hybrid index structure called IR-tree is presented, which also combines R-tree with the inverted files [9]. With this index structure, each node of the R-tree is augmented with an inverted file for the objects contained in the sub-tree rooted at that node. IR-tree is the most similar approach to our work since it considers space and text together.

Nevertheless, there are some major problems with IR-tree. First, one inverted file needs to be stored and possibly accessed for each node in the tree. For web, the total number of documents and the total number of keywords are very large, resulting in huge number of nodes in the tree and also large inverted files for each node. Another problem with IR-tree is that during the search process, it often needs to visit few nodes in the tree containing no relevant results. Finally, it is not clear that whether ranking proposed in [9] is an accurate spatial-keyword relevance ranking (see Section 6).

There are many other relevant topics such as extraction of geographical information [7][12], geo-coding of documents' locations [11], and geographic crawling [13]. In this

paper, we only focus on index structures, relevance ranking, and search algorithms for spatial-keyword queries.

### 3 Preliminaries

#### 3.1 Problem Definition

We assume a collection  $D = \{d_0, d_1, \dots, d_n\}$  of  $n$  documents (web pages). Each document  $d$  is composed of a set of keywords  $K_d$  and a set of locations  $L_d$ . Each location is represented by a minimum bounding rectangle (MBR) although any other arbitrary shape can be used.

**Spatial-keyword query:** A spatial-keyword query is defined as  $Q = \langle K_q, L_q \rangle$ , where  $L_q$  is the spatial part of query specified as one or more minimum bounding rectangles and  $K_q$  is a set of keywords in the query.

**Spatial relevance:** Spatial relevance between a document  $d$  and the query  $q$  is defined based on the type of the spatial relationship that exists between  $L_d$  and  $L_q$ . We focus only on the *overlap* relationship, although our approach can easily be extended to cover other spatial relationships. Subsequently, we define spatial relevance as follows: A document  $d$  and the query  $q$  are spatially relevant if at least one of the query's MBRs has a non-empty intersection with one of the document's MBRs, i.e.,  $L_q \cap L_d \neq \emptyset$ . The larger the area of the intersection is, the more spatially relevant  $d$  and  $q$  are. We denote spatial relevance of document  $d$  to query  $q$  by  $sRel_q(d)$ .

**Textual relevance:** A document  $d$  is textually relevant to the query  $q$  if there exists at least one keyword belonging to both  $d$  and  $q$ , i.e.,  $K_q \cap K_d \neq \emptyset$ . The more keywords  $q$  and  $d$  has in common, the more they are textually relevant. We represent textual relevance of document  $d$  to query  $q$  by  $kRel_q(d)$ .

**Spatial-keyword relevance:** A document  $d$  is spatial-keyword relevant to the query  $q$  if it is both spatially and textually relevant to the query  $q$ . Spatial-keyword relevance can be defined by a monotonic scoring function  $F$  of textual and spatial relevances. For example,  $F$  can be the weighted sum of the spatial and textual relevances:  $F_q(d) = \alpha \cdot sRel_q(d) + (1 - \alpha) \cdot kRel_q(d)$ .  $\alpha$  is a parameter assigning relative weights to spatial and textual relevances. The output of function  $F_q(d)$  is the spatial-keyword relevance score of document  $d$  and query  $q$ , and is denoted by  $skRel_q(d)$ . In Section 4 we show in details how to calculate spatial-keyword relevance using our proposed index.

**Spatial-keyword search:** A spatial-keyword search identifies all the documents (web pages) that are *spatial-keyword* relevant to  $q$ . The result is the top- $k$  most spatial-keyword relevant ranked documents sorted based on documents' spatial-keyword relevance scores. The parameter  $k$  is determined by the user.

#### 3.2 Textual Relevance

**tf-idf Score.** All current textual (keyword) search engines use a *similarity measure* to rank and identify potential (textual) relevant documents. In most keyword queries, a similarity measure is determined by using the following important parameters:

- $f_{d,k}$ : the frequency of keyword  $k$  in document  $d$
- $\max(f_{d,k})$ : maximum value of  $f_{d,k}$  over all the keywords in document  $d$
- $\overline{f_{d,k}}$ : normalized  $f_{d,k}$ , which is  $\frac{f_{d,k}}{\max(f_{d,k})}$
- $f_k$ : the number of documents containing one or more occurrences of keyword  $k$

Using these values, three monotonicity observations are enforced [4]: (1) less weight is given to the terms that appear in many documents; (2) more weight is given to the terms that appear many times in a document; and (3) less weight is given to the documents that contain many terms. The first property is quantified by measuring the inverse of frequency of keyword  $k$  among the documents in the collection. This factor is called *inverse document frequency* or the *idf* score. The second property is quantified by the raw frequency of keyword  $k$  inside a document  $d$ . This is called *term frequency* or *tf* score, and it describes how well that keyword describes the contents of the document [5][8]. The third property is quantified by measuring the total number of keywords in the document. This factor is called *document length*.

A simple and very common formula to calculate the similarity between a document  $d$  and the query  $q$  is shown in Equation [1]

$$w_{q,k} = \ln\left(1 + \frac{n}{f_k}\right); \quad w_{d,k} = \ln\left(1 + \overline{f_{d,k}}\right);$$

$$W_d = \sqrt{\sum_k w_{d,k}^2}; \quad W_q = \sqrt{\sum_k w_{q,k}^2}; \quad S_{q,d} = \frac{\sum_k w_{d,k} \cdot w_{q,k}}{W_d \cdot W_q}. \tag{1}$$

Variable  $w_{d,k}$  captures the *tf score* while variable  $w_{q,k}$  captures the *idf score*.  $W_d$  represents *document length* and  $W_q$  is query length (which can be neglected since it is a constant for a given query). Finally,  $S_{q,d}$  is the similarity measure showing how relevant document  $d$  and query  $q$  are. In this case (textual context) it is the same as  $tRel_q(d)$ .

keyword $k$	$f_k$	Inverted list for $k$
soccer	5	$\langle 1, 1 \rangle \langle 2, 1 \rangle \langle 3, 1 \rangle \langle 4, 1 \rangle \langle 5, 1 \rangle$
league	5	$\langle 1, 0.8 \rangle \langle 3, 1 \rangle \langle 4, 1 \rangle \langle 5, 1 \rangle \langle 6, 1 \rangle$

Fig. 2. Inverted file for Example 1

**Inverted Files.** Inverted file is the most popular and very efficient data structure for textual query evaluation. Inverted file is a collection of lists, one per keyword, recording the identifiers of the documents containing that keyword [4]. An inverted file consists of two major parts: vocabulary and inverted lists. The vocabulary stores for each keyword  $k$ : a count  $f_k$  showing number of documents containing  $k$ , and a pointer to the corresponding inverted list. The second part of inverted file is a set of inverted lists, each corresponding to a keyword. Each list stores for the corresponding keyword  $k$ : identifiers  $d$  of documents containing  $k$ , and normalized frequencies  $\overline{f_{d,k}}$  of term  $k$  in document  $d$  [4]. A complete inverted file for Example 1 is shown in Figure [2]

## 4 Seamless Spatial-Keyword Ranking

In this section, we define new scoring mechanism to calculate the spatial relevance and spatial-keyword relevance scores. Following the same intuitions and concepts used

in regular (textual) searches, we define new concepts and parameters for spatial data. Most notably, inspired by tf-idf in textual context, we define a new scoring mechanism called *spatial tf-idf* for the spatial context. Using (textual) tf-idf scores and spatial tf-idf scores, the spatial-keyword relevance is defined and can be used to rank the documents based on both the spatial and textual aspects of the data, simultaneously and efficiently. We discuss two different approaches to calculate the spatial-keyword relevance using the spatial tf-idf score. Several variants of the final similarity measure is also presented.

#### 4.1 Spatial tf-idf

In order to be able to use the analogous ideas used in the regular tf-idf score, we need to treat spatial data similar to textual data. Most importantly, we need to represent space which is coherent and continuous in nature, as disjunct and set-oriented units of data - similar to the textual keywords. Hence, we partition the space into grid cells and assign unique identifiers to each cell. Therefore, each location in document can be associated with a set of cell identifiers. Since we are using overlap as our main spatial query type, these cells are defined as the cells which overlap with the document location. With spatial tf-idf, the overlap of a cell with the document is analogous to the existence of a keyword in document with tf-idf. However, knowing the overlapping cells is not enough. We need to know how well a cell describes the spatial content of the document. We use the overlap area between each cell and the document to provide a measure of how well that cell describes the document. Analogous to *frequency of term  $t$  in document  $d$* , we define *frequency of cell  $c$  in document  $d$*  as follows:  $f_{d,c} = \frac{L_d \cap c}{c}$  which is the area of overlap between the document location  $L_d$  and cell  $c$  divided by the area of cell  $c$ . Similar to the frequency of a keyword which describes how well the keyword describes the documents textual contents ( $K_d$ ), the frequency of a cell describes how well the cell describes the documents spatial contents ( $L_d$ ). The more the overlap, the better this cell describes the document location and viceversa.

Now we can define the following parameters analogous to those of Section 3.2:

- $f_{d,c}$ : the frequency of cell  $c$  in document  $d$
- $\max(f_{d,c})$ : maximum value of  $f_{d,c}$  over all the cells in document  $d$
- $\overline{f_{d,c}}$ : normalized  $f_{d,c}$ , which is  $\frac{f_{d,c}}{\max(f_{d,c})}$
- $f_c$ : the number of documents containing one or more occurrences of cell  $c$

Using the above parameters, we revisit three monotonicity properties discussed in Section 3.2, this time in spatial context: (1) less weight is given to cells that appear in many documents; (2) more weight is given to cells that overlap largely with a document; and (3) less weight is given to documents that contain many cells.

The first property is quantified by measuring the inverse of frequency of a cell  $c$  among the documents in the collection. We call this *spatial inverse document frequency* or  $idf_s$  score. The second property is quantified by the frequency of cell  $c$  in document  $d$  (as defined earlier). This is called *spatial term frequency* or  $tf_s$  score and describes how well that cell describes the document spatial contents (i.e.  $L_d$ ). The third property is quantified by measuring the total number of cells in the document. This factor is called *document spatial length*.

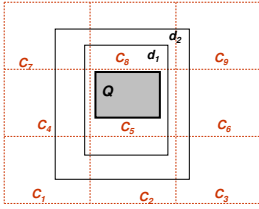


Fig. 3. Properties (2) and (3)

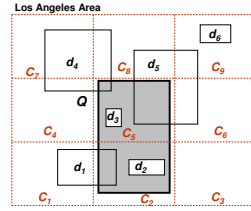


Fig. 4. Example 1 on the grid

Among the above properties, properties (2) and (3) are more intuitive. Property (2) states that more weight should be given to the cells having a large overlap area with the document. The larger the overlap, the better that cell describes the document location. For example, in Figure 3, cell  $c_8$  better describes the document  $d_2$  than cell  $c_9$ . Property (3) states that less weight should be given to those documents whose locations cover more cells. Assuming all the other parameters are equal, a document with a smaller coverage (fewer number of cells) should get a higher weight than a document with a larger coverage. To illustrate, consider Figure 3, where both documents  $d_1$  and  $d_2$  contain the query location and its corresponding cell (i.e.  $c_5$ ). In other words, they have equal spatial tf scores for cell  $c_5$ . Cell  $c_5$  also have identical spatial idf for all documents. Under these conditions (equal tf scores and equal idf scores), the smaller document ( $d_1$ ) should be ranked higher. This is analogous to the fact that in textual context, more weight is given to the documents that contain fewer keywords.

Contrary to properties (2) and (3), property (1) is not very intuitive. It states that less weight is given to the cells appearing in more documents. In the textual context, the idf score is a weighting factor determining the importance of each keyword independent of the query. It assigns more weight to keywords appearing in fewer documents, since those are more *meaningful* keywords. However, the definition of *meaningful cell* is not very clear in the spatial context. A popular cell (location) - a cell overlapping with many documents - is a very meaningful cell for some users/applications, while for some others, a distinctive cell (location) - cell appearing in few documents - is more meaningful. (in Example 1, one user may look for more popular locations for soccer leagues while another user may be interested in a less crowded, more private location). To cover both cases, we define spatial idf of cell  $c$  in two different ways: inverse of frequency of a cell  $c$  among the documents (*inverted idf<sub>s</sub>*) and direct frequency of a cell  $c$  among the documents (*direct idf<sub>s</sub>*).

### 4.2 Spatial-Keyword Relevance

In this section, we introduce two novel approaches for calculating spatial-keyword relevance between a document  $d$  and a query  $q$ . With the *single-score* approach, one similarity measure and one document length is used to combine the spatial relevance and textual relevance into one equation. With the *double-score* approach, spatial and textual relevance are calculated separately, using two document lengths, one for each relevance. Thus a new spatial similarity measure analogous to the textual similarity measure is defined. Both approaches can use the parameter  $\alpha$  to assign relative weights.



**Single-Score Approach.** After partitioning each document location to a set of cells, defining the spatial tf-idf score and creating one document spatial length for each document location, the cells are ready to be treated in a similar manner to the keywords. We define *term* as the smallest unit of data describing each document which is either a keyword or a cell. If we represent keywords associated with the document  $d$  by  $K_d$  and the cells associated with the same document by  $C_d$ , then the set of terms associated with document  $d$  is represented by  $T_d$  and defined as follows:

$$T_d = K_d \cup C_d.$$

Simply stated, the document’s terms are the union of the document’s keywords and cells. For Instance, in Example 1:  $T_{d_1} = \{\text{soccer, league, } c_1, c_2\}$  (see Figures [1\(b\)](#) and [4](#)). In order to be able to define a single similarity measure capturing both the textual and spatial relevances, we define the following parameters:

- $f_{d,t}$ : the frequency of term  $t$  in document  $d$
- $f_t$ : the number of documents containing occurrences of term  $t$

where each parameter gets its value from the corresponding parameter in the space or text domain (based on term type). For instance, value of  $f_{d,t}$  is equal to  $f_{d,k}$  when term is keyword and to  $f_{d,c}$  when term is cell. Having defined these new parameters, we can now easily redefine Equation [1](#), this time with terms instead of keywords. This is a new formulation capturing the keywords (textual relevance) and the cells (spatial relevance) in a unified manner.

$$\begin{aligned}
 w_{q,t} &= \begin{cases} (1 - \alpha) \cdot \ln(1 + \frac{f_t}{f_t}) & \text{if } t \text{ is keyword} \\ \alpha \cdot w_{q,c} & \text{if } t \text{ is a cell} \end{cases}; & w_{d,t} &= \begin{cases} (1 - \alpha) \cdot \ln(1 + \frac{f_{d,t}}{f_{d,t}}) & \text{if } t \text{ is keyword} \\ \alpha \cdot \ln(1 + \frac{f_{d,t}}{f_{d,t}}) & \text{if } t \text{ is cell} \end{cases}; \\
 \widehat{W}_d &= \sqrt{\sum_t w_{d,t}^2}; & \widehat{W}_q &= \sqrt{\sum_t w_{q,t}^2}; \\
 \widehat{S}_{q,d} &= \frac{\sum_t w_{d,t} \cdot w_{q,t}}{\widehat{W}_d \cdot \widehat{W}_q}.
 \end{aligned} \tag{2}$$

The variable  $w_{d,t}$  captures the *spatial-keyword term frequency* score ( $tf_{sk}$ ). The variable  $w_{q,t}$  captures the *spatial-keyword inverted document frequency* ( $idf_{sk}$ ). Parameter  $\alpha$  is integrated into the weighting scheme to capture the weighted relevance of space versus text.  $\widehat{W}_d$  represents *spatial-keyword document length* and  $\widehat{W}_q$  is (spatial-keyword) query length. Finally,  $\widehat{S}_{q,d}$  is the similarity measure showing how *spatial-keyword relevant* document  $d$  is to query  $q$ .

**Double-Score Approach.** In the *single-score* approach, keywords and cells are treated in exactly the same manner. Keywords and cells tf and idf scores are used in one equation and one similarity measure ( $\widehat{S}_{q,d}$ ) using one document length ( $\widehat{W}_d$ ) is used to calculate the final relevance score. There might be cases when most of the documents in the collection contain very large document location but very few keywords (or the opposite). In this situation, it is better to calculate the textual and spatial relevance scores separately. Hence, we discuss another approach to calculate the similarity measure between document  $d$  and query  $q$  in the spatial-keyword context. One can first calculate the spatial relevance and the textual relevance of document  $d$  and query  $q$  independently and then use an aggregation function to compute the overall spatial-keyword relevance

score. Using the spatial tf-idf parameters and the definitions, we calculate the spatial similarity measure between document  $d$  and query  $q$  analogous to the textual similarity measure as follows:

$$w_{q,c} = \begin{cases} \ln(1 + \frac{n}{f_c}) & \text{if inverted document frequency} \\ \ln(1 + \frac{f_c}{n}) & \text{if direct document frequency} \end{cases}; \quad w_{d,c} = \ln(1 + \overline{f_{d,c}});$$

$$W'_d = \sqrt{\sum_c w_{d,c}^2}; \quad W'_q = \sqrt{\sum_c w_{q,c}^2}; \quad S'_{q,d} = \frac{\sum_c w_{d,c} \cdot w_{q,c}}{W'_d \cdot W'_q}. \quad (3)$$

where  $S'_{q,d}$  is the spatial similarity measure between document  $d$  and query  $q$ . This value captures the spatial relevance  $sRel_q(d)$  defined in Section 3.1.

After calculating the spatial relevance using the above equation and computing the textual relevance using Equation 1, the aggregation function  $F$  can be used to calculate the final spatial-keyword relevance. More formally:  $skRel_q(d) = \alpha \cdot S'_{q,d} + (1 - \alpha) \cdot S_{q,d}$ .

**Variants.** We conclude this section by summarizing possible variants of the spatial-keyword relevance score. We defined two different approaches to calculate the spatial-keyword relevance scores. We also introduced two different ways to define the *spatial idf* factor score. Combining our two main approaches with the two definitions of the spatial idf score yields four different variants for our final similarity measure:

1. Single-Score with Inverted document frequency (*SSI*)  
Where  $skRel_q(d) = \widehat{S_{q,d}}$  and  $w_{q,c} = \ln(1 + \frac{n}{f_c})$
2. Single-Score with Direct document frequency (*SSD*)  
Where  $skRel_q(d) = \widehat{S_{q,d}}$  and  $w_{q,c} = \ln(1 + \frac{f_c}{n})$
3. Double-Score with Inverted document frequency (*DSI*)  
Where  $skRel_q(d) = \alpha \cdot sRel_q(d) + (1 - \alpha) \cdot kRel_q(d)$  and  $w_{q,c} = \ln(1 + \frac{n}{f_c})$
4. Double-Score with Direct document frequency (*DSD*)  
Where  $skRel_q(d) = \alpha \cdot sRel_q(d) + (1 - \alpha) \cdot kRel_q(d)$  and  $w_{q,c} = \ln(1 + \frac{f_c}{n})$

## 5 Spatial-Keyword Inverted File

Spatial-Keyword Inverted File (SKIF) is an inverted file capable of indexing and searching both the textual and spatial data in a similar, integrated manner using a single data structure. In this section, we first describe the structure of SKIF and the information it stores. Next, we show how spatial-keyword query evaluation is performed using SKIF. Two algorithms corresponding to our two approaches are presented. Finally, we discuss briefly how SKIF can be extended to more general cases.

### 5.1 SKIF Structure

Since SKIF is an inverted file, its structure is very similar to the structure of the regular inverted files. SKIF consists of two parts: vocabulary and inverted lists. The vocabulary contains all the *terms* in the system which includes all the (textual) keywords and cells (cell identifiers). For each distinct term, three values are stored in the vocabulary: 1)  $f_t$  representing the number of the documents containing the term  $t$ , 2) a pointer to

term $t$	$f_t$	type	Spatial-Keyword Inverted List for $t$
soccer	5	1	$\langle 1, 1 \rangle \langle 2, 1 \rangle \langle 3, 1 \rangle \langle 4, 1 \rangle \langle 5, 1 \rangle$
league	5	1	$\langle 1, 0.8 \rangle \langle 3, 1 \rangle \langle 4, 1 \rangle \langle 5, 1 \rangle \langle 6, 1 \rangle$
$c_1$	1	0	$\langle 1, 1 \rangle$
$c_2$	2	0	$\langle 1, 0.55 \rangle \langle 2, 1 \rangle$
$c_4$	1	0	$\langle 4, 0.25 \rangle$
$c_5$	3	0	$\langle 3, 1 \rangle \langle 4, 0.06 \rangle \langle 5, 1 \rangle$
$c_6$	1	0	$\langle 5, 0.35 \rangle$
$c_7$	1	0	$\langle 4, 1 \rangle$
$c_8$	2	0	$\langle 4, 0.3 \rangle \langle 5, 0.65 \rangle$
$c_9$	2	0	$\langle 5, 0.25 \rangle \langle 6, 1 \rangle$

Fig. 5. Spatial-keyword inverted file for Example 1

the corresponding inverted list and 3) the type of term which is used to help calculate the tf and idf scores. The second component of SKIF is a set of inverted lists each corresponding to a term. For the corresponding term  $t$ , each list stores the following values: identifiers of the documents containing term  $t$  and the normalized frequencies of term  $t$  for each document  $d$ . The latter is represented by  $\overline{f_{d,t}}$ . Figure 4 redraws the Example 1 on the grid and Figure 5 shows the complete SKIF for Example 1.

### 5.2 Query Processing

As discussed in Section 3.1, the spatial-keyword query consists of two parts: the query keywords  $K_q$  and the query location  $L_q$ . To process spatial-keyword queries, we first need to convert  $L_q$  to a set of cells  $C_q$ .  $C_q$  is the set of cells overlapping with the document location  $L_q$ . After calculating  $C_q$ , we define the set of terms associated with each query by  $T_q$  as follows:  $T_q = K_q \cup C_q$ .

Figures 1 and 2 show the algorithms to perform top-k spatial-keyword search using SKIF for the single-score and the double-score approaches respectively. Both the algorithms are very similar. Accumulators are used to store the partial similarity scores. The main difference is that Algorithm 1 uses one accumulator  $A_d$  while the Algorithm 2 uses two accumulators  $A_d$  and  $A'_d$ . After all the query terms are processed, similarity scores  $\widehat{S}_{q,d}$ ,  $S_{q,d}$  and  $S'_{q,d}$  are derived by dividing each accumulator value by the corresponding value  $\widehat{W}_d$ ,  $W_d$  and  $W'_d$  respectively. Finally, the  $k$  largest documents are identified and will be returned to the user.

### 5.3 Generalization

In this section we briefly show how we can extend SKIF into more general cases.

**Multiple Locations:** One of the advantages of our technique is that there is no limiting constraint on the representation of the document location. Instead of treating the document location as one large and sparse MBR, we can use several sperate, disjoint locations using SKIF. This is feasible because our final spatial relevance score can be computed by separately computing the spatial score of each cell intersecting with the various document locations. Another advantage of SKIF is its capability to represent the document location as any arbitrary shape and not necessarily as MBR or rectangle. The only information we need to calculate for spatial tf-idf score is the area of overlap between each cell and each document location.

**Algorithm 1.** single-score approach

---

```

1: Allocate an accumulator  $A_d$  for each document  $d$ 
2: Set  $A_d \leftarrow 0$ 
3: for each query term  $t$  in  $q$  do
4:   Calculate  $w_{q,t}$  and fetch the inverted list for  $t$ 
5:   for each pair  $\langle d, f_{d,t} \rangle$  in the inverted list do
6:     Calculate  $w_{d,t}$ 
7:     Set  $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$ 
8: Read the array of  $\widehat{W}_d$  values
9: for each  $A_d > 0$  do
10:  Set  $\widehat{S}_d \leftarrow A_d \div \widehat{W}_d$ 
11: Identify the  $k$  greatest  $\widehat{S}_d$  values

```

---

**Algorithm 2.** double-score approach

---

```

1: Allocate two accumulators  $A_d$  and  $A'_d$  for each document  $d$ 
2: Set  $A_d \leftarrow 0$ 
3: Set  $A'_d \leftarrow 0$ 
4: for each query term  $t$  in  $q$  do
5:   Calculate  $w_{q,t}$  and fetch the inverted list for  $t$ 
6:   for each pair  $\langle d, f_{d,t} \rangle$  in the inverted list do
7:     Calculate  $w_{d,t}$ 
8:     if type of  $t$  is a keyword then
9:       Set  $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$ 
10:    else
11:      Set  $A'_d \leftarrow A'_d + w_{q,t} \times w_{d,t}$ 
12: Read the array of  $W_d$  values
13: for each  $A_d > 0$  do
14:  Set  $S_d \leftarrow A_d \div W_d$ 
15: Read the array of  $W'_d$  values
16: for each  $A'_d > 0$  do
17:  Set  $S'_d \leftarrow A'_d \div W'_d$ 
18:  if  $A_d > 0$  then
19:     $\widehat{S}_d = \alpha \cdot S'_d + (1 - \alpha) \cdot S_d$ 
20: Identify the  $k$  greatest  $\widehat{S}_d$  values

```

---

**Points:** We assumed that each document location is a region. In the context of the web, this is a reasonable assumption, still in the rare cases when the document location is one geographical point (point  $p$ ) we can generalize our approach as follows. A circle centered at  $p$  with radius  $r$  is constructed. The MBR covering the circle is our new document location. Radius  $r$  is a parameter determined by user and also the context of the web-page (if available).

**Weights:** When querying the system, there are two types of weight users may want to manipulate 1) setting different weights to the spatial and textual relevance, and 2) setting different weights to different terms in the query. For the first scenario, we have used the parameter  $\alpha$  in this paper. SKIF can also support setting different weights for different terms. There are several existing methods to solve this problem for textual keywords. Since we are treating cells similar to keywords, those methods can also be applied to the spatial cells. As one possible solution, we define *query term weights*  $\alpha_{q,k}$  and  $\alpha_{q,c}$  as the weight of keyword  $k$  in query  $q$  and the weight of cell  $c$  in query  $q$ , respectively. By multiplying  $w_{d,k}$  and  $w_{d,c}$  values by  $\alpha_{q,k}$  and  $\alpha_{q,c}$ , respectively, query term weights are integrated into the relevance scores. This opens a wide possibility of complicated queries to the users.

**Leveraging Existing Search Engines:** One of the most practical advantages of the proposed approach is the fact that it can be integrated into the existing search engines easily and seamlessly. Since the structure of SKIF is very similar to the structure of the regular inverted files, same techniques used in regular search engines (built on inverted files) can be applied for our location-based search engine (built on SKIF).

The easy integration of our approach into the existing search engines is not only very beneficial for current search engines but also enables us to optimize SKIF using a body of work exists in this field. For example, *compression* techniques are very popular for inverted files [4][14]. Since the structure of the inverted lists are identical with both

SKIF and regular inverted files, no change is needed to apply the same compression techniques on SKIF. More interestingly, some of the optimization techniques seem to work better on SKIF. For instance, *caching* is another technique used in existing search engines. It is easy to see that with SKIF, by caching the inverted lists for the cells nearby the current query cell, we can improve the query performance significantly. It is very likely that nearby cells queried together or very close to each other.

## 6 Experiments

In this section, we experimentally evaluate the performance and accuracy of SKIF. Comparison is done with the most efficient proposed solutions: *MIR<sup>2</sup>-tree* [3] and *CDIR-tree* [9] which are optimized versions of IR<sup>2</sup>-tree and IR-tree, respectively. Since both of *MIR<sup>2</sup>-tree* and *CDIR-tree* use query points instead of query regions, we apply the following adjustment: Each query is executed using *MIR<sup>2</sup>-tree* and *CDIR-tree* separately for random query point  $q$  and for total number of results  $k$ . Subsequently, the farthest document in the union of the result sets is identified. Let  $r$  be the distance between  $q$  and this farthest document. We construct a circle centered at  $q$  with radius  $r$ , the MBR covering the circle is considered as the query location ( $L_q$ ).

**Table 1.** Dataset Details

Dataset	Total # of documents	Average # of unique keywords per document	Total # of unique keywords	Total # of keywords
DATASET1	19,841	64	31,721	1,269,824
DATASET2	250,000	230	50,000	57,500,000

Our experiments use two datasets which their properties are summarized in Table 1. DATASET1 is generated from a real world online web application called Shoah Foundation Visual History Archive (<http://college.usc.edu/vhi/>). Each document (testimony) is tagged with a set of textual and spatial keywords describing the content of the testimony. In preparing DATASET1, we extracted location names (spatial keywords) from all the testimonies and geo-coded the location names into spatial regions using Yahoo! Placemaker (<http://developer.yahoo.com/geo/placemaker/>). We run our experiments on all the documents in US. For DATASET1, we partition the space into  $100km \times 100km$  cells. DATASET2 is generated synthetically. A set of keywords (from 1 to 500) and one location are assigned randomly to each document. Space is partitioned into  $225 \times 225$  cells. The documents' keywords and locations are uniformly distributed.

Each query contains 1 to 4 randomly generated keywords and one rectangle. Each query *round* consists of 100 queries. All three structures are disk-resident and the page size is fixed at 4KB. *MIR<sup>2</sup>-tree* and *CDIR-Tree* implementations are the same as in [3] and [9], respectively (e.g. signature length = 189 bytes,  $\beta = 0.1$ , number of clusters = 5, etc.). Experiments were run on a machine with an Intel Core2 Duo 3.16 GHz CPU and with 4GB main memory. Due to space limitation and since the results for DATASET1 and DATASET2 were very similar, we only report the results for DATASET2.

**Performance.** With the first set of experiments, we evaluate the impact of the number of keywords in each query  $|K_q|$  on query cost. In this set of experiments, we vary  $|K_q|$  from 1 to 4 while fixing  $k$  at 10 and  $\alpha$  at 0.5. For each method, we report the average

query cost in processing each round. The results are shown in Figures 6(a) and 6(b). For almost all the cases, SKIF significantly outperforms both MIR<sup>2</sup>-tree and CDIR-tree. While for all the approaches, the query cost increases as  $|K_q|$  grows, the growth rate for SKIF is very marginal. While the I/O costs of CDIR-tree and MIR<sup>2</sup>-tree increase by a factor of 15 and 8 respectively, SKIF's query cost barely doubles when the number of keywords grows from 1 to 4. Both CDIR-tree and MIR<sup>2</sup>-tree will perform even worse if  $|K_q|$  increase further. This is because with IR<sup>2</sup>-tree, by increasing the number of keywords, fewer documents will contain all the keywords and hence more documents need to be searched (this also increases the number of false hits). With CDIR-tree, when query contains more keywords, the textual relevance of the query with each node of CDIR-tree is very similar, which makes the textual relevance pruning less effective. Therefore, both approaches need to search larger and larger number of documents as  $|K_q|$  increases. On the other hand, SKIF only searches for those documents containing the query keywords and therefore required to be scored.

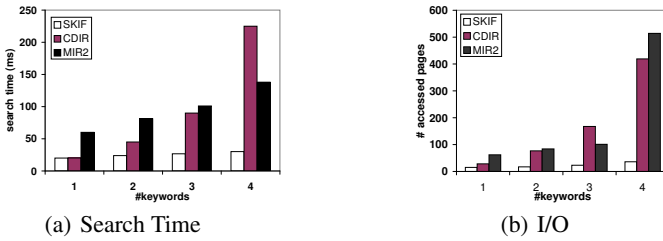


Fig. 6. Impact of  $|K_q|$  on query cost

With the second set of experiments, we evaluate the impact of the number of requested result  $k$  on the query performance. Again, we report the average query cost for each round.  $|K_q|$  is fixed at two and  $\alpha$  is fixed at 0.5 and  $k$  varies from 1 to 50. Figures 7(a) and 7(b) show the results for search time and number of page accesses, respectively. The first observation is that for SKIF, the query cost changes slightly as  $k$  increases. Since the average number of terms in the query as well as  $k$  are small, only few disk pages in the inverted lists of the few query terms are retrieved and processed. On the other hand, CDIR-Tree and MIR<sup>2</sup>-tree perform worse as  $k$  grows since they have to access and process more entities in their corresponding trees.

In the third set of experiments, we study the impact of the parameter  $\alpha$  on the performance of SKIF and CDIR-tree. As mentioned earlier,  $\alpha$  is the parameter that assigns relative weights to the textual and spatial relevances. We fix  $|K_q|$  at two and  $k$  at 10. Figures 8(a) and 8(b) show the results. The important observation is that the query cost for SKIF is weight independent while CDIR-tree performs very poorly when the spatial relevance is more important (large  $\alpha$ ). Since CDIR-tree takes into account document similarity, it performs well when the textual relevance is given higher importance and performs poorly when the spatial relevance is given higher importance. On the contrary, SKIF performs well for all the cases, since the query processing is the same for both keywords and space and is not affected by the relative weights.

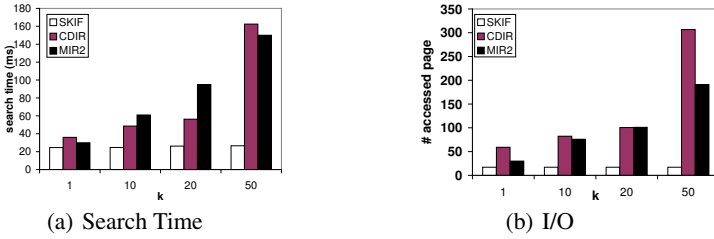


Fig. 7. Impact of  $k$  on query cost

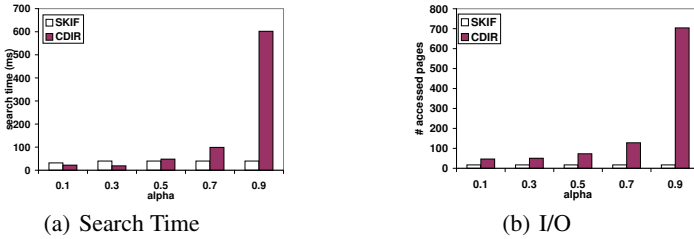


Fig. 8. Impact of  $\alpha$  on query cost

**Accuracy.** Our final set of experiments was conducted to evaluate the accuracy of our four proposed scoring approaches. Since spatial-keyword relevance ranking is new and no ground truth exists for our work, we conducted a user study to evaluate the effectiveness of our ranking methods. To conduct the user study, we utilized the user study in [15] (well-known paper in information retrieval) as our model and followed a similar procedure. We randomly selected 10 queries from our query set and found 10 volunteers. For each query, each volunteer was shown 6 result rankings, each one consisted of the top 10 results satisfying the query when the results were ranked with one of these approaches: *DSI*, *DSD*, *SSD*, *SSI*, *CDIR-tree* and *MIR<sup>2</sup>-tree*. Each volunteer was asked to select all documents which were "relevant" to the query, in their opinion. They were not told how many of the rankings were produced. We used *R-precision* [16] to evaluate the results of various rankings. R-precision is defined as follows. Let a document be considered as *relevant* if at least 6 of the 10 volunteers choose it as relevant for the query. Let *Rel* be a set that contains all such *relevant* documents and let  $|Rel|$  be the size of that set. Then, the R-precision of each list is the fraction of the top  $|Rel|$  documents that are deemed *relevant*. Hence, the higher the value of R-precision the more relevant the corresponding ranking. The R-precision of the six ranking approaches for each test query is shown in Table 2. We have also included the average R-precision for each ranking method. The first important observation is that for the majority of cases, our four proposed approaches generate results with R-precision equals to one, i.e., lists in which all the top  $|Rel|$  documents are *relevant*. The second observation is that the average R-precision for the rankings generated by our approaches is substantially higher than that of the other two rankings.

**Table 2.** R-precision of various rankings

Query	DSI	DSD	SSD	SSI	MIR <sup>2</sup> -tree	CDIR
1	1	1	1	1	1.00	0.67
2	1	1	1	1	1.00	1.00
3	1	1	1	0.95	1.00	1.00
4	1	1	0.8	0.8	0.00	0.10
5	0.9	0.7	0.9	0.8	0.00	0.20
6	1	1	1	1	0.88	0.88
7	1	1	1	1	0.00	0.38
8	1	1	1	1	1.00	1.00
9	1	1	1	1	1.00	0.80
10	1	1	1	1	1.00	1.00
Average	0.99	0.97	0.97	0.96	0.69	0.70

**Table 3.** Ranking preferred by users

Query	Preferred by Majority
1	DSI
2	DSI
3	DSI
4	DSD
5	DSI
6	DSD
7	SSD
8	MIR <sup>2</sup> -tree
9	MIR <sup>2</sup> -tree
10	DSI

Finally, Table 3 shows the rankings (actual order) preferred by the majority of the users. For nearly all the queries, a majority of the users preferred one of our proposed scoring methods. These results, further confirms the effectiveness of our proposed approaches.

## 7 Conclusions

In this paper we introduced the problem of ranking the spatial and textual features of web documents. We proposed new scoring methods to rank documents by seamlessly combining their spatial and textual features. We also proposed an efficient index structure which handles the spatial and textual features of data simultaneously and also supports the spatial-keyword relevance ranking. In particular we introduced SKIF and showed how it is used to search and rank the documents efficiently. We experimentally studied our methods, which proved its superior performance and accuracy.

**Acknowledgments.** This research has been funded in part by NSF grant CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

1. Zhou, Y., et al.: Hybrid index structures for location-based web search. In: CIKM (2005)
2. Hariharan, R., et al.: Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In: SSDBM (2007)
3. De Felipe, I., et al.: Keyword search on spatial databases. In: ICDE (2008)
4. Zobel, J., et al.: Inverted files for text search engines. ACM Comput. (2006)



5. Baeza-Yates, R., et al.: *Modern information retrieval*. Addison-Wesley, Reading (1999)
6. Chen, Y.: Efficient query processing in geographic web search engines. In: SIGMOD (2006)
7. McCurley, K.S., et al.: Geospatial mapping and Navigation of the Web. In: WWW (2001)
8. Salton, G., et al.: *Term-Weighting approaches in automatic text retrieval* (1988)
9. Cong, G.: Efficient retrieval of the top-k most relevant spatial web objects. In: PVLDB (2009)
10. Vaid, S., et al.: Spatio-textual indexing for geographical search on the web. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 218–235. Springer, Heidelberg (2005)
11. Amitay, E., et al.: Web-a-where: geotagging web content. In: SIGIR (2004)
12. Ding, J., et al.: Computing geographical scopes of web resources. In: VLDB (2000)
13. Gao, W., et al.: Geographically focused collaborative crawling. In: WWW (2006)
14. Zobel, J., et al.: Adding compression to a full-text retrieval system. *Sof. Prac. Exp.* (1995)
15. Haveliwala, T., et al.: Topic-sensitive PageRank. In: WWW (2002)
16. Manning, C., et al.: *Introduction to information retrieval*. Cambridge University Press, Cambridge (2008)

# Understanding the Human Genome: A Conceptual Modeling-Based Approach

(Extended Abstract)

Oscar Pastor

Centro de I+D en Métodos de Producción de Software (PROS)  
Universidad Politécnica de Valencia  
Camino de Vera s/n, 46022 Valencia, Spain  
opastor@dsic.upv.es

Nowadays, wide consensus exists about the importance of conceptual modelling in the definition of software systems that correctly fit the user's needs. In the middle of the MDD (Model-Driven Development) or MDA (Model-Driven Architecture) era, we can find a relevant set of widely spread modelling-based software production techniques, always based on well-founded concepts and methods that show that only having a precise system description, its corresponding software product can be properly understood. But it is interesting to see how these principles are not so commonly applied in current, challenging domains as Bioinformatics, where the management of tons of data is an increasing problematic issue, and where too often we can find serious problems in their manipulation that remember the data quality problems worked out for years within the Information Systems and Software Engineering communities.

This problem is often referred to as the chaos of genomic data, the main problem being that more and more are generated continuously, what makes the situation more and more complicated if we want to manage it adequately. With more than 1300 biological data sources reported to exist currently, an effective and efficient data searching strategy is a need for any activity related with the manipulation of genomic data. But the current situation makes very difficult –when not just impossible– to perform the data analysis that is required when all this huge amount of information has to be exploited. Even traditional database models that have been working successfully in the last decades are questioned when massive data analysis is needed and the volume of manipulated data is beyond the working capacity of the existing DBMS systems, normally relational-based.

In this scenario, where unstructured, low-quality data plays a prominent role, the goal of this keynote is to show how conceptual modeling techniques applied to human genome concepts can provide a working solution, by i) helping to understand and represent correctly the relevant concepts in a conceptual schema, and ii) facilitating a both rational and effective data exploitation strategy by assuring that the relevant information is properly understood from a conceptual perspective. The intention is to show that this it is the only way to enable effective management of the involved data. Only Conceptual Modeling techniques allow for providing a precise definition of relevant genomic concepts as basic as the concepts of Gene, Allele, Mutation, Splicing, Transcription Units, SNPs, Proteins... Unfortunately this view of the problem is

not –yet!- the normal practice in current Genomic based research, where relevant concepts are managed with a high dose of conceptual uncertainty, what generates serious problems in terms of having the required data quality to manipulate it correctly.

The keynote will develop the idea that current Bioinformatics practice should be much more Information Systems (IS)-based. It is obviously true that many so-called biological ontologies, data banks and very diverse sources of genomic information exist (just as an example we could cite HUGO, Gene Ontology, Entrez Gene, Human Gene Mutation Database, VEGA, DBSnp,...), but it is also dramatically true that all these data have not a precise conceptual schema behind them, and that the data are provided in numerous, disconnected databases, that conform a set of data silos where the lack of uniformly structured data affects many basic areas, especially in the biomedical research domain. In this domain, data rely heavily on integrating and interpreting data sets produced by different experimental methods at different levels of granularity, what makes conceptual models really needed to facilitate a systematic development of biological systems.

Under this generic perspective, the benefits of a Conceptual Modeling-Oriented Genome Systems Management will be introduced, explaining how to define and exploit a Conceptual Model of the Human Genome. What problems are to be solved will be discussed, together with the main procedural steps that must conform such a CM-based process: schema definition, data loading and maintenance from different biological data sources, and data exploitation in terms of linking adequately genotype and phenotype to better understand a very old question: why we are as we are.

Instead of using a bag of information, a Conceptual Schema will introduce a kind of well-structured “cupboard”, intended to assure that each piece of genomic information will be at the right place, what will make possible an efficient data exploitation, the right answers will be provided for the required questions through the achieved data integration, and the evolving nature of the genomic data will be dealt with at the right level: the conceptual one. The potential benefits of this approach will be also discussed, especially in the area of elaborating genomic reports, that is expected to open a new era in the domain of the personalized medicine.

After presenting a concrete proposal of a Conceptual Schema version for the Human Genome, the final keynote goal will be to justify its necessity, that will be based on the fact that its existence is strictly required...:

1 to enable and facilitate global research among the big set of various, distinct research groups that manipulate heterogeneous data, by fixing a conceptual gamut from which researchers can draw, in order to ensure that a 'standard dictionary of concepts' exists. This is to be achieved by:

- fixing the relevant concepts, their properties, behavior and relations, in the same way that a conceptualizing ontology would do.
- disambiguating existing biological concepts

2 to facilitate means of explicitly representing information so that:

- knowledge does not 'disappear' with it's creator
- it is clear what knowledge is present, and what is not

3 to enhance the pursued understanding of the human genome. Related to the previous comments, by looking at the selected genomic domain in terms of concepts, their

properties and relations, one disposes of a very powerful reasoning tool, useful at uncovering new relations, concepts and eventually fully understanding the given domain.

4 to drive an efficient and effective storage and processing policies for genomic data, conceptual-schema centric to assure that the well-known IS principles that enforces a data design of quality are properly considered. In that way the current manual methods applied in Bioinformatics that include tedious and repetitive tasks, with no explicit methods, prone to human errors and with a lot of required navigation through complex hyperlinks sequences, will be drastically improved.

The final objective of this work should be clear at the end of the keynote: if with Conceptual Models targeted at digital elements we have been able to improve Information Systems Development, with Conceptual Models targeted at life –as the Human Genome domain analysis makes possible- we could directly improve our living.

# Learning Semantic N-Ary Relations from Wikipedia

Marko Banek, Damir Jurić, and Zoran Skočir

University of Zagreb, Faculty of Electrical Engineering and Computing,  
Unska 3, HR-10000 Zagreb, Croatia

{marko.banek,damir.juric,zoran.skocir}@fer.hr

**Abstract.** Automated construction of ontologies from text corpora, which saves both time and human effort, is a principal condition for realizing the idea of the Semantic Web. However, the recently proposed automated techniques are still limited in the scope of context that can be captured. Moreover, the source corpora generally lack the consensus of ontology users regarding the understanding and interpretation of ontology concepts. In this paper we introduce an unsupervised method for learning domain n-ary relations from Wikipedia articles, thus harvesting the consensus reached by the largest world community engaged in collecting and classifying knowledge. Providing ontologies with n-ary relations instead of the standard binary relations built on the subject-verb-object paradigm results in preserving the initial context of time, space, cause, reason or quantity that otherwise would be lost irreversibly. Our preliminary experiments with a prototype software tool show highly satisfactory results when extracting ternary and quaternary relations, as well as the traditional binary ones.

## 1 Introduction

Ontologies have been designed as the core of the Semantic web. Early ontologies, constructed manually by domain experts, were of low usability due to their limited scope and to disagreements in understanding and interpreting their content, which arose between the ontology creators and the majority of potential users (who were not involved in the process of ontology creation or were not even able to propose some extensions). In the recent years, with a huge amount of available digital content, methodologies have been developed for automatically creating and populating ontologies from text sources. Most of them focus on class hierarchies and identifying instances, capturing only a minimum of class-to-class relations, such as *is-a*, *part-whole* or some relations with generic names such as *associated-with* or *related-to*, which do not represent a satisfactory semantic meaning.

Several existing ontology population techniques able to extract arbitrary semantic relations from text corpora [3,4,8,12] are focused exclusively on binary relations. Being noun-oriented in general, ontologies present binary relations (called properties in OWL [15]) as associations of the pivot domain class with

other classes. On the contrary, when humans express their thoughts by means of a natural language, they focus on verbs (at least, this is true for Indo-European languages, including English). Consider the following sentence: *John invited his friends to dinner on Friday*. The verb, *invite*, connects four nouns: *John*, *friends*, *dinner* and *Friday*. Certainly, the presented fourfold relation (as seen from the verb as the pivot) can be dissolved into many binary relations to populate ontologies in the standard manner. Recently, extensions to OWL have been proposed to include n-ary relations [14]. We believe that learning the original n-ary relations from the domain can lead to a significant improvement of ontology quality.

While the automatic construction of ontologies from text corpora provides a significant improvement in comparison to manual creation, there still remains an unsolved problem of selecting relevant sources, where there will be no problems with understanding and interpreting concepts. According to [7], ontologies are not just formal representations of a domain, but rather community contracts about such formal representations. Therefore, in order to be able to reflect the community consensus, the vast amount of Wikipedia entries should be reused as ontology components [7]. A Wikipedia entry (i.e. a titled article) may correspond to a class, an instance or a property.

In this paper we introduce an unsupervised method for learning domain n-ary relations from Wikipedia articles. By populating the ontologies with n-ary relations we achieve their higher expressivity and preserve the initial context that otherwise would be lost irreversibly. Another novelty of our approach is the fact that we extract only relations where the participating nouns correspond to Wikipedia titles. Hence, instead of having first to determine which nouns are relevant for the future ontology, which is necessary when relations are extracted from text corpora, we exploit the consensus of Wikipedia article authors regarding term relevance.

The paper is structured as follows. Related work is outlined in Sec. 2. The use of n-ary relations for discovering richer semantic context is explained in Sec. 3. The relation extraction process is elaborated in Sec. 4. The evaluation of the approach is illustrated in Sec. 5. In Sec. 6 conclusions are drawn and a list of future work tasks is presented.

## 2 Related Work

The initial efforts to extract relations from text, based on pattern matching [11], have worked well for a limited scope of relations which are known to hold in advance (such as *is-a* or *part-of*). The need of the ontology learning community to extract arbitrary relations has produced various methods that first discover verbs in text and then exploit syntactic dependencies between those verbs and other sentence constituents containing nouns [3,4]. In the next step the domain and the range of the relations must be generalized since the nouns in the text are mostly proper nouns or concepts at a level lower than the highest possible domain and/or range concept [4].

An unsupervised method for learning arbitrary semantic binary relations between ontological concepts in the molecular biology domain is presented in [3].

Sentences are parsed using a constituent syntactic parser and then searched in order to find the occurrences of the six relevant syntactic structures. A null hypothesis is stated implying that a particular pair of nouns and a particular verb forming the given syntactic structure do not occur together more frequently than expected at chance. Rejection of the null hypothesis by the subsequently applied chi-square test implies a new relation.

The unsupervised methodology presented in [12] assumes an incremental process, starting with the user's specification of the domain with a single term, and then extracting the relations containing the term, its hyponyms and hypernyms. Heuristic measures related to the occurrence of verbs in the source text are applied to single out the relations that are indeed relevant for the domain.

Meanwhile, the information extraction (IE) community has developed unsupervised methods to extract binary [1] or even n-ary relations [5] from web documents. However, those approaches use documents collected by search engines and apply a less restrictive approach to exploring syntactic dependencies (in comparison with the ontology learning community), which results in a very large portion of "dirty" relations unusable for ontology population.

### 3 N-Ary Relations in Text

Ontologies can be reused for multiple purposes and by a variety of users if they are able to capture precisely the context of the domain i.e. the context of the sentences that serve as their source. In English, the two main parts of a sentence are the subject and the predicate. The predicate must contain a verb, and, depending on the verb, may also contain objects (direct, indirect and prepositional), predicatives and/or adverbials. Relations capture the association between the verb and the nouns in the sentence, which are either classes or individuals (class instances). Binary relations describe an association between the subject noun, the verb and a noun in the predicate. On the other hand, n-ary relations give much richer and subtler definition of the context: in the majority of cases they include subject, object and prepositional phrases containing nouns that describe the context of time, space, etc.

We will show the plenty of context that can be inferred from a single sentence (taken from the Wikipedia article about Alexander the Great): *Born in Pella in 356 BC, Alexander succeeded his father Philip II of Macedon to the throne in 336 BC, after the King was assassinated, and died thirteen years later at the age of 32.* The main verb, *succeeded*, describes an event of *succession* (WordNet [6] enables finding derivationally related nouns of input verbs). A ternary relation (Fig. 1) associates the event to Alexander (subject), Philip (object) and year 336 BC (prepositional phrase). Particular associations obtain their names from the verb by adding suffices corresponding to the part of the sentence (subject: *is\_successor*, object: *succeeded\_by*). Possible inverse associations (e.g. *precede-succeed*) can be named by retrieving verb antonyms from WordNet [6].

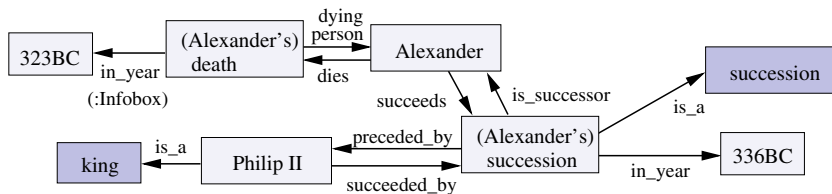


Fig. 1. Ontology structure extracted from Wikipedia article

## 4 Processing Wikipedia

As mentioned in Sec. 1, we create ontologies with classes and instances corresponding to Wikipedia titles. Hence, when we extract relations from sentences, all nouns declared as relation participants must be annotated as Wikipedia titles.

The source for the entire process are text files formatted in a special kind of markup. We shortly outline its features that are relevant for the relation extraction process. Each article has a title that exactly corresponds to its URL. A sentence from the article titled NAPOLEON I OF FRANCE is given in Fig. 2. Links

Six weeks later, on the first anniversary of his coronation, Napoleon defeated Austria and [[Russia]] at [[Battle of Austerlitz | Austerlitz]], ending the third coalition.

Fig. 2. A portion of Wikipedia article Napoleon I of France

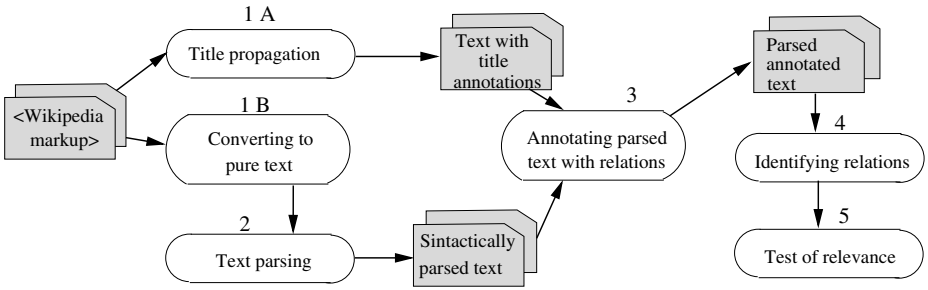
to other articles are given in double square brackets. The displayed text seeding the link may be different from the link title, which is denoted by a vertical bar within the double square brackets ([[Battle of Austerlitz | Austerlitz]]) in Fig. 2 means a link to the article BATTLE OF AUSTERLITZ, which is simply displayed as Austerlitz). The lack of a vertical bar means that the displayed text is identical to the title of the linked article (e.g. [[Russia]] in Fig. 2). It is also very important to note that links to other articles appear only once or several times within the text, although the denoted term itself may appear many times (there is a link to RUSSIA, but no link to AUSTRIA, since the latter already appears earlier in the text).

### 4.1 Processing Algorithm

Our algorithm for learning n-ary relations from Wikipedia articles consists of several activities (Fig. 3), which will be described in this subsection.

*Title Propagation.* In order to extract relations from Wikipedia text, we have to identify all occurrences of word tokens that are unambiguously related to the title of a Wikipedia article. Certainly, each link (i.e. double square brackets) points to a title. However, one of the conventions on writing Wikipedia articles is to create a link only for the first occurrence of titled term, while giving no

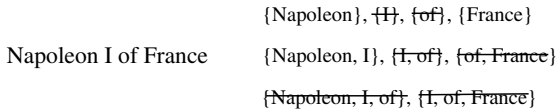




**Fig. 3.** The relation extraction algorithm

markup emphasis for any further appearances. Hence, for each title, which is either a link indicated by double square brackets or the title of the article itself, we try to find its other occurrences within the article. We call this procedure title propagation. The output of title propagation for the sentence in Fig. 2 is the following (capped text denotes titles): *Six weeks later, on the first anniversary of his coronation, NAPOLEON I OF FRANCE defeated AUSTRIA and RUSSIA at BATTLE OF AUSTERLITZ, ending the third coalition.*

For each title consisting of  $N$  multiple words we produce all possible subsets sized from 1 to  $N - 1$  where each word is allowed to follow only its immediate predecessor in the original title. Subsets that do not contain at least one noun or those that start or end with a preposition or a conjunction are eliminated (Fig. 4). Some subsets may be derived from more than one title present in the article. For instance, titles LOUIS XVI OF FRANCE and LOUIS XVIII OF FRANCE both produce the subset {Louis}. Consequently, we will not associate {Louis} with any title. The shorter the subset, the more ambiguity is to be expected. Therefore, we start the process of relating text tokens to titles with the longest subsets (all examples of {Louis, XVI} will be associated to LOUIS XVI OF FRANCE before even starting with the ambiguous subset {Louis}).

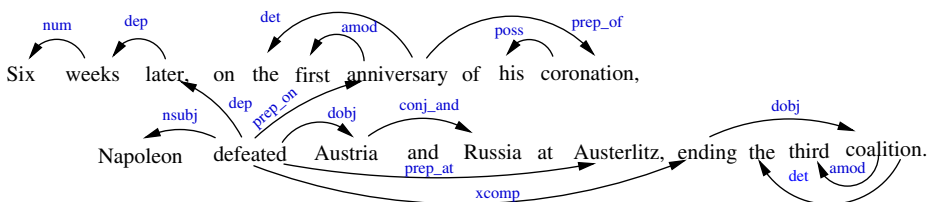


**Fig. 4.** Creating subsets during the title propagation process

We do perform title association if the ambiguous subset corresponds to the titular of the article. All occurrences of {Napoleon} in the NAPOLEON I OF FRANCE article are associated with Napoleon I, although one actually refers to his nephew, NAPOLEON III OF FRANCE. Hence, there may be false title associations, but we argue that producing several false associations is a reasonable price in comparison with losing many relations having the article titular as participant.

A related problem, linking portions of unstructured text to Wikipedia articles, is described in [10]. However, the approaches cannot be compared since their goals require substantially different behavior in case of term ambiguity.

*Parsing Sentences.* For each input sentence the Stanford dependency parser [9] produces its typed dependency graph, which describes all sentence relationships uniformly as typed relations. Parsing the sentence in Fig. 2 produces the output shown in Fig. 5. There are 55 relation types in total, but we ignore all clause relations (e.g. clausal complement, *xcomp*) and general dependencies (*dep*). The most important relations of our interest are active and passive subject (*nsubj*, *nsubjpass*), passive verb agent (*agent*), direct, indirect or prepositional object (*dobj*, *iobj*, *pobj*) and prepositional modifier (*prep*). Graphs containing conjunction (*conj*) may be split into several independent subgraphs.



**Fig. 5.** The typed dependency graph produced by the Stanford dependency parser

*Identifying Relations and Relevance Test.* An n-ary relation stemming from a verb  $v$  is a set of triples  $\{(C, v, r)\}$ , where  $C$  is an ontology concept (in our case a Wikipedia title), and  $r$  is a relation type. All triples share the same verb  $v$  and none of them share the same relation type. Considering the output of title propagation parsing, the sentence shown in Fig. 5, produces two ternary relations:  $\{(NAPOLEON\ I\ OF\ FRANCE, defeat, nsubj), (AUSTRIA, defeat, dobj), (BATTLE\ OF\ AUSTERLITZ, defeat, prep\_at)\}$  and  $\{(NAPOLEON\ I\ OF\ FRANCE, defeat, nsubj), (RUSSIA, defeat, dobj), (BATTLE\ OF\ AUSTERLITZ, defeat, prep\_at)\}$ . We use two different methods to identify relevant relations among a much larger number of the automatically identified ones: the chi-square test and a heuristic probabilistic formula. Wikipedia titles within the relevant relations can further be turned into WordNet concepts by following the approach explained in [13].

## 5 Evaluation

We processed 171 Wikipedia articles describing rulers or military commanders (46 articles), writers (71), philosophers (32) and battles (22). In total, this corpus contained 32809 sentences with 521334 words, which also included 24124 links (4,6% of the total word count). After title propagation, the number of title references increased to 65526 (12,5% of the total word count). They pointed to 15311 different titles, meaning that each of the references appeared 4.3 times in average. We automatically identified binary, ternary and quaternary relations

**Table 1.** Results of the relevance test

N	II	CHI-SQUARE				AE			
		RR	ST	RR/II	P	RR	ST	RR/II	P
2	1587	99	77	0,062	0,778	981	546	0,618	0,556
3	142	0	0	0	-	100	78	0,704	0,780
4	4	0	0	0	-	3	3	0,75	1

and tested their relevance by using two different approaches: the chi-square test and a heuristic probabilistic formula.

The chi-square test is applied in a fashion similar to [3]. Let  $A$  be an ordered tuple of Wikipedia titles participating in the  $n$ -ary relation. Let  $B$  be a pair, consisting of the verb and an ordered tuple of relation types (the order of relation types corresponds to the order of titles in  $A$ ). The null hypothesis states that  $A$  and  $B$  do not co-occur more frequently than by chance. We perform the test using the log-likelihood formula ( $G^2$ ). Relevant relations produced as the output of the algorithm are those for which the null hypothesis is rejected.

The other test applies the heuristic *above expectation* formula originally devised in [8]. The formula presented in Eq. II measures the conditional frequency of all relation concepts (given the verb and the relation types) compared to the conditional frequencies expected when each concept (Wikipedia title in our case) is related to the verb independently of other concepts:

$$AE(C_1 \wedge \dots \wedge C_n \mid v \wedge r_1 \wedge \dots \wedge r_n) = \frac{P(C_1 \wedge \dots \wedge C_n \mid v \wedge r_1 \wedge \dots \wedge r_n)}{P(C_1 \mid v \wedge r_1) \cdot \dots \cdot P(C_n \mid v \wedge r_n)} \quad (1)$$

The presented equation is our extension of the original formula [8], which could not distinguish between relation types.

The results of the evaluation are shown in Table 1 ( $II$  - no. of initially identified relations;  $RR$  - no. of relevant relations;  $ST$  - no. of semantically true i.e. correct relations;  $P$  - precision i.e.  $ST/RR$ ). Due to data sparseness, the chi-square test cannot confirm the relevance of any ternary or quaternary relation. However, the AE measure (with a threshold empirically set to 5) filters about 70% of the automatically identified relations with a satisfactory precision of 78%. Although the number of relevant binary relations obtained by the chi-square test is smaller than expected (6.2%), we still prefer chi-square to the AE measure due to higher precision (77,8% in comparison with 55,6%). Thus, we suggest using chi-square test to determine the relevant relations, while the AE measure can serve as its replacement in case of sparse data ( $n$ -ary relations with  $N > 2$ ).

## 6 Conclusion

In this paper we presented an unsupervised approach for learning semantic  $n$ -ary relations from Wikipedia. We restrict the nouns participating in relations to those corresponding to Wikipedia article titles, thus harvesting the consensus reached by the authors of Wikipedia - the largest world community engaged in

collecting and classifying knowledge. The process of discovering all occurrences of article titles within Wikipedia text, called title propagation, is a key step in identifying the relations. Due to data sparseness, the relevance of the identified relations with more than two members cannot be determined by the chi-square test, otherwise used for binary relations. Instead, satisfactory results are achieved by applying our extension of the "above expectation" formula.

Our future work will address the process of naming the extracted relations. Particular focus will be given to discovering verb-preposition patterns that imply the class of the noun in a prepositional phrase: place, time, etc. We will also exploit Wikipedia structures such as infoboxes for acquiring additional semantics.

## References

1. Banko, M., Etzioni, O.: The Tradeoffs between Open and Traditional Relation Extraction. Proc. Assoc. Comp. Linguistic (2008)
2. Buitelaar, P., Cimiano, P. (eds.): Ontology Learning and Population: Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text. IOS Press, Amsterdam (2008)
3. Ciaramita, M., Gangemi, A., Ratsch, E., Šarić, J., Rojas, I.: Unsupervised Learning of Semantic Relations for Molecular Biology Ontologies. In: [2]
4. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer, Heidelberg (2006)
5. Eichler, K., Hensen, H., Neumann, G.: Unsupervised Language Extraction from Web Documents. In: Proc. LREC 2008, pp. 1674–1679 (2008)
6. Fellbaum, C. (ed.): WordNet. An Electronic Lexical Database. MIT Press, Cambridge (1998)
7. Hepp, M., Bachlechner, D., Siorpaes, K.: Harvesting Wiki Consensus - Using Wikipedia Entries as Ontology Elements. In: Workshop Semantic Wikis (2006)
8. Kavalec, M., Svátek, V.: A study on automated relation labelling in ontology learning. In: Buitelaar, P., Cimiano, P., Magnini, B. (eds.) Ontology learning from text: methods, evaluation and applications. IOS Press, Amsterdam (2005)
9. de Marneffe, C.-M., MacCartney, B., Manning, C.D.: Generating Typed Dependency Parses from Phrase Structure Parses. In: Proc. LREC 2006, pp. 449–454 (2006)
10. Milne, D.N., Witten, I.H.: Learning to link with Wikipedia. In: CIKM, pp. 509–518 (2008)
11. Pantel, P., Pennacchiotti, M.: Automatically Harvesting and Ontologizing Semantic Relations. In: [2]
12. Sánchez, D., Moreno, A.: Learning non-taxonomic relationships from web documents for domain ontology construction. Data Knowl. Eng. 64(3), 600–623 (2008)
13. Suchanek, F.M., Kasneci, M., Weikum, G.: Yago: a Core of Semantic Knowledge. In: Proc. WWW 2007, pp. 697–706 (2007)
14. W3C: Defining N-ary Relations on the Semantic Web. W3C Working Group Note (2006), <http://www.w3.org/TR/2006/NOTE-swpb-n-aryRelations-20060412/>
15. W3C: OWL Web Ontology Language. W3C Recommendation (2004), <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

# RIF Centered Rule Interchange in the Semantic Web

Xing Wang<sup>1</sup>, Z.M. Ma<sup>1</sup>, Fu Zhang<sup>1</sup>, and Li Yan<sup>2</sup>

<sup>1</sup> College of Information Science and Engineering, Northeastern University, China

<sup>2</sup> School of Software, Northeastern University, China

mazongmin@ise.neu.edu.cn

**Abstract.** In the Semantic Web, rule interchange has gained considerable attention. To be a general rule interchange format, RIF (Rule Interchange Format) should first support rule interchange with three important evolving rule languages SWRL (Semantic Web Rule Language), RuleML (Rule Markup Language) and R2ML (REVERSE Rule Markup Language). In the paper, we propose a rule metamodel. Based on the metamodel, we construct RIA (Rule Interchange Architecture), which supports bidirectional rule interchange between RIF and SWRL, RuleML and R2ML, and also between these four languages' metamodels and XML syntaxes. Based on RIA, we design and implement a rule interchange system RIA 1.0.

**Keywords:** Semantic Web; rule interchange; RIF; metamodel; RIA; RIA 1.0.

## 1 Introduction

The Semantic Web [1] has gained considerable attention in recent years. One of the declared goals of the Semantic Web is to represent knowledge which is pervaded on the Web. As an important representation, rules become a mainstream topic of the Semantic Web. With the development of rules, Semantic Web rule languages including SWRL (Semantic Web Rule Language) [2] and rule systems based on these languages come into being. It is unavoidable that these systems communicate with each other (i.e., interchange rules). And thus problems in sharing rules arise. Because of heterogeneity of different languages, creating a generally accepted interchange format is by no means a trivial task [3]. RuleML (Rule Markup Language) [4], R2ML (REVERSE Rule Markup Language) [5] and RIF (Rule Interchange Format) [6] are proposed to markup and interchange different rules. Among others, RIF has become a candidate recommendation of the W3C (WWW Consortium), whose goal is to construct a Web standard for interchanging rules between different rule languages.

Rule interchange based on metamodels and centered on a language (called central language) becomes an important trend. Metamodels can represent implicit relations between information. Rule interchange based on metamodels guarantees the correctness of syntaxes. Rule interchange centered on a central language reduces the number of mappings, and leads to a high transformation efficiency. A model transformation to share rules between SWRL and R2ML is proposed in [7], and a transformation based on metamodels between SWRL and OCL is discussed in [8]. G. Wagner implements a translator [9], which supports transformations between R2ML

and some others languages. The transformations above are all based on R2ML, and R2ML is the de facto central language.

So far, there is no published literature to investigate RIF metamodels and metamodel mappings between RIF and other rule languages. And there are no rule interchange structures and interchange systems centered on RIF. Rule interchange centered on RIF is the mainstream direction of rule interchange. Before completely fulfilling rule interchange with other rule languages, it is necessary for RIF to support rule interchange with the common languages SWRL, RuleML, and R2ML. In the paper, we visually model RIF, and propose a rule metamodel. Furthermore, we propose RIA (Rule Interchange Architecture), a rule interchange architecture centered on RIF, which supports abstract-concrete and abstract-abstract transformations. In addition, we design and implement RIA 1.0, a prototype system of RIA.

## 2 RIF Rule Metamodel

RIF aims at developing a Web standard for exchanging rules between different rule languages. It is an extensible framework for rule-based languages, called RIF dialects, which include specifications of the syntax, semantics and XML serialization. RIF-BLD (RIF Basic Logic Dialect) [10] is the first dialect of RIF dialects. It corresponds to the languages of definite Horn rules with equality and standard first-order semantics, and it also has a number of extensions to support features including objects and frames as in F-logic, IRIs (Internationalized Resource Identifiers), and XML Schema datatypes. In the rest of the paper, RIF stands for RIF-BLD, unless stated otherwise.

RIF is defined through a presentation syntax which is defined in “mathematical English” (a special form of English for communicating mathematical definitions, examples, etc) and XML syntax, and most care is spent on the formal semantics. From the respective of modelling, RIF does not have metamodel to describe features of rules. Metamodels can simplify conceptual modelling, decrease syntactic and semantic errors, and increase readability. Therefore, it’s necessary for RIF to define its own rule metamodel.

In RIF, there are two important terms, i.e., *term* and *formula*, which are the most fundamental elements of RIF dialects. *Term* includes constants, variables, positional terms, terms with named arguments, equality, membership, frame, and external terms, where constants and variables are called *simple terms*, and simple terms, positional terms, terms with named arguments and external terms are called *base terms*. *Formula* consists of atomic formulas, condition formulas, rule implications, universal facts, groups and documents. In all the above concepts (for a full account of them, the readers may refer to the literature [10]), rule implication is the core element.

A rule implication (as shown in Fig. 1) consists of two parts: antecedent and consequent, which are also called rule body and rule head, respectively. The relations between rule and its two parts, between antecedent and its atoms, and between consequent and its atoms can be represented by aggregation link. The reason is that the same antecedent or consequent may appear in several rules, and the same atom is allowed in different antecedents or consequents. An antecedent is a combination of disjunction, existentials and atoms, while a consequent is a combination of

conjunction and atoms, i.e., an antecedent is a set of conjunction, disjunction and existentials of atoms, while a consequent is an atomic formula or a conjunction of atoms. Both antecedent and consequent can possibly be empty. A rule means that if all atoms of the antecedent hold, then the consequent holds. An empty antecedent is treated as trivially true, whereas an empty consequent is treated as trivially false.

Here, atoms mean atomic formulas and simple terms. Atomic formulas consist of positional terms, named-argument terms, frames and external terms. In essence, these atomic formulas are classes, properties, datatypes and builtIns (built-in predicates and built-in functions), where properties can be n-ary, and datatypes and builtIns are introduced from the RIF-DTB (RIF Datatypes and Built-Ins) [11] document. Simple terms are variables and constants. Variables have two types, i.e., individual variables and data variables (RIF does not explicitly distinguish the types of variables), while constants are individuals or data values. Data variables and data values are also imported from the RIF-DTB document.

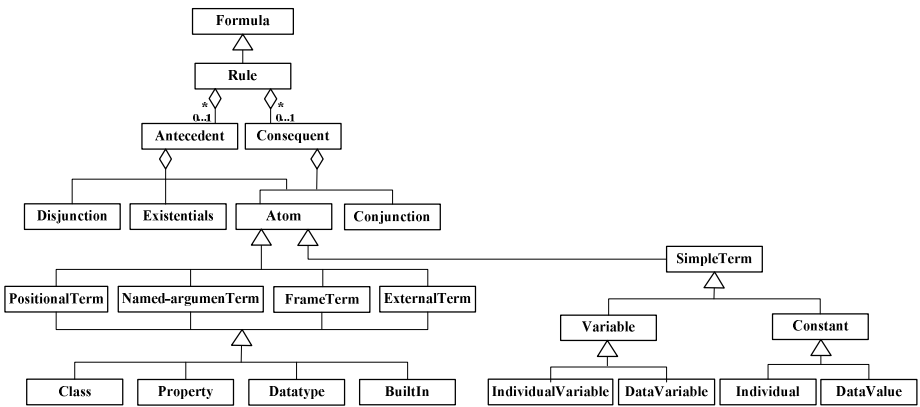


Fig. 1. RIF rule metamodel

### 3 RIA

As can be seen from Fig. 2, rule interchange architecture RIA is centered on RIF. It supports rule transformations between RIF and SWRL, R2ML and RuleML. Pairwise transformations among the four languages can also be supported. RIA also supports rule interchange between languages which are centered or based on SWRL, R2ML and RuleML. Among others, all the transformations are bidirectional and they are based on metamodels of the languages to be transformed, rather than based on their XML syntaxes. Another advantage of using metamodels is that metamodels are context-sensitive, and they can represent implicit information between elements, whereas XML syntax cannot. Because of the extensibility of RIF dialects, our architecture is also extensible. With the enhancement of expressiveness of RIF, RIA will be capable of transforming more rule languages, and its goal, implementing rule interchange with other rule languages, will come true at last.

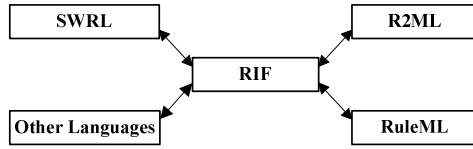


Fig. 2. The structure of RIA

RIA is composed of two parts. The first part depicts mappings between the four languages’ abstract syntaxes and concrete syntaxes, while the second part describes three pairs of transformations based on abstract syntaxes, i.e., transformations between RIF and SWRL, between RIF and R2ML, and between RIF and RuleML.

**3.1 Bridging Metamodels and XML Syntaxes of RIF, SWRL, R2ML and RuleML**

The rule metamodel of RIF has been constructed in Section 2, and metamodels of SWRL, R2ML and RuleML [12] [5] [13] already exist, both of which are in the MOF (Meta Object Facility) technical space, but there is a gap between abstract syntaxes and concrete syntaxes. Therefore, it is necessary to connect MOF technical space and XML technical space, i.e., constructing the bridge between metamodels and XML syntaxes of RIF, SWRL, RuleML and R2ML.

The core of RIA is model transformation. Fortunately, the OMG (Object Management Group) has adopted MOF2 QVT (Query View Transformation) specification to address this issue. We employ ATL (ATLAS Transformation Language) as the main language for model transformations. One of the advantages of ATL is that it has tools to inject or extract XML rules into/from the MOF representation. ATL can play an important role of bridging between one language’s abstract syntax and concrete syntax, and between abstract syntaxes of different languages (in the next subsection). Let us take RIF as an example to bridge XML and MOF technical spaces.

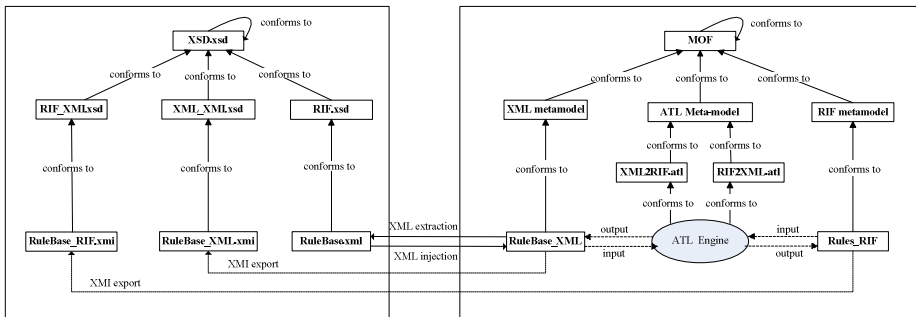


Fig. 3. Transformation between RIF concrete syntax and abstract syntax

As shown in Fig. 3, our transformation is bidirectional. The steps of the transformation are as follows. Firstly, we use XML injector, an important part of ATL



and also an important feature of ATL, to transform RIF XML document into models which conform to MOF-based XML metamodel that is used to define XML. After injecting RIF XML rules into RuleBase\_XML, the transformed models can be dealt with like other MOF compliant models. And then, such XML models can be represented in the form of XML XMI (i.e., XML Metadata Interchange, which is a standard of the OMG to define mappings of MOF compliant models onto XML documents). Based on the above, we begin the transformation between Rules\_XML (i.e., XML models) and Rules\_RIF (i.e., RIF models). The most important thing is to construct two ATL transformation documents, i.e., XML2RIF.atl and RIF2XML.atl (because of the unidirectivity of ATL language, we have to construct two inverted ATL documents to implement the mappings between models of RIF and XML). These two ATL documents are the bridge between the RIF XML-style concrete syntax and RIF MOF compliant abstract syntax. Fig. 3 is a three-layer structure (generally, the M0 layer is omitted), and the transformations are all located in the model layer (i.e., M1 layer in the MOF architecture). All the input and output models should be compliant to their respective metamodels in the middle layer. In this way, we can guarantee the validity of all RIF XML-style rules through the RIF metamodel. Because of the two-way feature of our transformations, we can transform RIF models into XML models, that later can be exported into RIF XML-style concrete syntax, which is important when exchanging the other three languages (SWRL, R2ML and RuleML) into the RIF XML-style concrete syntax. Once we have transformed RIF rules into its model representation, we can also export RIF rules in the XMI format, and thus we can share RIF rules with any MOF compliant structure. This way, RIF extends its range of influence, and this further establishes the central position of RIF in rule interchange.

We have introduced the transformations between the abstract syntax and concrete syntax of RIF above. For the mappings between abstract syntaxes and concrete syntaxes of SWRL, R2ML and RuleML, the key point is to construct ATL transformation documents, i.e., XML2RDM.atl, RDM2XML.atl, XML2R2ML.atl, R2ML2XML.atl, XML2RuleML.atl and RuleML2XML.atl (here, RuleML means the metamodel of RuleML), which are also constructed via ATL language. Because of similarity of transformation processes, we omit the abstract-to-concrete and concrete-to-abstract bidirectional transformations of SWRL, R2ML and RuleML.

### 3.2 Metamodel Transformations between RIF and SWRL, R2ML and RuleML

Through analyzing rule definition metamodel RDM [12] (which is the metamodel of SWRL), R2ML metamodel [5] and RuleML metamodel [13], we can see that the common character these three languages possess is that their abstract syntaxes (i.e., metamodels) are all MOF compliant, which is exactly the reason that all the four languages can exchange rules through our architecture. It is these MOF-based metamodels that constitute the core of the transformations in RIA. This common character also does good to validity checking of the four languages' concrete syntaxes.

In the following, let's take the transformation between RIF and SWRL as an example to show abstract-abstract transformations. This kind of transformation is based on the conceptual mappings between RIF metamodel and metamodels of the other three languages. The transformations between these two types of metamodels

are defined as a sequence of rules in ATL, where ATL constructors, such as matched rules, lazy rules and others are frequently used.

The transformation of the abstract syntaxes between RIF and SWRL is shown in Fig. 4. The RIF metamodel defined in Section 2 is the foundation of this transformation. RIF models will be transformed into RDM models. And the transformations are all located in the model layer. All the input and output models should be compliant to their respective metamodels in the middle layer. As with the transformation in Fig. 3, we also use ATL as the primary language for model transformation. Because unidirectivity of ATL, two ATL transformation documents, RIF2RDM.atl and RDM2RIF.atl, should be constructed. Because of the two-way feature of our transformations, we can also transform RDM models into RIF models. And taking advantage of the transformation in Fig. 3, we can obtain the XML models of RIF models or RDM models, and then obtain XML-style concrete syntaxes of RIF or SWRL. The validity of XML syntaxes can be guaranteed by the abstract syntaxes.

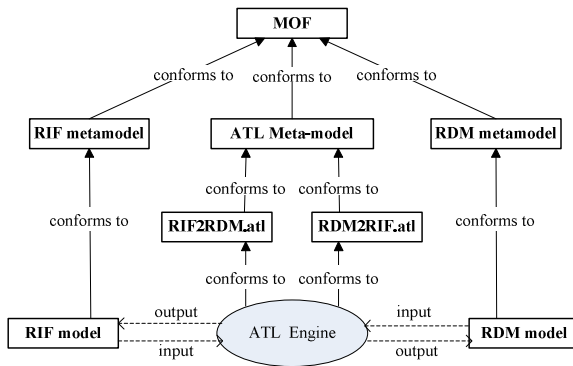


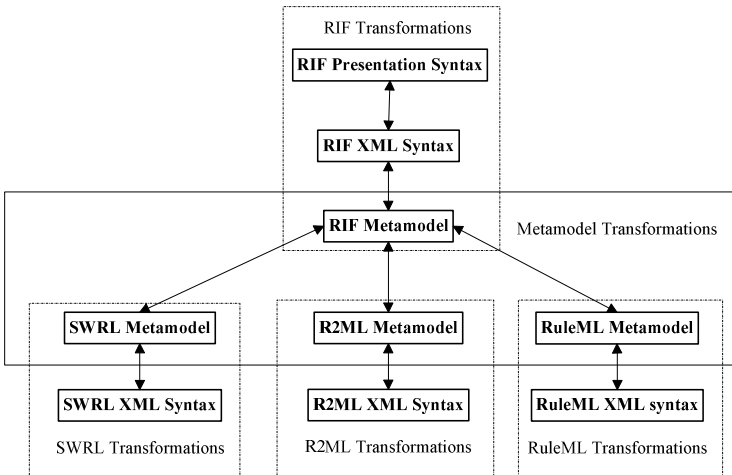
Fig. 4. Metamodel transformation between RIF and SWRL

In order to share rules between SWRL and RIF, we define mappings between constructors and axioms of SWRL and RIF on the level of their abstract syntaxes. Every SWRL rule is mapped to a RIF universal rule, and there will be a rule implication in it. In Table 1, we show the mappings between SWRL and RIF atoms and axioms in detail. In all mappings shown in the table, C represents classes, P denotes properties, an expression  $\chi_{\text{RIF}}$  is a transformation function (i.e.,  $\chi_{\text{RIF}}(\text{SWRL atoms}) = \text{RIF terms}$ , where RIF terms include positional terms, arguments-named terms and frame), and t is a variable. The semantics of SWRL rules which can be transformed into RIF rules in these mappings is preserved.

We have described metamodel transformations and mappings of atoms and axioms between SWRL and RIF. For transformations between R2ML and RIF, and between RuleML and RIF, the mapping processes can be obtained analogously. The most important thing is still to accurately construct ATL transformation documents, i.e., R2ML2RIF.atl and RIF2R2ML.atl, and RuleML2RIF and RIF2RuleML. For saving space, we omit these two transformations.

**Table 1.** Mappings of atoms and axioms between SWRL and RIF

SWRL expression	RIF expression
ClassAtom(classID,t)	t##classID(*membership axioms*)
ClassAtom(Unionof(C <sub>1</sub> ,C <sub>2</sub> ),t)	Or( $\chi_{RIF}(\text{ClassAtom}(C_1,t)),\chi_{RIF}(\text{ClassAtom}(C_2,t))$ )
ClassAtom(Intersectionof(C <sub>1</sub> ,C <sub>2</sub> ),t)	And( $\chi_{RIF}(\text{ClassAtom}(C_1,t)),\chi_{RIF}(\text{ClassAtom}(C_2,t))$ )
ClassAtom(ObjectRestriction(objPropID, allValuesFrom(C)),t)	UniversalRule(x,Implication( $\chi_{RIF}(\text{ObjectRestriction}(objPropID,t,x)),\chi_{RIF}(\text{ClassAtom}(C,t))$ )
ClassAtom(ObjectRestriction(objPropID, someValuesFrom(C)),t)	ExistentialRule(x,And( $\chi_{RIF}(\text{ObjectRestriction}(objPropID,t,x)),\chi_{RIF}(\text{ClassAtom}(C,t))$ )
DatarangAtom(DatatypeID,t)	t##DatatypeID(membershipaxioms)
DatarangeAtom(DataRestriction(dataPropID, allValuesFrom(DatatypeID)),t)	UniversalRule(x,Implication( $\chi_{RIF}(\text{DataRestriction}(dataPropID,t,x)),\chi_{RIF}(\text{DatarangeAtom}(datatypeID,t))$ )
DatarangeAtom(DataRestriction(dataPropID, someValuesFrom(DatatypeID)),t)	UniversalRule(x,And( $\chi_{RIF}(\text{DataRestriction}(dataPropID,t,x)),\chi_{RIF}(\text{DatarangeAtom}(datatypeID,t))$ )
IndividualvaluedPropertyAtom(objectID <sub>1</sub> , objectID <sub>2</sub> )	$\chi_{RIF}(\text{IndividualvaluedPropertyAtom}(\text{IndividualvaluedPropertyID},\text{objectID}_1,\text{objectID}_2))$
DatavaluedPropertyAtom(objectID <sub>1</sub> ,objectID <sub>2</sub> )	$\chi_{RIF}(\text{DatavaluedPropertyAtom}(\text{DatavaluedPropertyID},\text{objectID},\text{datavalue}))$
SubClassof(C <sub>1</sub> ,C <sub>2</sub> )	C <sub>1</sub> #C <sub>2</sub> (*Subclass axiom*)
EquivalentClasses(C <sub>1</sub> ,C <sub>2</sub> ),i.e.,Conjunction( SubClassOf(C <sub>1</sub> ,C <sub>2</sub> ),SubClassOf(C <sub>2</sub> ,C <sub>1</sub> ))	C <sub>1</sub> =C <sub>2</sub> (*Equality axiom*)
SubPropertyof(P <sub>1</sub> ,P <sub>2</sub> )	P <sub>1</sub> #P <sub>2</sub> (*Subclass axiom*)
EquivalentProperties(P <sub>1</sub> ,P <sub>2</sub> ),i.e.,Conjunction( SubPropertyof(P <sub>1</sub> ,P <sub>2</sub> ),SubPropertyof(P <sub>2</sub> ,P <sub>1</sub> ))	P <sub>1</sub> =P <sub>2</sub> (*Equality axiom*)



**Fig. 5.** Functions of RIA 1.0

## 4 RIA 1.0

RIA 1.0 has implemented two parts of RIA and a bidirectional transformation between RIF presentation syntax and RIF XML syntax. The implementation system

of RIA 1.0 is based on Java programming language and implemented on the Eclipse platform. All the functions of RIA 1.0 are shown in Fig. 5. There are five parts in this system: (1) RIF transformations: between presentation syntax and XML syntax, and between XML syntax and metamodel; (2) SWRL transformations: between metamodel and XML syntax; (3) R2ML transformations: between metamodel and XML syntax; (4) RuleML transformations: between metamodel and XML syntax; (5) metamodel transformations: between RIF and SWRL, R2ML and RuleML. The fifth part is the core of the whole system. It bridges the other four parts.

## 5 Conclusions

We propose a rule metamodel of RIF, and deeply analyze the structure of RIF rules. And we construct a rule interchange architecture RIA, which supports bidirectional rule interchange between RIF and SWRL, R2ML and RuleML. Based on RIA, we design a rule interchange system RIA 1.0, which implements a bidirectional transformation between RIF's presentation syntax and XML syntax, besides the functions of RIA.

In future, we will extend RIA to support production rules and nonmonotonic rules. In order that fuzzy rules representing imprecise and uncertain knowledge in the Semantic Web can be interchanged in RIA, we will fuzzify RIA.

**Acknowledgements.** This work is supported by the National Natural Science Foundation of China (60873010) and the Fundamental Research Funds for the Central Universities (N090504005 & N090604012).

## References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Journal of the Scientific American* 284(5), 34–43 (2001)
2. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A Semantic Web Rule Language | Combining OWL and RuleML, <http://www.w3.org/Submission/SWRL/>
3. Boley, H., Kifer, M., Patranjan, P.L., Polleres, A.: Rule Interchange on the Web. In: Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.-L., Tolksdorf, R. (eds.) *Reasoning Web 2007*. LNCS, vol. 4636, pp. 269–309. Springer, Heidelberg (2007)
4. Boley, H., Grosz, B., Tabet, S.: RuleML Tutorial, <http://ruleml.org/papers/tutorial-ruleml-20050513.html>
5. Wagner, G., Giurca, A., Lukichev, S.: A Usable Interchange Format for Rich Syntax Rules Integrating OCL, RuleML and SWRL. In: *Reasoning on the Web 2006* (2006)
6. Rule Interchange Format Working Group Charter, <http://www.w3.org/2005/rules/wg/charter.html#w3c-xml>
7. Milanović, M., Gašević, D., Guirca, A., Wagner, G., Lukichev, S., Devdžić, V.: Model Transformations to Share Rules between SWRL and R2ML. In: *3rd International Workshop on Semantic Web Enabled Software Engineering at 4th European Semantic Web Conference, Innsbruck, Austria* (2007)

8. Milanović, M., Gašević, D., Giurca, A., Wagner, G., Devedžić, V.: On Interchanging Between OWL/SWRL and UML/OCL. In: OCLApps 2006, Genova, Italy, pp. 81–95 (2006)
9. R2ML Specification, <http://oxygen.informatik.tu-cottbus.de/R2ML>
10. RIF Basic Logic Dialect, <http://www.w3.org/TR/rif-blld/>
11. RIF Datatypes and Built-Ins 1.0, <http://www.w3.org/TR/rif-dtb/>
12. Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: A Metamodel and UML Profile for Rule-extended OWL DL Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 303–316. Springer, Heidelberg (2006)
13. Wagner, G., Tabet, S., Boley, H.: MOF-RuleML: The Abstract Syntax of RuleML as MOF Model. In: Integrate 2003, OMG Meeting, Boston (2003)

# f-SPARQL: A Flexible Extension of SPARQL

Jingwei Cheng, Z.M. Ma, and Li Yan

Northeastern University, Shenyang, 110819, China  
cjingwei@gmail.com, mazongmin@ise.neu.edu.cn

**Abstract.** We propose an flexible extension of SPARQL by introducing fuzzy set theory and the  $\alpha$ -cut operation of fuzzy numbers into SPARQL. We show how to efficiently compute the top-k answers of flexible SPARQL queries with regard to membership degrees and user-defined weights. Based on our method, a flexible query service is implemented and evaluated on the LUBM platform. Results of a preliminary user study demonstrate that the method for construction and translation of flexible queries and results ranking can capture the users's intension effectively.

## 1 Introduction

RDF is a directed, labeled graph data format for representing information in the Web. The RDF abstract syntax is a set of triples, called the RDF graph. SPARQL [1] can be used to express queries over RDF data sets. SPARQL FILTERs restrict solutions to those for which the FILTER expression evaluates to TRUE. Until now, however, standard SPARQL processes information only in a crisp way. Therefore, existing SPARQL implementations, such as ARQ [2] and Sesame [3], do not allow users to form queries with preferences or vagueness, which could be desirable for the following reasons [4]: (i) to express soft query conditions; (ii) to control the size of the answers; (iii) to produce a discriminated answer.

*Example 1.* An advertisement company requires 30 models which are *close to* 175cm and *not very young and not very old*.

Apparently, SPARQL can not efficiently express and answer such a request. To address this problem, we propose a flexible extension of SPARQL, called f-SPARQL. It allows, in FILTER constraint, the occurrence of fuzzy terms, e.g. *young* or *tall*, and fuzzy operators, e.g. *close to* or *at most*. The fuzzy terms and fuzzy operators along with the query variables form the so-called fuzzy constraints. Furthermore, we need to take into account the membership degree threshold for every fuzzy constraint. The reason for this is the fact that each tuple satisfies a fuzzy constraint to a certain degree and hence if no threshold specified, all tuples are retrieved always. To avoid reinvent the wheel, we develop a set of translation rules for converting flexible queries formalized with f-SPARQL into

---

<sup>1</sup> <http://jena.sourceforge.net/ARQ/>

<sup>2</sup> <http://www.openrdf.org/>

traditional crisp SPARQL queries, and thus we can still make use of existing SPARQL implementations. However, there is still a sweet nuisance if more than 30 models satisfied the fuzzy constraint user specified. This poses a new challenge in case we must rank these candidates according to a certain standard and select the top 30. The introduction of weights allows different fuzzy constraints to have different importance. We show how to efficiently compute the top-k answers of f-SPARQL queries.

## 2 Fuzzy Set Theory

In fuzzy set theory [3], an element belongs to a set with certain degree, which is described with the aid of a membership function valued in the real unit interval [0,1]. A fuzzy set  $A$  with regard to a universe  $U$  is characterized by a membership function  $\mu_A : U \rightarrow [0, 1]$  (or simply  $A(x)$ ), which assigns a membership degree to each element  $u$  in  $U$ , denoted by  $\mu_A(u)$ .  $\mu_A(u)$  gives us an assessment of the degree that  $u$  belongs to  $A$ . Typically, if  $\mu_A(u) = 1$  then  $u$  definitely belongs to  $A$ , whereas  $\mu_A(u) \geq 0.8$  means that  $u$  is “likely” to be an element of  $A$ . In addition, when using Gödel T-norm, T-conorm and Lukasiewicz negation for interpreting conjunctions, disjunctions and complements respectively, we have: for all  $u \in U$  and for all fuzzy sets  $A_1, A_2$  with respect to  $U$ ,  $\mu_{A_1 \cap A_2}(u) = \min\{\mu_{A_1}(u), \mu_{A_2}(u)\}$ ,  $\mu_{A_1 \cup A_2}(u) = \max\{\mu_{A_1}(u), \mu_{A_2}(u)\}$ ,  $\mu_{\bar{A}}(u) = 1 - \mu_A(u)$ , where  $\bar{A}$  is the complement of  $A$  in  $U$ .

A fuzzy set  $A$  w.r.t  $U$  is called *convex*, iff for all  $u_1, u_2 \in U$ ,  $\mu_A(\lambda u_1 + (1 - \lambda)u_2) \geq \min(\mu_A(u_1), \mu_A(u_2))$ , where  $\lambda \in [0, 1]$ . A fuzzy set  $A$  w.r.t  $U$  is called *normal*, if  $\exists u \in U$ , s.t.  $\mu_A(u) = 1$ . A *fuzzy number* is a convex, normal fuzzy set. The set of elements whose membership degrees in  $A$  are greater than (greater than or equal to)  $\alpha$ , where  $0 \leq \alpha < 1$  ( $0 < \alpha \leq 1$ ), is called the *strong (weak)  $\alpha$ -cut* of  $A$ , denoted by  $A_{\alpha+} = \{u \in U | \mu_A(u) > \alpha\}$  and  $A_\alpha = \{u \in U | \mu_A(u) \geq \alpha\}$ . The  $\alpha$ -cut of a fuzzy number corresponds to an interval of  $U$ . Let  $A$  and  $B$  be two fuzzy numbers of  $U$ , and  $A_\alpha = [x_1, y_1]$  and  $B_\alpha = [x_2, y_2]$  the  $\alpha$ -cuts of  $A$  and  $B$ , respectively. Then we have  $(A \cup B)_\alpha = A_\alpha \sqcup B_\alpha$ ,  $(A \cap B)_\alpha = A_\alpha \sqcap B_\alpha$ , where  $\sqcup$  and  $\sqcap$  denote the union and the intersection operators between two intervals, respectively. They are defined as follows,

$$A_\alpha \sqcup B_\alpha = \begin{cases} [x_1, y_1] \sqcup [x_2, y_2], & \text{if } A_\alpha \cap B_\alpha = \emptyset \\ [\min(x_1, x_2), \max(y_1, y_2)], & \text{if } A_\alpha \cap B_\alpha \neq \emptyset \end{cases} \tag{1}$$

$$A_\alpha \sqcap B_\alpha = \begin{cases} \emptyset, & \text{if } A_\alpha \cap B_\alpha = \emptyset \\ [\max(x_1, x_2), \min(y_1, y_2)], & \text{if } A_\alpha \cap B_\alpha \neq \emptyset \end{cases} \tag{2}$$

## 3 f-SPARQL

In this section, we define a flexible extension of SPARQL. As in the case with databases [2][4], the extension mainly takes place in the FILTER constraint part. For each such a flexible query, we declare with #FQ# before a SELECT query indicating that this is the case. In the FILTER part, the original expression of the form FILTER (?X op Y) is extended by allowing fuzzy terms and fuzzy operators.

### 3.1 Fuzzy Terms Extension

The basic form of FILTER constraint allowing fuzzy terms is FILTER (?X  $\theta$  FT) [WITH  $\alpha$ ], where FT denotes a fuzzy term, e.g. “tall” or “young”, which corresponds to a fuzzy number, and  $\theta$  is one of the operators including  $>$ ,  $<$ ,  $=$ ,  $>=$ ,  $<=$ ,  $!=$ , *between*, and *not between*. Note that if the operator  $\theta$  is *between* or *not between*, then the syntax of FILTER constraint is a little different, i.e., FILTER (?X *between/not between* FT1 and FT2) [WITH  $\alpha$ ], where FT1 and FT2 are fuzzy numbers. The optional parameter [WITH  $\alpha$ ] indicates the condition that must be satisfied as the minimum membership degree threshold in  $[0,1]$ . Users choose an appropriate value of  $\alpha$  to express his/her requirement. If not specified, then use 1 as default. There are three kinds of fuzzy terms : *simple (atomic) fuzzy term*, *modified (composite) fuzzy term*, and *compound fuzzy term*.

**Simple fuzzy terms.** A simple fuzzy term such as “young” or “tall” is defined in terms of a fuzzy number with a membership function.

**Modified fuzzy terms.** A modified fuzzy term, e.g. “very young” or “more or less tall”, is described by a simple fuzzy term and a fuzzy modifier. Let  $A$  be a simple fuzzy term represented by a fuzzy number in the universe of discourse  $U$  and its membership function is  $\mu_A : U \rightarrow [0, 1]$ , then we have the following rules.

- Concentration rule:  $\mu_{\text{very } A}(u) = (\mu_A(u))^2$ . More generally,  $\mu_{\text{very } \dots \text{very } A}(u) = (\mu_A(u))^{2 \times (\text{times of very})}$
- Dilation rule:  $\mu_{\text{more or less } A}(u) = (\mu_A(u))^{1/2}$

**Compound fuzzy terms.** A compound fuzzy term is represented by simple fuzzy terms or modified fuzzy terms connected by *or* (union), *and* (intersection), or *not* (complement) connectors, such as “young or very young”.

### 3.2 Fuzzy Operators

Now consider fuzzy relations as operators within FILTER constraint, formalized as FILTER (?X  $\tilde{\theta}$  Y) [with  $\alpha$ ], where  $\tilde{\theta}$  denotes a fuzzy operator, Y a string, a integer or any other data types allowed in RDF. The fuzzy operator  $\tilde{\theta}$  and the crisp value Y constitute a fuzzy number  $\tilde{\theta}Y$ .

We, in this paper, mainly discuss three types of fuzzy relations, which are “close to (around)”, “at least”, and “at most”.

**Close to.** According to [5], the membership function of the fuzzy number “close to Y (around Y)” on the universe of discourse  $U$  is defined as

$$\mu_{\text{close to } Y}(u) = \frac{1}{1 + (\frac{u-Y}{b})^2}. \tag{3}$$

It is worth noting that the fuzzy number above is a simple fuzzy term. Based on it, we may have modified fuzzy terms, e.g. “very close to Y”, “very very ... very close to Y”, and “more or less close to Y”.



**At least.** The membership function of the fuzzy number “at least  $Y$ ” on the universe of discourse is defined by

$$\mu_{at\ least\ Y}(u) = \begin{cases} 0, & \text{if } u \leq \omega \\ \frac{u-\omega}{Y-\omega}, & \text{if } \omega < u < Y \\ 1, & \text{if } u \geq Y \end{cases} \quad (4)$$

**At most.** The membership function of the fuzzy number “at most  $Y$ ” on the universe of discourse is defined by

$$\mu_{at\ most\ Y}(u) = \begin{cases} 1, & \text{if } u \leq Y \\ \frac{\delta-u}{\delta-Y}, & \text{if } Y < u < \delta \\ 0, & \text{if } u \geq \delta \end{cases} \quad (5)$$

Note that, parameters of  $b$ ,  $\omega$ , and  $\delta$  in membership functions of “close to  $Y$ ”, “at least  $Y$ ”, and “at most  $Y$ ” are chosen according to the concrete value of  $Y$  and vary over different use cases.

### 3.3 User-Defined Weights

In the context of SPARQL, a user may have different preferences for different triple patterns, e.g. `?X ex:hasAge ?Age` and `?X ex:hasHeight ?Height`. In order to provide users with a query language of adequate expressibility, we additionally allow users to specify, for each triple pattern, the weight of importance. The query criteria user specified can reflect the users’s preferences on the triple. For each such a flexible query, we declare with `#top-k FQ#` with  $k$  before a `SELECT` query to indicate the query type.

*Example 2.* Given the query in Example [1](#), the manager of the advertisement company believes models with similar heights (close to 175cm) is the most important factor in the advertisement, so he/she specify a weight of 0.8 for the triple `?X ex:hasHeight ?Height`, and therefore 0.2 for `?X ex:hasAge ?Age`. The f-SPARQL query is formed as follows.

```
#top-k FQ# with 30
SELECT ?X ?Age ?Height WHERE {
  ?X rdf:type Model
  ?X ex:hasAge ?Age with 0.2.
  FILTER (?Age=not very young && ?Age=not very old) with 0.9.
  ?X ex:hasHeight ?Height with 0.8 .
  FILTER (?Height close to 175cm) with 0.8.
}
```

### 3.4 Syntax of f-SPARQL

Table [1](#) presents the syntax of f-SPARQL. f-SPARQL extends two elements of SPARQL, i.e. the “Query” and the “Constrain”. Each `SELECT` query is extended with the element *QueryType*, which can be further divide into `#FQ#`

(flexible queries) and #top-k FQ# with k (top-k flexible queries). The “Constraint” element is extended with an additional “FlexibleExpression” element, as illustrated in the above subsections.

**Table 1.** f-SPARQL modifications to the SPARQL standard grammar

Query	::= Prologue ( <i>QueryType</i> SelectQuery   ConstructQuery   DescribeQuery  AskQuery )
QueryType	::= '#FQ#'   #top-k FQ# with k
Constraint	::= BrackettedExpression   BuiltInCall   FunctionCall   FlexibleExpression
FlexibleExpression	::= FuzzyTermExpression   FuzzyOperatorExpression
FuzzyTermExpression	::= (Var ['=', '!=', '>', '>=', '<', '<='] FuzzyTerm)? [with threshold]
FuzzyOperatorExpression	::= Var FuzzyOperator NumericLiteral
FuzzyOperator	::= (Modifier)*FuzzyOperator
FuzzyTerm	::= FuzzyTerm and FuzzyTerm   FuzzyTerm or FuzzyTerm   not FuzzyTerm   ModifiedFuzzyTerm
ModifiedFuzzyTerm	::=(Modifier)* ModifiedFuzzyTerm   (Modifier)* SimpleFuzzyTerm

## 4 Query Translation

In order to still make use of existing SPARQL implementations for coping with fuzzy queries, a set of translation rules is needed.

### 4.1 Translation of Fuzzy Terms

We make a case study for different types of fuzzy terms.

#### Case 1: FT is a simple fuzzy term or a modified fuzzy term

Recall the query form `FILTER (?X θ FT [WITH α])` introduced in Section 3.1. Let  $\alpha$  be a given threshold,  $FT_\alpha$  the  $\alpha$ -cut of FT. It is clear that  $FT_\alpha$  is an interval, written as  $FT_\alpha = [a, b]$ . Fuzzy queries are then translated into crisp ones according to the rules given in Table 2.

**Table 2.** Translation rules for fuzzy term FT with  $FT_\alpha = [a, b]$

Flexible queries	Queries translated
<code>FILTER (?X = FT WITH α)</code>	<code>FILTER (?X &gt;= a &amp;&amp; ?X &lt;= b)</code>
<code>FILTER (?X &gt; FT WITH α)</code>	<code>FILTER (?X &gt; b)</code>
<code>FILTER (?X &gt;= FT WITH α)</code>	<code>FILTER (?X &gt;= b)</code>
<code>FILTER (?X &lt; FT WITH α)</code>	<code>FILTER (?X &lt; a)</code>
<code>FILTER (?X &lt;= FT WITH α)</code>	<code>FILTER (?X &lt;= a)</code>
<code>FILTER (?X != FT WITH α)</code>	<code>FILTER (?X &lt; a    ?X &gt; b)</code>

#### Case 2: FT is a compound fuzzy terms

Firstly, we consider compound fuzzy terms FT of the form “FT1 and FT2”, where FT1 and FT2 are simple or modified fuzzy terms. As is shown in Equation (11),

the  $\alpha$ -cut of FT is of the form  $[a, b] \sqcup [c, d]$ . Queries of this kind can be translated into crisp SPARQL according to the rules shown in Table 3. Then, consider compound fuzzy terms FT of the form “FT1 or FT2”, where FT1 and FT2 are simple or modified fuzzy terms. Since the  $\alpha$ -cut of FT is of the form  $[a, b]$  when  $A \cap B \neq \emptyset$  (see Equation (2)), queries of this kind can be translated using rules in Table 2. For FILTER constraints of the form  $?X \theta \text{ not FT}$ , they can firstly be translated into the form of  $?X \text{ not } \theta \text{ FT}$ . The  $\text{not } \theta$  is defined as, e.g.  $\text{not } > = <=$ , and  $\text{not } = = !=$ . Compound queries in more complex forms can be translated by iteratively using the aforementioned translation rules.

**Table 3.** Translation rules for fuzzy term FT with  $FT_\alpha = [a, b] \sqcup [c, d]$

Flexible queries	Queries translated
FILTER (?X = FT WITH $\alpha$ )	FILTER (?X >= a && ?X <= b    ?X >= c && ?X <= d)
FILTER (?X > FT WITH $\alpha$ )	FILTER (?X > max(b, d) )
FILTER (?X >= FT WITH $\alpha$ )	FILTER (?X >= max(b, d) )
FILTER (?X < FT WITH $\alpha$ )	FILTER (?X < min(a, c) )
FILTER (?X <= FT WITH $\alpha$ )	FILTER (?X <= min(a, c) )
FILTER (?X != FT WITH $\alpha$ )	FILTER (?X < a    ?X > b    ?X < c    ?X > d)

### 4.2 Translation of Fuzzy Operators

Given the syntax of FILTER constraint with fuzzy relations as fuzzy operators, i.e., FILTER (?X  $\tilde{\theta}$  Y), the fuzzy operator along with the crisp value Y is actually a fuzzy term, so the expression can be translated into an equivalent one of the form FILTER (?X = FT), where FT denotes  $\tilde{\theta} Y$ .

## 5 Ranking

A naive solution to the top-k retrieval problem is illustrated with the query in Example 1 as follows: (i) A flexible query is firstly translated into a crisp SPARQL query, then the latter is sent to a SPARQL implementation, e.g. ARQ. (ii) ARQ returns answers for the translated crisp query. (iii) The answers are ranked according to a scoring function.

Given an f-SPARQL query Q, the FILTER constraint conditions is  $F = (F_1, \dots, F_n)$ , where  $F_i$  denotes the  $i$ -th constraint condition. We use  $A = (A_1, \dots, A_n)$  to denote one of the answers for  $F$ , then the score of  $A$  can be calculated with the following scoring function.

$$Score(A) = \sum_{i=1}^n memDegree(A_i) \times weight(F_i). \tag{6}$$

where  $memDegree(A_i)$  denotes the membership degree of the answer for  $F_i$ , and  $weight(F_i)$  denotes the user-defined weight for  $F_i$ .

## 6 Experiment

We generate, from the Lehigh University Benchmark LBUM [6], the data set which contains 6,000k triples. The data is stored in and managed by Mysql 5.0.11. The translation and ranking algorithms are implemented using JAVA. Jena ARQ and SDB (which provides for large scale storage and query of RDF data sets using conventional SQL databases) are used for crisp queries processing. The applications run on a windows XP professional system with P4 3G CPU and with 1G RAM.

We develop four queries which are shown in Table 4, and request the top-100 results. The constraint with *italic* font denotes the first constraint, while the constraint with **bold** font denotes the second constraint.

**Table 4.** Experimental queries

$F_1$ : Find the <i>not famous</i> and <b>busy</b> teachers
$F_2$ : Find the <i>famous</i> and <b>busy</b> teachers
$F_3$ : Find the <i>not famous</i> and <b>busy</b> students
$F_4$ : Find the <i>famous</i> and <b>busy</b> students

We use the membership function of “busy” and “famous” in [7].

$$Busy(n) = \frac{2}{1 + \exp(-0.4n)} - 1 \quad (7)$$

$$Famous(n) = \frac{2}{1 + \exp(-0.1n)} - 1 \quad (8)$$

where  $n$  in  $Busy(n)$  represents the number of courses taken by a student or taught by a teacher, and  $n$  in  $Famous(n)$  represents the number of papers published by a teacher or a student.

We employ five users to specify for every constraint the threshold and the weight. The response time of queries are listed in the Table 5. We invite the five testers to evaluate how closely the results satisfied their initial intentions. We define scores of 2, 1, and 0 for “very satisfying”, “satisfying”, and “dissatisfying”, respectively. Table 6 shows the evaluation scores for top-100 results. The preliminary user study demonstrate that our flexible queries and answer ranking methods can capture the user intension effectively.

## 7 Conclusions

In order to provide users with a easy-to-grasp yet powerful way for querying the RDF data sets, we have presented, in this paper, a flexible extension of SPARQL, called f-SPARQL. f-SPARQL extends SPARQL by allowing the occurrence of fuzzy terms and fuzzy relations. Queries formulated in f-SPARQL differ from

**Table 5.** Response time of queries

	$F_1$	$F_2$	$F_3$	$F_4$
user1	176	202	195	217
user2	199	231	189	272
user3	353	243	215	271
user4	256	277	316	317
user5	376	272	295	257

**Table 6.** Evaluation results

	$F_1$	$F_2$	$F_3$	$F_4$
user1	166	173	185	147
user2	176	155	195	177
user3	176	200	193	177
user4	156	182	168	198
user5	190	187	186	200

both crisp SPARQL queries against crisp RDF data sources, and fuzzy extended SPARQL against fuzzy RDF/OWLs, e.g. in [7]. On the basis of fuzzy set theory and  $\alpha$ -cut of fuzzy number, a set of translation rules is developed, converting f-SPARQL into crisp ones, so as to take advantage of existing implementations of SPARQL. We also have proposed a scoring method for calculating the order of query results. The experiments on real data sets identified that flexible queries can express user intentions efficiently. Rather, the top-k ranked answers can be returned fast and achieve high accuracy as well. It would be interesting to investigate how to minimize the updating cost when the data sets and membership functions are varied.

**Acknowledgments.** The work is supported by the National Natural Science Foundation of China (60873010) and the Fundamental Research Funds for the Central Universities (N090504005 and N090604012), and in part by the Program for New Century Excellent Talents in University (NCET- 05-0288).

## References

1. Prud'hommeaux, E., Seaborne, A.: Sparql query language for rdf. W3C Recommendation (January 15, 2008)
2. Kraft, D.H., Petry, F.E.: Fuzzy information systems: managing uncertainty in databases and information retrieval systems. *Fuzzy Sets and Systems* 90(2), 183–191 (1997); *Fuzzy Sets: Where Do We Stand? Where Do We Go?*
3. Zadeh, L.A.: Fuzzy sets. *Information and Control* 8(3), 338–353 (1965)
4. Ma, Z.M., Yan, L.: Generalization of strategies for fuzzy query translation in classical relational databases. *Information & Software Technology* 49(2), 172–180 (2007)
5. Chen, S.M., Jong, W.T.: Fuzzy query translation for relational database systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 27(4), 714–721 (1997)
6. Guo, Y., Pan, Z., Heflin, J.: An evaluation of knowledge base systems for large owl datasets. In: *International Semantic Web Conference*, pp. 274–288 (2004)
7. Pan, J.Z., Stamou, G.B., Stoilos, G., Taylor, S., Thomas, E.: Scalable querying services over fuzzy ontologies. In: *WWW*, pp. 575–584 (2008)

# Geo Linked Data

Francisco J. Lopez-Pellicer<sup>1</sup>, Mário J. Silva<sup>2</sup>, Marcirio Chaves<sup>3</sup>,  
F. Javier Zarazaga-Soria<sup>1</sup>, and Pedro R. Muro-Medrano<sup>1</sup>

<sup>1</sup> IAAA, Universidad de Zaragoza, Spain  
{fjlopez,javy,prmuro}@unizar.es

<sup>2</sup> LaSIGE, Universidade de Lisboa, Portugal  
mjs@di.fc.ul.pt

<sup>3</sup> Universidade Atlântica, Portugal  
marcirioc@uatlantica.pt

**Abstract.** Semantic Web applications that include map visualization clients are becoming common. When the description of an entity contains coordinate pairs, semantic applications often lay them as pins on maps provided by Web mapping service applications, such as Google Maps. Nowadays, semantic applications cannot guarantee that those maps provide spatial information related to the entities pinned to them. To address this issue, this paper proposes a refinement of Linked Data practices, named Geo Linked Data, which defines a lightweight semantic infrastructure to relate URIs that identify real world entities with geospatial Web resources, such as maps.

## 1 Introduction

Location pins on maps are a powerful way to convey spatial information, but they are just presentation tools in Semantic Web applications. Today, we can dereference a URI, such as <http://dbpedia.org/resource/Lisbon>, to obtain machine-processable data about the city of Lisbon, and use a location pin to show the user where Lisbon is. However, the use of such maps in Semantic Web applications can produce undesired situations, such as representing data about Lisbon over a map provided by Google Maps that shows the exception message: “*We are sorry, but we don't have imagery at this zoom level for this region*”. We believe that the juxtaposition of data and maps in Semantic Web applications should not be justified by the location. The juxtaposition of data and maps could only be justified when the location and the semantics are compatible.

The research question behind this issue is *how to relate semantic and geospatial depictions of a real world entity in the Semantic Web*. In other words, we are looking for a feasible approach for using maps as data in Semantic Web applications. This paper proposes a solution that intertwines formal and geospatial depictions based on the Linked Data principles. Linked Data is an initiative for interconnecting data on the Web using the Semantic Web standards, so that users and machines can explore data on the Web. Linked Data [15] promotes (1) the use of HTTP URIs for identifying concepts and real world entities, and (2)

a sound use of the HTTP protocol to assert that a given entity identified by a URI has a Web resource as description.

The proposed solution refines Linked Data practices to use descriptions available as geospatial Web resources, such as a map from a Web mapping service. Web mapping services are part of the Geospatial Web or Geoweb [16], a set of specifications and applications which adopts the Internet and Web services to publish, access and transform geospatial content related to real world entities. Our contribution involves (1) the *characterization of geospatial proxies*, that is, geospatial Web resources that could complement Semantic Web descriptions about entities in some scenarios, (2) the identification of the *roles* of these proxies in semantic applications, (3) the *recipe* for extending Linked Data with geospatial proxies, and (4) the *advertisement of presence, role and location* of geospatial proxies using RDF graphs. We use the term *Geo Linked Data*, as it specializes the practice followed by the Linked Data community.

This paper is organized as follows: Section 2 describes related work, Section 3 details Geo Linked Data concepts, Section 4 illustrates the role of Geo Linked Data in a scenario, and, finally, we present conclusions and future work.

## 2 Related Work

Tabulator [3] and DBpedia Mobile [2] represent the mainstream approach for the visualization of geographic properties in Semantic Web applications: the extraction of geographic points from a description in RDF, and then, its conversion into a marker added to a map viewer client that shows the location of the entity. That is, the geospatial meaning of an entity as complex as Lisbon is often (1) *simplified* to  $(x, y)$  pairs, and then (2) *overlaid* on a geospatial description generated in a remote Geoweb server.

The Geoweb emerged in 2000 with the service specification Web Map Server 1.0 [5] published by Open Geospatial Consortium (OGC). OGC has also published service specifications for data access and catalogues [12], and geographic markup languages, such as GML [13]. Nevertheless, the release in 2005 of Google Maps, which enabled an easy integration of maps to existing Web applications, made the Geoweb part of the mainstream Web.

The Geoweb is perceived from the Semantic Web community as a provider of datasets rich in geographic descriptions that need to be extracted from their silos. This is the approach of the publication as RDF of OpenStreetMap [1]. On the other side, the Semantic Web is perceived from the Geoweb community as the provider of formal infrastructure (i.e. Semantic Web standards, such as OWL). The application of Semantic Web technologies includes from enabling meaningful geospatial information retrieval using geospatial ontologies [6] to the development of profiles of Geoweb services with RDF and OWL support [7]. With Geo Linked Data, we propose an alternative approach: the use of Semantic Web best practices to publish Geoweb resources alongside their metadata in RDF.

### 3 Geo Linked Data

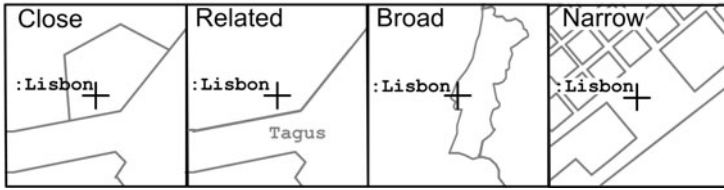
In the Linked Data approach, when a URI acts as identifier for an entity, which may exist outside the Web, the URI can be dereferenced to a Web resource that provides useful information about the entity [15]. The user agent works under the following assumption: when the URI gives access to a Web resource with a different URI at a given time, the Web resource could be interpreted as a *proxy for* an entity, at least in that given time. The concept *proxy for* is part of the *identity of resources and entities on the Web* (IRE) model proposed by Presutti and Gangemi [14], a framework for reasoning when a Web URI can be associated to an entity. The *proxy for* association between a Web resource (e.g. a semantic description) and an entity (e.g. Lisbon) means that the representation of the Web resource (e.g. a RDF document) materializes information (e.g. a pair of geographic coordinates) about the entity.

The IRE model classifies the *proxy for* relations as *exact* or *approximate*, and as *formal* or *informal*. An *exact proxy for* relation means that the Web resource only describes one entity, and otherwise it is *approximate*. Nevertheless, an *exact proxy* may contain references to related entities. For example, satellite images about Portugal may contain parts of the Atlantic Ocean and Spain. However, satellite images are meant to be *exact proxies for* Portugal. A *formal proxy for* relation means that the Web resource is represented in a formal language. If not, it is an *informal proxy for* relation.

The *proxy for* concept is independent of the technology and the information about the entity. Hence, the definition of *proxy for* is applicable when the Web resource is a Geoweb resource conveying spatial characteristics of other entities. We designate as a *geospatial proxy* any Web resource conveying spatial information about other entities using Geoweb standards (e.g. a GML document describing the location of Lisbon, a satellite image in JPEG describing the Earth). A *geospatial proxy* is an *exact geospatial proxy* if it only describes one entity.

Linked Data principles require that the information must be available in RDF and, for humans, should be available in HTML. We can characterize RDF and HTML representations as having *exact formal proxy for* and *exact informal proxy for* relations with an entity, respectively. For humans, an *exact geospatial proxy* and a HTML representation describing the same entity could contain equivalent spatial information. From a machine-processable point of view, an *exact geospatial proxy* could be considered as an alternative representation of the spatial information of an RDF representation when it is possible to map the content of the geospatial proxy to a formal model. The mapping is possible because the information in Geoweb representations is specified by standardized data models [9], and we can find in the literature mappings to formal models, such as the proposed in the Geospatial Semantic Web Interoperability Experiment [10]. We can conclude that *exact geospatial proxies* conform to Linked Data rules. That is, when a semantic application requires spatial information, they could be an alternative for HTML and RDF representations.





**Fig. 1.** Different examples of maps that may act as *geospatial proxy* for the official boundary of Lisbon, capital of Portugal: *close* is a sketch of the boundary, *related* shows the area where the boundary of Lisbon is, *broad* shows the boundary of Portugal, and *narrow* shows the *Praça do Comércio* (Commerce Square), a landmark of Lisbon.

### 3.1 Roles of a Geoweb Description

We identify four roles that could be useful to understand how to use Geoweb descriptions in Semantic Web applications (Figure 1):

**close** A *close proxy* is a proxy that some information systems can use as source for an alternative identifier of the entity. For example, a marker that identifies an entity in a map provided by a GML document is a *close proxy*.

**related** A *related proxy* provides an indirect description of the resource through the spatial characteristics of the proxy. For example, a satellite image of an area is a *related proxy* of the entities of the area.

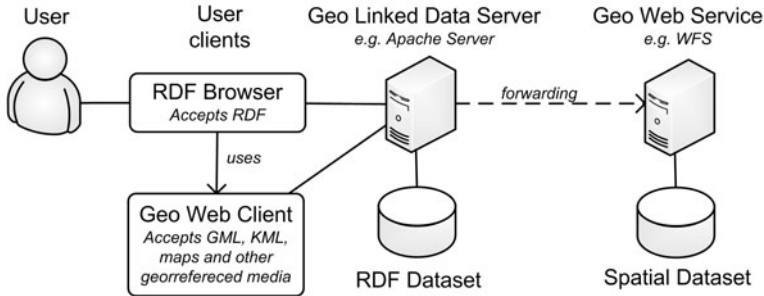
**broad** A *broad proxy* is a kind of related proxy that realizes essential characteristics of an entity that is a parent of the described entity. For example, a large regional map is a broad geospatial proxy for capital cities.

**narrow** A *narrow proxy* is similar to a broad proxy but realizes essential characteristics of an entity part of the described entity. An example is a map that shows a landmark of a city when the described entity is the city.

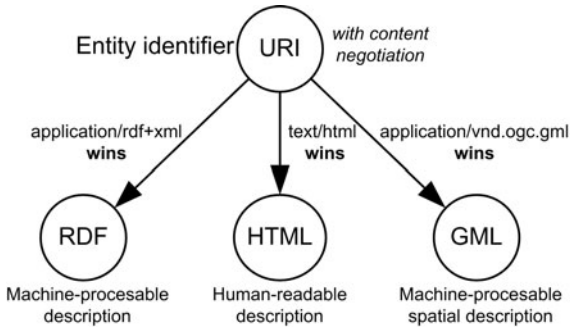
### 3.2 Geo Linked Data Recipe

We can summarize the Geo Linked Data recipe as follows (see Figure 2b and Figure 2a):

1. Entity URIs are dereferenced following the Linked Data principles (see Bizer et al. [4]).
2. If such a URI is dereferenced accepting a Geoweb MIME-type (e.g. *application/vnd.ogc.gml*), the server must return a geospatial description of the entity that matches the MIME-type (e.g. GML) when available.
3. RDF descriptions may contain properties that advertise the presence and role of Geoweb descriptions; clients can use these clues to discover their MIME-type and the best use of these representations.
4. RDF descriptions may contain RDF links to navigate to geospatial descriptions provided by Geoweb services.



(a) Simple setup with a Geoweb data access service which serves geospatial resources in GML.



(b) The content negotiation model of Linked Data is extended with support to Geoweb MIME-types.

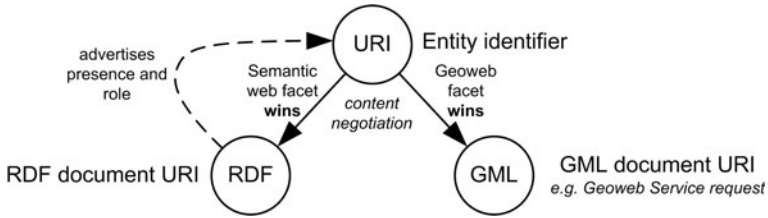
**Fig. 2.** Schematic representation of the relations in a simple setup. With a URI that identifies a real world resource (e.g. Lisbon), the user client can access a RDF or HTML representation through the Geo Linked Data Server in the same server, or a geospatial representation (GML) in a Geoweb service.

### 3.3 Making Proxies Visible

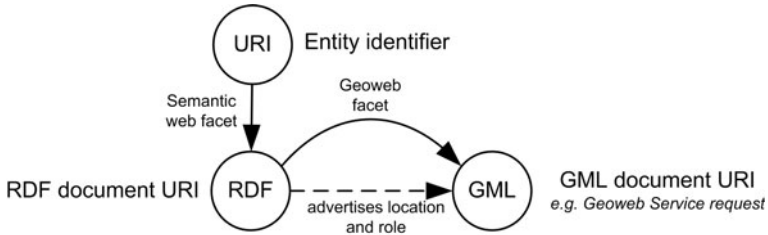
We can use RDF to advertise the *presence*, the *role* and the *location* of a geospatial proxy. We can relate a URI to a geospatial proxy adding the following statement to the RDF description of the entity:

`<entity URI> p <MIME-type> .`

The semantics of the property *p* should entail that there is a dereferenceable geospatial proxy for the entity identified by the URI. The property *p* could also describe the role of the proxy. Figure 3a shows how this assertion should be interpreted. The URI can be dereferenced again, but this time requesting the MIME-type asserted in the statement, to retrieve the geospatial representation. The server that owns the URI is responsible for redirecting the user to the effective location of the geospatial proxy. This way of advertising presence is limited to only one proxy for each content type. If the server cannot be properly



(a) The document asserts presence and role of the proxy.



(b) The document asserts explicit location and role of the proxy.

**Fig. 3.** Advertisement of geospatial proxies in RDF documents.

configured, or the entity has several geospatial proxies with the same MIME-type, we could make an explicit advertisement of the location of each geospatial proxy. The advertisement requires the assertion of statements like:

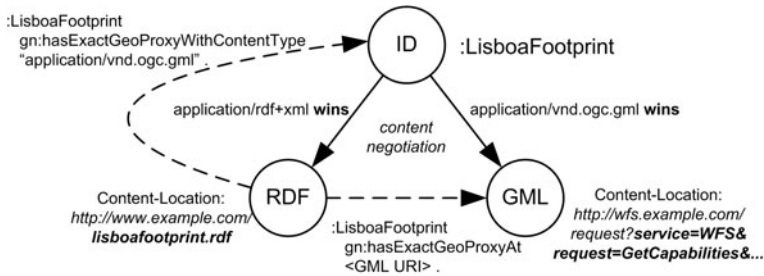
`<entity URI> q <Geoweb document URI> .`

The semantics of the property `q` should assert that the Geoweb resource is a geospatial proxy, and, additionally, describe its role (see Figure 3b).

## 4 A Geo Linked Data Scenario

*Geo-Net-PT 02* [11] is an authoritative RDF dataset about named places of Portugal available in the XLDB Node of Linguateca [1]. *Geo-Net-PT 02* defines 701,209 instances, most of them named places. Postal codes, streets and settlements are the most common types. *Geo-Net-PT 02* has 21 different sources. The *Instituto Geográfico Português* provides the Official Administrative Boundaries Map (CAOP) of Portugal. The CAOP dataset is available through a Geoweb data access service application [8], which makes accessible up-to-date GML documents that are *exact geospatial proxies* of the footprints of the administrative units. Figure 4 describes an example of the advertisement of presence, role and location of geospatial proxies in the *Geo-Net-PT 02* dataset. The exact link is possible for administrative units. Each administrative unit in Portugal has a unique official identifier as attribute, which exists in the CAOP server and in *Geo-Net-PT 02*. The setup of the Geo Linked Data endpoint follows the recipe

<sup>1</sup> [http://xldb.di.fc.ul.pt/wiki/Geo-Net-PT\\_02](http://xldb.di.fc.ul.pt/wiki/Geo-Net-PT_02)



**Fig. 4.** Use case scenario: geospatial proxies for Geo-Net-PT 02; the prefix `gn:` identifies the terms added to the vocabulary for advertising Geo Linked Data.

presented in Subsection 3.2. It allows clients to dereference a Geo-Net-PT 02 URI, such as `:LisboaFootprint`, and discover that a representation in GML is also available. Then, the client can dereference again but asking for the MIME-type of GML, and then, for example, display the result on a map.

## 5 Conclusions

This paper introduced Geo Linked Data, an approach for incorporating references to Geoweb data and services in the Semantic Web. The references, named geospatial proxies, were characterized, and the example based on Geo-Net-PT 02 showed how to use this approach in a real scenario. Geo Linked Data requires the update of RDF datasets with assertions of presence of spatial descriptions, which may include information about their role and location. The roles ease the use of Geo Linked Data by Semantic Web applications. The locations allow users to navigate between Linked Data and Geoweb datasets. Future work will include the formalization of the categorization of geospatial proxies, and the automatic generation and classification of bindings between RDF datasets and Geoweb resources.

## Acknowledgments

Thanks to Aneta J. Florczyk and Walter Renteria Agualimpia for their comments and insights. This work was partially supported by FCT (Portuguese research funding agency) for its LaSIGE Multi-annual support, GREASE-II project (grant PTDC/EIA/73614/2006), by the Spanish Government (projects “España Virtual” CENIT 2008-1030 and TIN2009-10971), the Aragón Government (project PI075/08), and GeoSpatiumLab S.L.

## References

1. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData – Adding a spatial Dimension to the Web of Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009)

2. Becker, C., Bizer, C.: Exploring the Geospatial Semantic Web with DBpedia Mobile. *Web Sem.: Science, Services and Agents on the WWW* 7(4), 278–286 (2009)
3. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: *Proceedings of the 3rd International Semantic Web User Interaction Workshop* (2006)
4. Bizer, C., Cyganiak, R., Heath, T.: *How to Publish Linked Data on the Web* (2007), <http://www4.wiwi.fu-berlin.de/bizer/pub/LinkedDataTutorial/>
5. Doyle, A.: *OpenGIS Web Map Server Interface Implementation Specification*. OGC Standard OGC 00-028. Open Geospatial Consortium Inc., version 1.0.0 (2000)
6. Egenhofer, M.J.: Toward the Semantic Geospatial Web. In: *GIS '02: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*, pp. 1–4. ACM, New York (2002)
7. Janowicz, K., Schade, S., Bröring, A., Kefler, C., Maué, P., Stasch, C.: Semantic Enablement for SDIs. *Transactions in GIS* 14(2), 111–129 (2010)
8. Julião, R.P., Mas, S., Rodriguez, A., Furtado, D.: Portugal and Spain twin SDI's from national projects to an Iberian SDI. In: *GSDI-11: Spatial Data Infrastructure Convergence*, Rotterdam, The Netherlands, June 15-19 (2009)
9. Kresse, W., Fadaie, K.: *ISO Standards for Geographic Information*. Springer, Berlin (2004) ISBN 978-3-54020-130-4
10. Lieberman, J.: *Geospatial Semantic Web interoperability experiment report*. OpenGIS® Discussion Paper OGC 06-002r1, OGC, Inc. (2006)
11. Lopez-Pellicer, F.J., Silva, M.J., Chaves, M.: Linkable geographic ontologies. In: *GIR '10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, pp. 1–8. ACM, New York (2010)
12. Nogueras-Iso, J., Zarazaga-Soria, F.J., Lacasta, J., Béjar, R., Muro-Medrano, P.R.: Metadata standard interoperability: application in the geographic information domain. *Computers, Environment and Urban Systems* 28(6), 611–634 (2004)
13. Portele, C.: *OpenGIS Geography Markup Language (GML) Encoding Standard*. OpenGIS Standard OGC 07-036, Open Geospatial Consortium Inc. (July 2007)
14. Presutti, V., Gangemi, A.: Identity of Resources and Entities on the Web. *International Journal on Semantic Web and Information Systems* 4(2), 49–72 (2008)
15. Sauermann, L., Cyganiak, R.: Cool URIs for the Semantic Web. [Online] W3C note, W3C (March 2008), <http://www.w3.org/TR/2008/NOTE-cooluris-20080331/>
16. Scharl, A., Tochtermann, K.: *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society*. Springer, New York (2007)

# Approximate Instance Retrieval on Ontologies<sup>\*</sup>

Tuvshintur Tserendorj<sup>1</sup>, Stephan Grimm<sup>1</sup>, and Pascal Hitzler<sup>2</sup>

<sup>1</sup> FZI Research Center for Information Technology, Karlsruhe, Germany

<sup>2</sup> Kno.e.sis Center, Wright State University, Dayton, Ohio

**Abstract.** With the development of more expressive description logics (DLs) for the Web Ontology Language OWL the question arises how we can properly deal with the high computational complexity for efficient reasoning. In application cases that require scalable reasoning with expressive ontologies, non-standard reasoning solutions such as approximate reasoning are necessary to tackle the intractability of reasoning in expressive DLs. In this paper, we are concerned with the approximation of the reasoning task of instance retrieval on DL knowledge bases, trading correctness of retrieval results for gain of speed. We introduce our notion of an approximate concept extension and we provide implementations to compute an approximate answer for a concept query by a suitable mapping to efficient database operations. Furthermore, we report on experiments of our approach on instance retrieval with the Wine ontology and discuss first results in terms of error rate and speed-up.

## 1 Introduction

For description logics, there are two main approaches to reasoning. Tableau-based methods [1] implemented in tools such as Pellet [2] and Racer [3] have been shown to be efficient for complex TBox reasoning tasks with expressive DLs. In contrast, the reasoning techniques based on reduction to disjunctive datalog as implemented in KAON2 [4] scale well for large ABoxes, with support for the DL *SHIQ*. Besides these two directions, other approaches such as rule engines and database-based techniques scale very well for large ABoxes, but are in principle limited to lightweight language fragments [5].

Observing the application domain of these approaches, an issue which remains to be investigated is the problem of scalable reasoning over expressive ontologies with large ABoxes as well as complex or large TBoxes. From a theoretical point of view we know that it is impossible to find any tractable algorithm for reasoning over expressive ontologies due to the underlying high computational complexities [6]. Thus, non-standard reasoning solutions like approximate reasoning [7,8] are helpful in time-critical applications when it is acceptable to sacrifice soundness or completeness for increased efficiency. Approximate reasoning algorithms can be tractable although the underlying language is not, in contrast to limiting attention only to inexpressive tractable fragments as e.g. [9].

---

\* Research reported in this paper was supported by the German Federal Ministry Economics (BMWi) under the Theseus project, <http://theseus-programm.de> and EU project SEALS (FP7-ICT-238975), <http://www.seals-project.eu/>

Investigations into approximate reasoning usually start from a sound and complete algorithm and system and directly addresses performance bottlenecks in order to improve efficiency, i.e. the algorithms are altered, leading to approximate outputs, while improving speed and keeping the introduced error ratio as low as possible.

In previous work [10,11] we have shown how to approximate instance retrieval for named classes within the KAON2 approach. In this paper we show how instance retrieval for complex classes can be approximated by reducing it to instance retrieval for named classes. For this, we compute what we call *approximate extensions* of complex classes by means of combining extensions of named classes, e.g. by using standard database operations. The approach leads to a speedup of about factor 10, while the number of introduced errors varies depending on the query, but is within reasonable bounds.

The present paper is structured as follows. After recalling necessary preliminaries, we will present our approach to approximate instance retrieval, and then describe our approximate algorithms. We conclude by reporting on corresponding evaluation results and with ideas for further work. For a more elaborate version of this paper we refer to our technical report [16].

## 2 Preliminaries

**Description Logics.** Description logics (DLs) are a family of knowledge representation formalisms. The basic constituents to represent knowledge in DLs are *concepts*  $C$ , *roles*  $r$  and *individuals*  $a$ . They are used to form *axioms* collected in a *knowledge base*  $KB$  to make statements about a domain of interest. We primarily consider the DL *SHIQ* with concept and role assertion axioms of the form  $C(a)$ ,  $r(a, b)$  that assign an individual to a concept or relate two individuals via a role, and concept inclusion axioms of the form  $C \sqsubseteq D$  that state subclass relationships. For a detailed presentation of DLs we refer to [12]. The *signature* of a knowledge base  $KB$ , denoted by  $\sigma(KB)$ , is the set of all individual, concept and role names that occur in the axioms within  $KB$ . In particular,  $\sigma(KB)$  comprises all individuals occurring in  $KB$ .

Instance retrieval with DL knowledge bases builds on the standard reasoning task of instance checking. An individual  $a \in \sigma(KB)$  is an instance of a concept  $C$  with respect to a knowledge base  $KB$  if the axiom  $C(a)$  is a logical consequence of  $KB$ , which is denoted by  $KB \models C(a)$ . Instance retrieval can be interpreted as the repeated application of instance checking for all known individuals of  $KB$  and a given concept. We call the result of retrieving all instances of concept  $C$  from  $KB$  the (*conventional*) *extension* of  $C$  with respect to  $KB$ , denoted by  $|C|$ , and define it as follows.<sup>1</sup>

$$|C| := \{x \in \sigma(KB) \mid KB \models C(x)\}$$

In the context of instance retrieval, the concept  $C$  is often also called the *query*.

<sup>1</sup> Notice that this notion of extension refers to a particular knowledge base and is different from the model-theoretic notion of extension defined for an interpretation.

**Relational Algebra.** Relational Algebra is the formal underpinning of modern relational database systems and is used to formalise database operations on the relational model originally introduced by Codd [13]. The main construct for representing data in the relational model is a *relation*, denoted by  $R(a_1, \dots, a_n)$ , that represents a database table with column attributes  $a_1$  to  $a_n$  and rows that instantiate the columns as tuples of values. Relational algebra expressions are used to formulate queries on the thus represented database tables and result themselves in relations, such that expressions can be nested. Attributes in a relation can be referred to by means of path expressions of the form  $R.a_i$ , e.g. within conditions.

We briefly recall the relational operators that are used in this paper. A *projection*  $\pi_{[a_1, \dots, a_m]}(R(a_1, \dots, a_n))$  restricts the columns of the resulting relation to the attributes  $a_1, \dots, a_m$  for  $m < n$ . A *selection*  $\sigma_{[\text{condition}]}(R(a_1, \dots, a_n))$  selects those rows for which *condition* holds. A *cross product*  $R_1(a_1, \dots, a_n) \times R_2(b_1, \dots, b_m)$  generates a combined relation  $R(a_1, \dots, a_n, b_1, \dots, b_m)$  in the sense of the Cartesian product by multiplying rows, which is used for join operations. Other set operations are used for relations as usual, namely *union*  $R_1 \cup R_2$ , *intersection*  $R_1 \cap R_2$  and *difference*  $R_1 \setminus R_2$ , operating on relation tuples in the usual way. For a detailed description of relational algebra see e.g. [14].

### 3 Approximation of Instance Retrieval

Our approach for the approximation of instance retrieval queries is based on the notion of the *approximate extension*  $\langle C \rangle$  of a concept  $C$  with respect to a knowledge base  $KB$ . Intuitively,  $\langle C \rangle$  is the set of instances that are obtained through interpreting complex concepts in  $C$  as simple set operations on the individuals known to  $KB$ , starting from the atomic extensions of concepts and roles that occur in  $C$ . In this way, the model-theoretic semantics of DLs is approximated by a straightforward combination of results for atomic queries that requires less effort to compute than the reasoning process for complex instance retrieval queries in DLs does. The exact definition of an approximate extension is given in Table 1 recursively for all language constructs. For an example, consider the knowledge base  $KB = \{C \sqsubseteq A \sqcup B, A(a_1), C(a_2)\}$  and the instance retrieval query  $A \sqcup B$ . The conventional extension of the concept  $A \sqcup B$  contains both individuals  $a_1$  and  $a_2$ , i.e.  $|A \sqcup B| = \{a_1, a_2\}$ . However, the approximate extension of  $A \sqcup B$  contains only  $a_1$ , i.e.  $\langle A \sqcup B \rangle = \{a_1\}$ .

The more complex the query concept  $C$  is, the more the approximate extension deviates from the conventional extension. For the simplest queries, such as atomic concepts, the two types of extensions coincide and no errors are made in instance retrieval. This characteristic is captured by the following proposition.

**Proposition 1 (soundness and completeness of simple approximate extensions).** *For a knowledge base  $KB$  and a concept  $C$  of the form  $C = A_1 \sqcap \dots \sqcap A_m \sqcap \neg B_1 \sqcap \dots \neg B_n$ , with all  $A_i$  and  $B_j$  atomic, the approximate extension of  $C$  is equivalent to its conventional extension, i.e.  $\langle C \rangle = |C|$ .*



**Table 1.** Definition of an approximate extension.  $A$  stands for atomic classes and  $C, D$  for complex (non-atomic) classes.  $R$  stands for roles and  $n$  for a natural number.

Approximate Extensions	
$\langle \top \rangle =$	$ \top $
$\langle \perp \rangle =$	$\emptyset$
$\langle A \rangle =$	$ A $
$\langle \neg A \rangle =$	$ \neg A $
$\langle R \rangle =$	$\{(x, y) \mid KB \models r(x, y)\}$
$\langle R^- \rangle =$	$\{(x, y) \mid KB \models r(y, x)\}$
$\langle C \sqcap D \rangle =$	$\langle C \rangle \cap \langle D \rangle$
$\langle C \sqcup D \rangle =$	$\langle C \rangle \cup \langle D \rangle$
$\langle \neg C \rangle =$	$\langle \top \rangle \setminus \langle C \rangle$
$\langle \exists R.C \rangle =$	$\{x \in \langle \top \rangle \mid \exists y : (x, y) \in \langle r \rangle \wedge y \in \langle C \rangle\}$
$\langle \forall R.C \rangle =$	$\{x \in \langle \top \rangle \mid \forall y : (x, y) \in \langle r \rangle \rightarrow y \in \langle C \rangle\}$
$\langle \leq n R.C \rangle =$	$\{x \in \langle \top \rangle \mid \#\{y \mid (x, y) \in \langle r \rangle \wedge y \in \langle C \rangle\} \leq n\}$
$\langle \geq n R.C \rangle =$	$\{x \in \langle \top \rangle \mid \#\{y \mid (x, y) \in \langle r \rangle \wedge y \in \langle C \rangle\} \geq n\}$

Proposition 1 states that, for queries that have the form of conjunctions of possibly negated named concepts, the approximate and conventional extensions have exactly the same instances. In other words, computing the approximate extension is *sound and complete* with respect to the conventional extension.

For more complex queries, however, the approximation might deviate significantly from the correct answer in both that it might miss instances as well as show improper instances. In particular the approximation of the complement constructor is supposed to cause significant deviation as it interprets negation in a closed-world sense, potentially including improper instances in an answer. Hence, we aim at eliminating general complements by means of normalisation, avoiding this source of error.

For standard reasoning in DLs a query concept can be expressed in various normal forms and semantics-preserving transformations do not affect the result of instance retrieval. For the calculation of approximate extensions, however, the result depends on the form of the concept, and different semantically equivalent concept expressions can have different approximate extensions. We can exploit this characteristics by choosing a normal form for query concepts that fits best the process of approximation in terms of both error rate and ease of computation. In this light, we consider the negation normal form [15] of concept expressions for queries, denoted by  $NNF(C)$  for a concept  $C$ , in which negation symbols are pushed inside to occur only in front of atomic concepts. This eliminates the case of considering the approximation for general complements with its rather drastic closed-world interpretation. Besides the lower expected error rate this also avoids the computationally costly handling of large sets of individuals in case of large ABoxes by an algorithm that computes approximate extensions. The positive effect that elimination of complement approximation has on the error rate in instance retrieval can be expressed by the following property, which ensures that

approximation of concepts in negation normal form only gives up completeness but preserves soundness at least for a certain class of queries.

**Proposition 2 (soundness of limited approximate instance retrieval).** *Let  $KB$  be a knowledge base and  $C$  be a concept such that  $NNF(C)$  contains no  $\forall$ - and no  $\leq$ - and  $\geq$ -constructs. The approximate extension of  $NNF(C)$  only contains instances that are also contained in the conventional extension of  $C$  with respect to  $KB$ , i.e.  $\langle NNF(C) \rangle \subseteq |C|$ .*

Proposition 2 states that, for queries that do not make use of the  $\forall$ ,  $\geq$  and  $\leq$  constructs (after normalisation), the approach of approximating concepts in their negation normal form yields an extension that might miss some instances but has no improper instances in it. In other words, computing the approximate extension is *sound* with respect to the conventional extension.

## 4 Computing Approximate Extensions

In this section, we will present algorithms for computing the approximate extension of a query concept. We will lay out the architecture of a system for approximate instance retrieval and elaborate on two implementations of the algorithms, one in a database and one in memory.

### 4.1 System Architecture

Our system for approximate instance retrieval takes as input a  $SHIQ$  knowledge base  $KB$  and a complex query concept  $Q$  to compute the approximate extension of  $Q$  with respect to  $KB$  as a set of individuals. This is depicted on the right-hand side of Figure 1. The principle behind computing  $\langle Q \rangle$  is always to start from the individuals in the conventional extensions of (possibly negated) atomic concepts and (possibly inverse) roles that occur in  $Q$  and to recursively combine these according to the structure of concepts in  $Q$ , reflecting the set operations from Table 1. According to Propositions 1 and 2, this results in an answer that is sound and complete for some cases, only sound for others, or neither sound nor complete, depending on the language constructs used in the query.

We have implemented the approximate instance retrieval method in two different ways and distinguish between *database* and *in-memory* computation: in the first case computation is delegated to underlying database operations, whereas in the second case it is performed in main memory. While for the database variant the atomic extensions are pre-computed prior to query-time and materialised in the database using a sound and complete reasoner, the in-memory variant allows for two possibilities to access the atomic extensions: in *online processing* a sound and complete DL reasoner is invoked at query-time to compute atomic

<sup>2</sup> We use  $SHIQ$  since we build on KAON2 for our experimental results. However, our approximation approach can easily be extended to nominals, the missing feature for handling OWL ontologies.

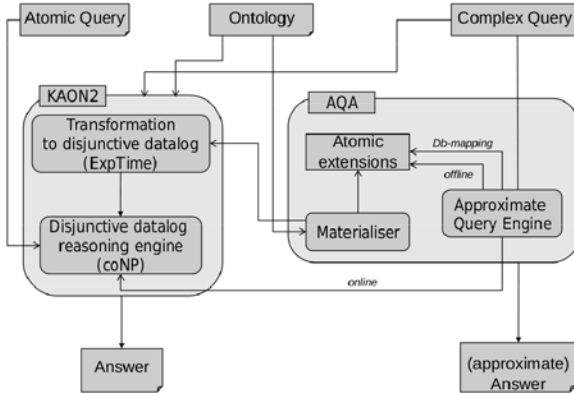


Fig. 1. An overview of the system architecture

extensions, while in *offline processing* they are again pre-computed and materialised either in a database or in memory if possible. Database computation and offline processing are very useful when dealing with large amounts of data in scenarios with frequent querying on rather static ontologies, for which materialisation can be done in advance. Online processing is intended to be used in such cases where a materialisation is hardly manageable as ontologies are subject to frequent changes.

For online processing we utilise the KAON2 reasoner, as illustrated in Figure 1. The reason for this choice is that KAON2 was designed to be an efficient ABox reasoner on knowledge bases with large ABoxes and simple TBoxes in comparison to other state-of-art DL reasoners, which typically perform better on knowledge bases with large (or complex) TBoxes and small ABoxes. As depicted on the left-hand side of Figure 1, KAON2 transforms the TBox together with complex queries into a disjunctive datalog program in a first step, to perform ABox reasoning in a second step based on the result of this transformation. Hence, for every complex ABox query KAON2 needs to repeatedly perform the TBox transformation, which is computationally costly. For ABox queries that have the form of atomic concepts, however, this transformation is not necessary and can be bypassed. For the variant with in-memory and online processing we can take advantage of this because for computing atomic extensions with KAON2 the costly TBox translation is saved.

## 4.2 Delegation of Computation to Database

The variant that performs database computation is a presumably efficient implementation of approximate instance retrieval as the pre-computed atomic extensions are materialised and approximate extensions are computed by making use of highly optimised database operations. This variant is essential in practice for handling ontologies with large ABoxes that cannot be processed efficiently in memory. Here, the recursive combination of atomic extensions in terms of

**Table 2.** Mapping of DL concept expression to Relational Algebra Expression

<i>Concept Expression</i>	<i>Relational Algebra Expression</i>
$\tau_{db}(A)$	$\pi_{[ind]}(\sigma_{[class=A]}(\text{Ext}^C))$
$\tau_{db}(\neg A)$	$\pi_{[ind]}(\sigma_{[class=\neg A]}(\text{Ext}^C))$
$\tau_{db}(\exists r.C)$	$E_C := \tau_{db}(C)$ $E_r := \sigma_{[role=r]}(\text{Ext}^r)$ $\pi_{[ind_1]}(\sigma_{[ind=ind_2]}(E_C \times E_r))$
$\tau_{db}(\forall r.C)$	$E_C := \tau_{db}(C)$ $E_r := \sigma_{[role=r]}(\text{Ext}^r)$ $E_- := \pi_{[ind_1]}(\sigma_{[ind \neq ind_2]}(E_C \times E_r))$ $\pi_{[ind_1]}(\text{Ext}^C) \setminus E_-$
$\tau_{db}(\leq n R.C)$	$E_C := \tau_{db}(C)$ $E_r := \sigma_{[role=r]}(\text{Ext}^r)$ $\pi_{[ind]}(\sigma_{[count(ind_1) \leq n \wedge ind=ind_2]}(E_C \times E_r))$
$\tau_{db}(\geq n R.C)$	$E_C := \tau_{db}(C)$ $E_r := \sigma_{[role=r]}(\text{Ext}^r)$ $\pi_{[ind]}(\sigma_{[count(ind_1) \geq n \wedge ind=ind_2]}(E_C \times E_r))$
$\tau_{db}(C_0 \sqcap C_1 \sqcap \dots \sqcap C_n)$	$\tau_{db}(C_0) \cap \tau_{db}(C_1) \cap \dots \cap \tau_{db}(C_n)$
$\tau_{db}(C_0 \sqcup C_1 \sqcup \dots \sqcup C_n)$	$\tau_{db}(C_0) \cup \tau_{db}(C_1) \cup \dots \cup \tau_{db}(C_n)$

set operations as defined in Table 1 is completely delegated to the underlying database, which benefits performance. As a basis for this form of computation we use a database schema that consists of two Relations, namely  $\text{Ext}^C(ind, class)$  for storing concept extensions and  $\text{Ext}^r(ind_1, role, ind_2)$  for storing role extensions. In their schema, the attribute  $ind_{(i)}$  stands for individual names,  $class$  for names of possibly negated concepts and  $role$  for names of possibly inverse roles. Starting from a knowledge base  $KB$ , these two relations are initialised as follows.

$$\text{Ext}^C(ind, class) = \{(a, C) \mid KB \models C(a)\}, \text{ for } C = A \mid \neg A \text{ with } A \in \sigma(KB)$$

$$\text{Ext}^r(ind_1, role, ind_2) = \{(a, r, b) \mid KB \models r(a, b)\}, \text{ for } r = p \mid p^- \text{ with } p \in \sigma(KB)$$

Notice that, for the purpose of approximate instance retrieval,  $\text{Ext}^C$  and  $\text{Ext}^r$  form a complete representation of the original knowledge base  $KB$ .

A complex query concept  $Q$  is answered by transforming its negation normal form  $\text{NNF}(Q)$  into a relational algebra expression according to a mapping  $\tau_{db}$  and posed as a query to the underlying database system. The complete mapping definition for  $\tau_{db}$  is given in Table 2. The left-hand side shows the concept constructors that can occur in  $\text{NNF}(Q)$  and the right-hand side shows their respective relational algebra expression. Recursive application of  $\tau_{db}$  ultimately produces a single database query  $\tau_{db}(\text{NNF}(Q))$  that is used for computing  $\langle Q \rangle$ .

For an example consider the query  $Q = A \sqcap \exists r. \neg B$ . The mapping  $\tau_{db}$  produces the following nested relational algebra expression.

$$\tau_{db}(Q) = \pi_{[ind]}(\sigma_{[class=A]}(\text{Ext}^C)) \cap \pi_{[ind_1]}(\sigma_{[ind=ind_2]}(\sigma_{[role=r]}(\text{Ext}^r) \times \pi_{[ind]}(\sigma_{[class=\neg B]}(\text{Ext}^C)))) .$$

When posed to the underlying database, this rather large expression is subject to efficient internal query optimisation strategies as they are typically employed by database systems.

### 4.3 In-Memory Computation

Both the variants with online and offline processing share the same implementation of the approximate algorithm. The difference is the handling of atomic extensions which is presented by an additional function. This function takes as parameters a knowledge base and an atomic concept or atomic role for which the atomic extension is to be computed while the actual algorithm accepts the knowledge base and a complex concept query for which the approximate extension is to be computed. For the computation of the atomic extension, depending on the chosen variant, the introduced function invokes either a complete and sound reasoner or retrieves the atomic extension from the database. For the exact details of this algorithm, the interested reader may refer to [16].

## 5 Conclusion

In our experiments, using the WINE ontology, which has been designed as a showcase for the expressivity of OWL, we compared our algorithms with KAON2 as a sound and complete DL reasoner. Running the approximation algorithm in the database variant, we obtained a significant performance improvement for each  $\exists$ -query<sup>3</sup> about 90%. Running the algorithm in offline processing where the approximation is computed in memory, we obtained another significant performance gain, indeed about 99% compared to KAON2. For the details of our experiments including complex queries, the interested reader may refer to [16].

We have presented an approach to approximate instance retrieval based on approximate extensions. Compared with a complete and sound DL reasoner, our approach can significantly improve the performance of reasoning over expressive ontologies with large ABoxes and TBoxes. We presented several instantiations of our approach resulting online and offline in-memory and database variants. We evaluated the approaches and showed that a significant speed-up of around 90% can be obtained while the number of introduced errors remains relatively small.

Future work includes improvements on the online variant using logic programming engines, further experiments for complex queries, combinations with other approximate reasoning methods, extension to more expressive language features and applications of our approach in suitable use case scenarios.

## References

1. Schmidt-Schauß, M., Smolka, G.: Attributive Concept Descriptions with Complements. *Artificial Intelligence* 48(1), 1–26 (1991)

---

<sup>3</sup> Queries of the form  $\exists r.A$  where  $A$  is a named class and  $r$  is a role.

2. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. In: *Web Semantics: Science, Services and Agents on the World Wide Web*, pp. 51–53 (2007)
3. Haarslev, V., Möller, R.: Racer System Description. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) *IJCAR 2001*. LNCS (LNAI), vol. 2083, pp. 701–706. Springer, Heidelberg (2001)
4. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Universität Karlsruhe (2006)
5. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM – A Pragmatic Semantic Repository for OWL. In: Dean, M., Guo, Y., Jun, W., Kaschek, R., Krishnaswamy, S., Pan, Z., Sheng, Q.Z. (eds.) *WISE 2005 Workshops*. LNCS, vol. 3807, pp. 182–192. Springer, Heidelberg (2005)
6. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. PhD thesis, RWTH Aachen, Germany (2001)
7. Fensel, D., van Harmelen, F.: Unifying reasoning and search to web scale. *IEEE Internet Computing* 11(2), 94–95 (2007)
8. Rudolph, S., Tserendorj, T., Hitzler, P.: What is Approximate Reasoning? In: Calvanese, D., Lausen, G. (eds.) *RR 2008*. LNCS, vol. 5341, pp. 150–164. Springer, Heidelberg (2008)
9. Baader, F., Brandt, S., Lutz, C.: Pushing the  $\mathcal{EL}$  Envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK*, pp. 364–369. Professional Book Center (2005)
10. Hitzler, P., Vrandečić, D.: Resolution-Based Approximate Reasoning for OWL DL. In: Gil, Y., et al. (eds.) *ISWC 2005*. LNCS, vol. 3729, pp. 383–397. Springer, Heidelberg (2005)
11. Tserendorj, T., Rudolph, S., Krötzsch, M., Hitzler, P.: Approximate OWL-Reasoning with Screech. In: Calvanese, D., Lausen, G. (eds.) *RR 2008*. LNCS, vol. 5341, pp. 165–180. Springer, Heidelberg (2008)
12. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge (2007)
13. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. *ACM Commun.* 26(1), 64–69 (1983)
14. Ramakrishnan, R., Gehrke, J.: *Database Management Systems*. Osborne/McGraw-Hill, Berkeley (2000)
15. Schmidt-Schauß, M., Smolka, G.: Attributive Concept Descriptions with Complements. *Journal of Artificial Intelligence* 48(1), 1–26 (1991)
16. Tserendorj, T., Grimm, S., Hitzler, P.: Approximate Instance Retrieval on Ontologies. Technical report, Kno.e.sis Center, Wright State University, Dayton, Ohio (2010), <http://knoesis.wright.edu/faculty/pascal/resources/publications/aqa2010%.pdf>

# Removing the Redundancy from Distributed Semantic Web Data

Ahmad Ali Iqbal<sup>1,2</sup>, Maximilian Ott<sup>2</sup>, and Aruna Seneviratne<sup>1,2</sup>

<sup>1</sup> School of EE&T, University of New South Wales (UNSW), Australia

<sup>2</sup> National Information and Communication Technology Australia (NICTA)

**Abstract.** Peer-to-peer databases have proven to be an effective way for sharing data. However, distributed knowledge management in P2P databases brings a variety of non-trivial challenges along with its benefits. Such challenges include determining the right content provider(s) and removing the duplicate data transfer if a relatively larger portion of data is redundant and is made available in distributed providers. The aim of this paper is to address data redundancy removal problem such that excessive bandwidth usage due to in-network duplicate data transfer can be minimized. We provide analytical and experimental evaluation of our schemes in terms of the number and size of the packets that flow in the network while keeping confidence level of results high.

## 1 Introduction

As the tremendous growth of peer-to-peer networks for file sharing continues to strain, research community continues to focus on its use for knowledge sharing [4] in order to exploit its further benefits. P2P databases like Xpeer [7] and AmbientDB [3] emerge as a way for retrieving data from an unstructured and decentralized P2P network and overcome the limitations posed in distributed databases by eliminating constraints like static topology and heavy administration requirements. Such databases have diverse information stores to serve the individual clients and thus, these stores are often not well integrated with each other and they collect the data in uncooperative environment. Due to having no control over information stores, duplicate data storage across multiple stores becomes a serious problem for the clients who search through these stores and costs for redundant data transfer.

These systems demand the following two core operations (i) discovery of information stores for a given query also known as *resource selection* and (ii) redundancy removal in order to minimize bandwidth usage while transferring the data between peers. Resource selection algorithm determines the most suitable information stores for a query provided as an input and gives the ranked list of the candidates. Resource selection [4][5][8] has been under the focus in database community since last decade and is out of scope of this paper. The aim of this paper is to study data redundancy removal problem when it is made available from distributed Resource Description Framework (RDF) based information stores. In order to achieve this, we present summary exchange algorithms

for coordination between peers that employ bloom filter. They are widely used in variety of database and networking applications [1] to minimize the size of message exchange at the cost of compromising confidence level. Due to the space limitation, we omit the operation of bloom filter and refer [1][2] to the interested readers.

Next, we define *confidence level* (CL) as accuracy of results from distributed stores. Formally, it is defined as a function of  $1 - FP_{avg}$  where  $FP_{avg}$  is the average of false positive results from the distributed information systems for a given query. We define *average response time* (ART) as the delay between the query generation and final result preparation by the information receiver after merging the results received from distributed information stores. *Selection Set* (SS) is a special type of message used by information receiver to inform the information stores about the selection of element that receiver expects to receive.

Rest of this paper is structured as follows; Redundancy removal schemes are presented in section 2. In section 3, we compare the performance of these schemes. Finally, we conclude this paper in section 4.

## 2 Duplication Removal

In order to reduce the duplicate data transmission on the network, we present 3 flavors of summary exchange algorithms and discuss their improvement in comparison with brute-force approach. The prerequisite of these algorithms is a *resource selection algorithm* [5] that gives the ranked list of potential information stores.

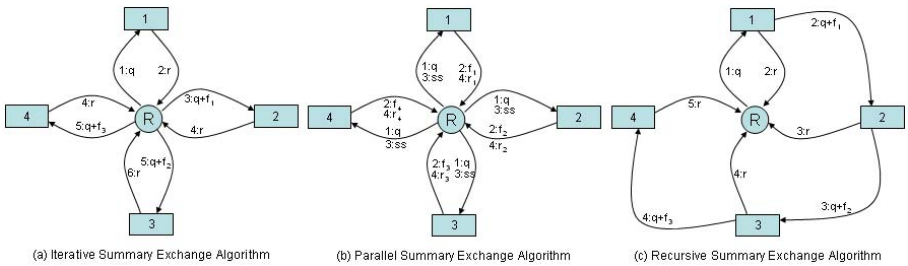


Fig. 1. Flow of Messages in ISE, PSE and RSE algorithms

### 2.1 Iterative Summary Exchange (ISE) Algorithm

This is an evolutionary algorithm where information receiver (R) monotonically receives the set of RDF statements from multiple information stores. In this algorithm, information receiver iteratively approaches the information stores and gets part of relevant results to a query as follows;

1. Information receiver sends the query (q) to the information store that was ranked highest in the list of information stores for q.



2. Information store executes the query onto its repository and send the matching RDF statements ( $r$ ) to information receiver.
3. Information receiver creates  $m$  bit bloom filter on the results it has received for query  $q$  using  $k$  hash functions such that  $k$  hashes are created on each statement of the result set and relevant bits are marked in the bloom filter.
4. Information receiver sends the query, bloom filter ( $q + f_n$  refers to both) and configuration parameters (CP) to information store ranked next in the ranking list for a query  $q$ . CP defines all those parameters which are used to create the bloom filter like size, number and type of hash functions.
5. Information store executes the query to retrieve the results.
6. Despite transmitting all matching RDF statements like highest ranked information store did, this store filters out the results indicated by the bloom filter and sends only the set of those statements which were not marked in the received bloom filter. For testing the membership of statements, information store uses CP received from information receiver (see step 4).
7. Step 3 to 6 repeat for the rest of the information stores within the list of potential candidates for a given  $q$ .
8. Finally, information receiver combines all the received RDF statements in order to prepare final result for  $q$ .

The message flow between information receiver and distributed stores is shown in figure 1a. Suppose that there are  $l$  information stores that contain the data matching a query  $q$  and  $n_i$  is the number of RDF statements received by R after iteration  $i$ , each iteration is a combination of querying and receiving the results from exactly one information store and the number of bloom filters created by ISE algorithm is  $l - 1$ . Then confidence level for a given query  $q$  when ISE algorithm is used can be calculated by

$$CL_q := 1 - \frac{\sum_{i=0}^{l-1} \left[ 1 - \left( 1 - \frac{1}{m} \right)^{k \cdot n_i} \right]^k}{l - 1}. \quad (1)$$

## 2.2 Parallel Summary Exchange (PSE) Algorithm

In PSE algorithms, individual information stores evolve independently of their repositories from their peers thus query execution for creating and membership testing of RDF statements is straightforward and thus decreases the waiting time. In this algorithm, receiver's query is executed in parallel on selected information stores as follows;

1. Information receiver multicast the query to all information stores selected by a resource selection algorithm.
2. Those stores create bloom filter on the matching RDF statements to the receivers query.
3. Information stores send those filters to the receiver.
4. Information receiver distinguishes between corresponding set bits of the all received bloom filters for unique RDF statements and duplicate RDF statements by performing *AND* operation and chooses the candidate information store for each duplicate RDF statement.

5. Selection of RDF statements that each information store will provide is informed by information receiver.
6. Each information store provides the RDF statements according to the selection notified by information receiver

**Explanation.** PSE algorithm needs more message passing and processing from bloom filter as compared to ISE algorithm as shown in figure 1b. It is because the receiver has to decide the information store(s) for overlapping RDF statements based upon their corresponding bloom filters. In order to inform the selection of contents, IR transmits SS message to the ISs where  $SS \leq bloomfiltersize$ .

Suppose that there are  $l$  information stores that contain RDF statements matching a query  $q$  and unlike in equation 1,  $n_i$  is the number of RDF statements matching a query  $q$  in  $i$ th information store and total bloom filters created in PSE algorithm are  $l$ . Then  $CL_q$  for a  $q$  can be calculated by

$$CL_q := 1 - \frac{\sum_{i=0}^{l-1} \left[ 1 - \left( 1 - \frac{1}{m} \right)^{k \cdot n_i} \right]^k + C_i}{l} \tag{2}$$

where  $C_i$  is the constant boosting the false positive probability due to multiple hashing algorithms used to create the bloom filter. The ground truth about the bloom filter is that it uses more than one hashing algorithms and they are not perfect thus generate false positives. The problem with PSE when more than one hashing algorithms are used is that the bit set by one hashing algorithm for one element in one IS may overlap with the corresponding bit of another hashing algorithm for another element in another IS. This problem is known as *bit conflict* that further supplements false positive probability. In order to deal with this issue, we partitioned the total bloom filter space into  $k$  subspaces such that each hashing algorithm uses only its assigned subspace. This way the scope for each hashing algorithm in a bloom filter becomes independent from others.

### 2.3 Recursive Summary Exchange (RSE) Algorithm

The basic idea of RSE algorithm is that ISs gossip (negotiate) among themselves in order to coordinate and reduce the redundant data transfer cost. Flow of messages between information stores and receiver is shown in figure 1c. The algorithm works as follows;

1. Information receivers unicast a query (q) along with configuration parameters (CP) to IS that has the highest priority for a given query  $q$ . As mentioned earlier priority list of IS for a specific query  $q$  is the output of *resource selection algorithm* that runs in advance for each query. CP contains the size of bloom filter (m), number (k) and selection of hashing algorithms.
2. The information store with the highest priority executes the query q and retrieve the matching RDF statements from its repository.
3. Information store creates the bloom filter on the retrieved RDF statements using CP.

4. Information store removes itself from the priority list and sends those statements to the receiver and bloom filter along with CP to the next information store in the priority list.
5. Despite sending all RDF statements as the highest ranked information stores do, this store filters out the results indicated by the bloom filter and send only the set of those statements which were not marked by the information receiver. For testing the membership of statements with the bloom filter, information store uses CP received from information receiver (see step 1).
6. This information store updates the previously received bloom filter by adding the entries for RDF statements retrieved from its local repository. For this sake, it uses the CP received from information receiver.
7. Step 4 to 6 repeat until all the IS are removed from the priority list.

In order to formalize confidence level of the system, assume that there are  $l$  information stores that contain RDF statements matching a query  $q$  and like equation 2,  $n_i$  is the number of RDF statements matching a query  $q$  in  $i$ th information store then  $CL_q$  can be calculated by

$$CL_q := 1 - \frac{\sum_{i=0}^{l-1} \left[ 1 - \left( 1 - \frac{1}{m} \right)^{k \cdot n_i} \right]^k}{l - 1}. \quad (3)$$

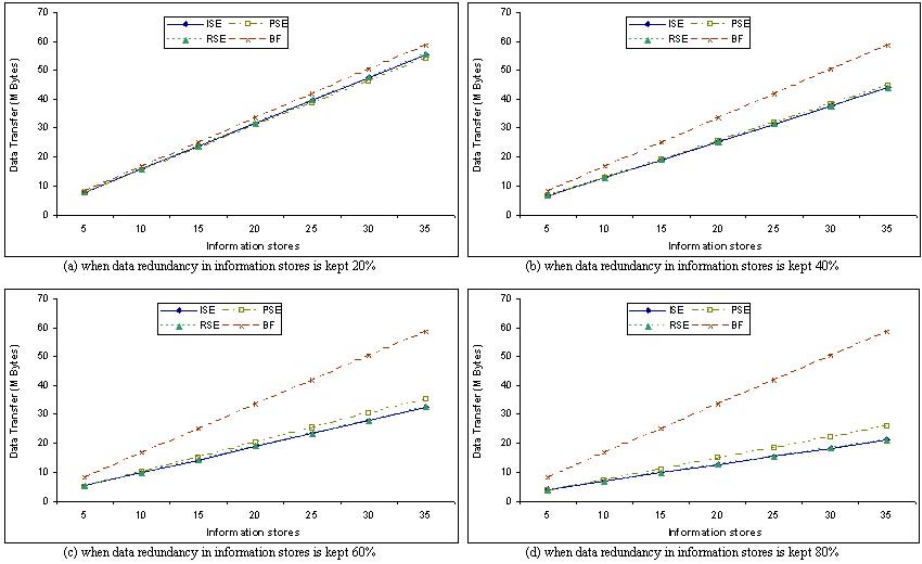
## 2.4 Discussion

These algorithms have their own pros and cons. ISE algorithm is more simplistic while it is suitable for applications where ART is a trivial factor. On the other hand, in PSE and RSE algorithms, participating information stores process queries faster thus reduces latency for end users. In PSE algorithm, participatory information stores process the query in parallel and receiver decides the portion of information it receives from but in RSE algorithm, information stores negotiate with each other thus they have more control on the information that they provide than receiver. In view point of false positive probability, RSE algorithm outperforms ISE and PSE algorithms because each RDF statement distributed across  $l$  information stores for a query  $q$  is used only once for creating a bloom filter (see equation 3) and there is no additional probability. In contrast, PSE algorithm generates higher false positive probability in comparison to ISE and RSE because of the additional false positive (see equation 2).

## 3 Evaluation

Although, there is no other published work on distributed ontology based redundancy removal to be compared with ours, packet and message level redundancy removal for pub/sub system has been discussed for years. We use the following performance metrics: ART, message exchanges, data transfer cost and confidence level of results. This experiment is setup on native simulator<sup>1</sup> developed in Java

<sup>1</sup> <http://ee.unsw.edu.au/~z3197878/simulator/>

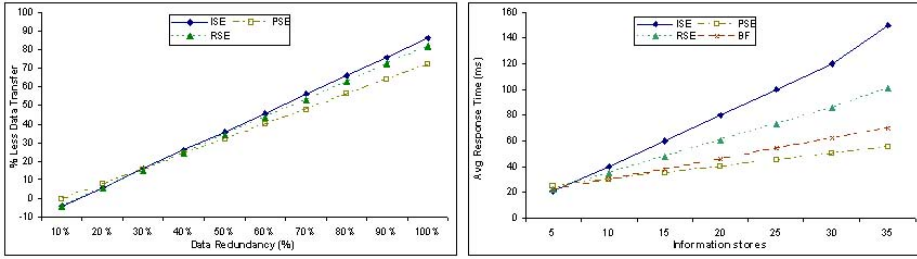


**Fig. 2.** Data transfer in distributed information retrieval when varying the data redundancy in ISs

and the simulation is developed using a grid topology of  $20 \times 20 = 400$  nodes. We transfer  $n$  times 104858 elements of size 128 bits each where  $n$  is the number of ISs that varies from 5 to 35 with the increment of 5. We used 9 hash functions (i.e.  $k=9$ ) using [6] for the bloom filter and kept bloom filter size ( $m$ )  $2 * n * k$ . The reason for choosing this  $m$  is to achieve upto 99% confidence level of results as stated in [2]. We kept the size of SS message equal to  $m$  for the sake of simplicity, however, it can be compressed further for further reduction in overhead traffic. Furthermore, we set 20% data redundancy between IS contents where it is not explicitly stated otherwise.

Figure 2 compares our redundancy removal algorithms in comparison with brute-force approach in the presence of 20, 40, 60 and 80 % data redundancy in ISs. In order to answer receiver’s query, total data transfer by varying the number of ISs is shown in figure 2. It is clear that after a certain threshold percentage (reasonably small value) in data redundancy, summary exchange algorithms outperforms over brute-force and once this threshold is achieved, improvement in terms of data transfer becomes proportional to the percentage of redundant data in distributed ISs as shown in figure 3(a). Please note that our experiments are based on 20% data redundancy except the ones shown in figure 2 and 3(a).

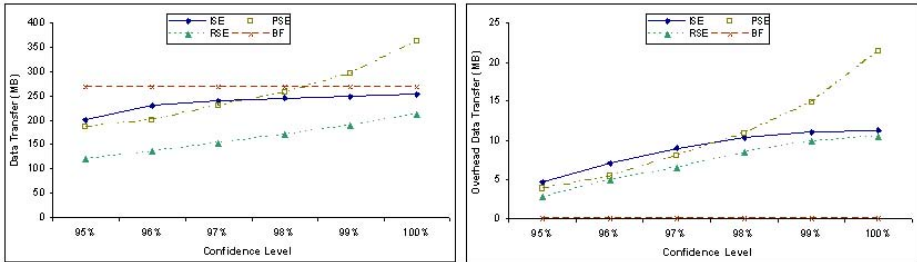
ART for ISE as shown in figure 3(b), is more that other algorithms because IR queries ISs one after the other and waits for the response from one before sending it to another. In contrast, PSE algorithm outperforms others in terms of ART because query in a set of ISs is executed in parallel and receiver simultaneously gets the redundancy removed RDF statements. The reason that PSE gets better response time over brute-force is due to (i) lesser data transfer



**Fig. 3.** (a) Summary of improvement in terms of data transfer reduction with redundancy removal algorithms in comparison to brute-force approach, (b) Effect on ART by varying the number of distributed ISs

and (ii) no redundancy removal by the receiver during merging time after receiving from distributed ISs. On the other hand, ART for RSE is slightly less than ISE because ISs in ISE are contacted sequentially by the IR while in RSE ISs directly communicate with each other. This certainly saves waiting time for message exchange between ISs and IR for RSE.

Figure 4 shows the total data transfer trend in order to achieve confidence level from 95% to 100%. False positive in this setting yields in missing the data which should actually be delivered due to matching the receiver’s query. Thus, in order to get higher confidence level as depicted in figure 4, larger size of the bloom filters are required to transfer, causing more data transfer in terms of overhead traffic. In case of PSE, overhead traffic exponentially increases because in addition to regular false positive generated by the bloom filter, it is further supplemented during partitioning mechanism.



**Fig. 4.** To achieve higher confidence level, more overhead data have to transfer and it makes significant increase on total data transfer

## 4 Conclusion

This paper develops and analyzes three redundancy removal algorithms from distributed semantic information stores. In contrast to brute-force, these schemes are not focused on finding the perfect solutions but rather on finding good enough

solutions. The algorithms reduce upto 80% data transfer cost in the presence of duplicate data in distributed information stores. Furthermore, these algorithms have different characteristics and can achieve higher confidence level of results at the cost of overhead traffic. In comparison to brute-force, reduction in data transfer cost in the presence of redundant data in distributed information stores is proportional to redundancy among the data in these stores and if data redundancy crosses the threshold (approx. 15%) percentage, cut in cost for data transfer can be achieved.

## Acknowledgments

NICTA is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

## References

1. Broder, A., Mitzenmacher, M.: Network applications of bloom filter: A survey. *Internet Mathematics* 1(4), 485–509 (2003)
2. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: A scalable wide-area web cache sharing protocol. *IEEE Transactions on Networks* 8(3) (2000)
3. Fontijn, W., Boncz, P.: Ambientdb: P2p data management middleware for ambient intelligence. In: *PERCOMW'04, USA* (2004)
4. Haase, P., Siebes, R., Harmelen, F.: Peer selection in peer-to-peer networks with semantic topologies. In: *International Conference on Semantics of a Networked World: Semantics for Grid Databases* (2004)
5. Iqbal, A., Ott, M., Seneviratne, A.: Resource selection from distributed semantic web stores. In: *Int. Conf. on Data and Knowledge Engineering* (2010)
6. Kirsch, A., Mitzenmacher, M.: Less hashing, same performance: Building a better bloom filter. In: *European Symposium on Algorithms* (2006)
7. Sartiani, C., Manghi, P., Ghelli, G., Conforti, G.: Xpeer: A self-organizing xml p2p database system. In: *Workshop on P2P and Databases* (2004)
8. Si, L., Callan, J.: Relevant document distribution estimation method for resource selection (2003)

# Author Index

- Abásolo, José I-46  
Adachi, Jun II-141  
Agrawal, Gagan II-64  
Ahmed, Usman II-159  
Andrzejewski, Witold II-315  
Appelrath, H. -Jürgen I-261  
Aramburu, María José I-62  
Asano, Yasuhito II-1  
Atallah, Mikhail II-300
- Banaei-Kashani, Farnoush I-432  
Banek, Marko I-470  
Bao, Zhifeng II-391  
Barreiro, Álvaro II-407  
Bača, Radim I-358  
Ben-Saad, Myriam I-1  
Berlanga, Rafael I-62  
Bernroider, Edward I-221  
Bertino, Elisa I-368, II-17  
Bhattacharya, Arnab II-149  
Bleja, Michał I-323  
Bolles, André I-261  
Bressan, Stéphane I-331  
Brezo, Felix I-213  
Bringas, Pablo G. I-213, II-185
- Ceci, Michelangelo II-470  
Chakraborty, Abhirup I-251  
Chang, Ya-Hui I-290  
Chao, Kun-Mao I-290  
Chaves, Marcirio I-495  
Chen, Arbee L.P. I-105  
Chen, Gang I-198  
Cheng, Jingwei I-487  
Chen, Hanxiang II-437  
Chen, Ke I-198  
Chen, Qiming I-306  
Chen, Yuexian I-385  
Choi, Sunoh I-368  
Chovanec, Peter I-358  
Ciaramella, Alessandro I-31  
Cimino, Mario G.C.A. I-31  
Cong, Gao II-267  
Conroy, Kenneth II-462
- Crolotte, Alain I-340  
Cung, Van-Dat I-46
- Dash, Ranjan I-269  
Debenham, John II-126  
Deliège, François I-137  
Demidova, Elena II-240  
Demiryurek, Ugur I-432  
Deng, Fan I-152  
De Nicola, Antonio I-76  
Desai, Aditya II-168  
Dieng, Cheikh Tidiane II-225  
Dobbie, Gillian II-391
- Eder, Johann I-315  
Eduardo Ares, M. II-407
- Fan, Gaofeng I-16  
Fegaras, Leonidas I-269, II-452  
Fischella, Marco I-152  
Flesca, Sergio II-285  
Fu, Junming I-385  
Fukagawa, Daiji II-141  
Furfaro, Filippo II-285  
Furuse, Kazutaka II-437
- Gançarski, Stéphane I-1  
Gao, Xiaofeng II-80  
Ge, Jiaqi II-485  
Ghazal, Ahmad I-340  
Ghinita, Gabriel I-368  
Gkoulalas-Divanis, Aris I-400  
Grawunder, Marco I-261  
Grimm, Stephan I-503  
Gruenwald, Le I-122  
Guisado-Gámez, Joan I-231  
Gu, Jun I-385  
Gu, Yu I-167  
Gunturi, Viswanath II-149  
Guraya, Teresa II-185
- Härder, Theo I-183  
He, Zhenying I-298  
Hitzler, Pascal I-503  
Hossain, Shahriyar I-349

- Hsu, Meichun I-306  
 Hsu, Wynne II-209  
 Hu, Hao I-298  
 Hu, Tianlei I-198  
 Hu, Wei I-198  
  
 Iftikhar, Nadeem II-111  
 Iqbal, Ahmad Ali I-512  
  
 Jacobi, Jonas I-261  
 Jamil, Hasan I-349  
 Javier Zarazaga-Soria, F. I-495  
 Jen, Tao-Yuan II-225  
 Jensen, Claus A. I-137  
 Jurić, Damir I-470  
  
 Khodaei, Ali I-450  
 Kim, Hea-Suk II-17  
 Kim, Sang-Pil II-17  
 Kitagawa, Hiroyuki II-437  
 Kitsuregawa, Masaru I-241  
 Koncilia, Christian I-315  
 Kowalski, Tomasz I-323  
 Krátký, Michal I-358  
 Krishna Reddy, P. II-194  
 Krogsgaard, Jan II-32  
 Kurasawa, Hisashi II-141  
  
 Larriba-Pey, Josep-Lluís I-231  
 Laurent, Dominique II-225  
 Leclère, Michel II-330  
 Lee, Dongjoo II-422  
 Lee, Mong Li II-209  
 Lee, Sang-goo II-346, II-422  
 Levine, David II-452  
 Li, Chen I-450  
 Li, Xuhui II-375  
 Ling, Tok Wang II-391  
 Lin, Rung-Ren I-290  
 Lin, Shukuan I-167  
 Liu, Jianquan II-437  
 Liu, Mengchi II-375  
 López-de-Uralde, Juan II-185  
 Lopez-Pellicer, Francisco J. I-495  
 Loukides, Grigorios I-400  
 Lukasik, Tomasz II-177  
 Luo, Min II-49  
  
 Ma, Z.M. I-16, I-478, I-487  
 Maeda, Akira II-248  
  
 Marcelloni, Francesco I-31  
 Masegla, Florent I-91  
 Middelfart, Morten II-32  
 Mierzwa, Marek II-177  
 Miquel, Maryvonne II-159  
 Missikoff, Michele I-76  
 Mohamed, Khalil Ben II-330  
 Moon, Yang-Sae II-17  
 Mugnier, Marie-Laure II-330  
 Mungure, Ester M. I-137  
 Muntés-Mulero, Victor I-231  
 Muro-Medrano, Pedro R. I-495  
  
 Nakamura, Satoshi II-258  
 Nakano, Miyuki I-241  
 Nejdil, Wolfgang I-152, II-240  
 Nicklas, Daniela I-261  
 Nieves, Javier I-213  
 Nishikawa, Norifumi I-241  
  
 Obwegeser, Nikolaus I-221  
 Oelze, Irina II-240  
 Okariz, Ana II-185  
 Okorodudu, Anthony II-452  
 Ott, Maximilian I-512  
 Ou, Yi I-183  
  
 Parapar, Javier II-407  
 Pardede, Eric I-331  
 Parisi, Francesco II-285  
 Park, Jaehui II-346, II-422  
 Park, Nam Hun I-281  
 Parthasarathy, Srinivasan II-209  
 Pastor, Oscar I-467  
 Patel, Dhaval I-416, II-209  
 Pedersen, Torben Bach I-137, II-32,  
 II-111  
 Pehlivan, Zeynep I-1  
 Peng, Huanchun I-385  
 Playa, Yoseba K. I-213  
 Pérez, María I-62  
 Pomares, Alexandra I-46  
 Proietti, Maurizio I-76  
 Pudi, Vikram II-168  
  
 Qiao, Jianzhong I-167  
 Qi, Yinian II-300  
  
 Raghavan, Venkatesh I-281  
 Rahayu, Wenny I-331  
 Riazati, Dariush II-96



- Roantree, Mark II-462  
 Roncancio, Claudia I-46  
 Ruiz, Iraide II-185  
 Rundensteiner, Elke A. I-281  
  
 Saito, Akinori II-277  
 Santos, Igor I-213, II-185  
 Sanz, Ismael I-62  
 Saraswat, Mala II-360  
 Schall, Daniel I-183  
 Schikuta, Erich I-315  
 Schulte, Jonas I-315  
 Schunk, Lars K. II-267  
 Seid, Dawit I-340  
 Seneviratne, Aruna I-512  
 Servigne, Sylvie II-159  
 Shahabi, Cyrus I-432, I-450  
 Shao, Jianhua I-400  
 Shekhar, Shashi II-149  
 Shiblee Sadik, Md. I-122  
 Shimizu, Toshiyuki II-277  
 Shim, Junho II-422  
 Shi, Yan II-80  
 Shou, Lidan I-198  
 Sierra, Carles II-126  
 Silva, Mário J. I-495  
 Singh, Ajit I-251  
 Singh, Himanshu II-168  
 Skočir, Zoran I-470  
 Smith, Fabrizio I-76  
 Śnieżyński, Bartłomiej II-177  
 Sørensen, Kenneth I-137  
 Stark, Konrad I-315  
 Stix, Volker I-221  
 Straccia, Umberto I-31  
 Subieta, Kazimierz I-323  
 Su, Hui Zhu I-105  
  
 Takasu, Atsuhiko II-141  
 Tanaka, Katsumi II-258  
 Taniar, David I-331  
 Tao, Yue I-298  
 Tatedoko, Masashi II-277  
 Tchounikine, Anne II-159  
  
 Tezuka, Taro II-248  
 Thi Le, Dung Xuan I-331  
 Thom, James A. II-96  
 Tserendorj, Tuvshintur I-503  
 Tsukuda, Kosetsu II-258  
 Tu, Yicheng II-485  
  
 Uday Kiran, R. II-194  
  
 Villamil, María-del-Pilar I-46  
  
 Wang, En Tzu I-105  
 Wang, Fan II-64  
 Wang, Wei I-298  
 Wang, Xiansheng I-298  
 Wang, Xing I-16, I-478  
 Wang, Yanqiu I-167  
 Watanabe, Akitsugu II-49  
 Wolski, Antoni I-231  
 Wrembel, Robert II-315  
 Wu, Huayu II-391  
 Wu, Weili II-80  
  
 Xia, Yuni II-485  
 Xu, Chang I-198  
 Xu, Chuanfei I-167  
 Xu, Liang II-391  
  
 Yamamoto, Takehiro II-258  
 Yan, Li I-478, I-487  
 Ye, Xiaojun I-385  
 Yokota, Haruo II-49  
 Yoshikawa, Masatoshi II-1, II-277  
  
 Zhang, Chongsheng I-91  
 Zhang, Fu I-16, I-478  
 Zhang, Jiaqi I-298  
 Zhang, Xinpeng II-1  
 Zhang, Xiuzhen II-96  
 Zhang, Yongfa II-375  
 Zhong, Jiaofei II-80  
 Zhou, Xuan II-240  
 Zubillaga, Agustín II-185  
 Zuzarte, Calisto I-231